<$nopage>ActiveX Automation (interface):proyectos de automatización. <$nopage>Índice de colores de AutoCAD (números). <$nopage>Color (propiedad):<$nopage>TrueColor (propiedad):<$nopage>VB:proyectos de automatización. <$nopage>VBA:proyectos de automatización. <$nopage>Visual Basic para aplicaciones. <$nopage>Visual Basic. Lice<$nopage>proyectos de automatización:migrar a AutoCAD 2004: <$nopage>proyectos:automatización. <$nopage>nombres de colores estándar.

Introducción

En esta introducción se explica el concepto de exposición de objetos de AutoCAD a través de una interfaz de ActiveX Automation y de la programación de estos objetos en el entorno de programación Visual Basic para aplicaciones.

Presentación general de la tecnología ActiveX de AutoCAD

Con AutoCAD® ActiveX® puede manipular AutoCAD mediante la programación desde dentro o fuera de AutoCAD. Este método consiste en exponer objetos de AutoCAD al “mundo exterior.” Una vez expuestos, se facilita el acceso a ellos desde numerosos entornos y lenguajes de programación, así como desde aplicaciones tales como Microsoft® Word VBA o Excel VBA.

ADD AX_INTRO graphic

La inclusión de una interfaz de ActiveX para AutoCAD ofrece dos ventajas:

- El acceso programático a los dibujos de AutoCAD puede realizarse desde muchos otros entornos de programación. Antes de ActiveX Automation, los desarrolladores debían limitarse a una interfaz AutoLISP® o C++.
- La posibilidad de compartir información con otras aplicaciones para Windows®, como Microsoft Excel y Word, se ha facilitado enormemente.

Los objetos constituyen el bloque de integración principal de todas las aplicaciones ActiveX. Cada objeto expuesto representa un componente concreto de AutoCAD. Existen multitud de tipos de objetos diferentes en la interfaz de ActiveX de AutoCAD. Por ejemplo:

- Los elementos gráficos como las líneas, los arcos, el texto y las cotas son objetos.
- Los parámetros de estilo como el tipo de línea y el estilo de cota son objetos.
- Las estructuras de organización como las capas, los grupos y los bloques son objetos.

- Las pantallas de dibujo, como vistas y ventanas gráficas, son objetos.
- Se consideran objetos incluso el dibujo y la aplicación AutoCAD.

¿Comentarios?

Presentación general de la interfaz entre AutoCAD y Visual Basic para aplicaciones (VBA)

Microsoft VBA es un entorno de programación orientado a objetos, concebido para suministrar funciones avanzadas de desarrollo similares a las de Visual Basic 6 (VB). La diferencia principal entre VBA y VB es que el primero se ejecuta en el mismo espacio de proceso que AutoCAD, lo que proporciona un entorno de programación muy rápido y compatible con AutoCAD.

VBA permite también la integración con otras aplicaciones que admiten VBA. Lo que significa que AutoCAD puede, mediante las bibliotecas de objetos de otras aplicaciones, funcionar como controlador de automatización de otras aplicaciones como Microsoft Word o Excel.

Las ediciones de desarrollo independientes de Visual Basic 6, que deben adquirirse por separado, complementan a VBA de AutoCAD con componentes adicionales, como un motor de base de datos externo y funciones de generación de informes.

La inclusión de ActiveX para AutoCAD ofrece cuatro ventajas:

- VBA y su entorno resultan fáciles de aprender y de usar.
- VBA se ejecuta junto con AutoCAD. Esto se traduce en una ejecución de programa muy rápida.
- La construcción de cuadros de diálogos es rápida y eficaz. Esto permite a los desarrolladores realizar prototipos de aplicaciones y recibir información sobre diseños de forma rápida.
- Los proyectos pueden ser independientes o estar incrustados en los dibujos. Esta opción ofrece a los desarrolladores una gran flexibilidad para la distribución de sus aplicaciones.

Nota Microsoft no se compromete a proporcionar bibliotecas SDK de VBA de 64

bits (.dll). Posteriormente, AutoCAD de 64 bits ya no podrá ejecutar VBA como componente durante el proceso; los componentes de VBA se ejecutan ahora como componentes COM fuera de proceso de 32 bits y proporciona una organización temporal para los usuarios de VBA con AutoCAD de 64 bits Puede que esta organización requiera algunos cambios en el código de VBA existente. Esta previsión temporal no se tendrá en cuenta en futuras versiones de AutoCAD y se aconseja a los usuarios que cambien su código VBA existente a uno VB. NET.

- **Cómo está integrado VBA en AutoCAD**

¿Comentarios?

Cómo está integrado VBA en AutoCAD

VBA envía mensajes a AutoCAD mediante la interfaz de ActiveX Automation de AutoCAD. VBA de AutoCAD permite la ejecución simultánea de AutoCAD y el entorno VBA, y proporciona un control por programación de AutoCAD mediante la interfaz de ActiveX Automation. Esta cooperación entre AutoCAD, ActiveX Automation y VBA constituye una interfaz muy avanzada, no sólo para manipular objetos de AutoCAD sino también para enviar o recuperar datos de otras aplicaciones.

Existen tres elementos fundamentales que definen la programación ActiveX y VBA en AutoCAD. El primero es el propio AutoCAD, que incluye un amplio conjunto de objetos que engloba entidades, datos y comandos de AutoCAD. Puesto que AutoCAD está diseñado como una aplicación de arquitectura abierta, con multitud de niveles de interfaz, el uso eficaz de VBA requiere un cierto grado de familiaridad con la programación en AutoCAD. Si tiene experiencia en el uso de AutoLISP para el control mediante programación de AutoCAD, ya posee conocimientos suficientes de las funciones de AutoCAD. No obstante, el planteamiento de VBA, basado en objetos, es muy diferente del de AutoLISP.

El segundo elemento es la interfaz ActiveX Automation de AutoCAD, que establece mensajes (comunicación) con los objetos de AutoCAD. La programación en VBA requiere un conocimiento fundamental de ActiveX Automation. Puede encontrar una descripción de la interfaz de ActiveX Automation de AutoCAD en *ActiveX and VBA Reference*. Incluso los programadores avanzados de VB encontrarán en ActiveX Automation de AutoCAD una interfaz de valor incalculable para conocer y desarrollar aplicaciones AutoCAD VBA.

El tercer elemento es el entorno de programación VBA, que cuenta con su propio conjunto de objetos, palabras clave, constantes, etc., los cuales facilitan el flujo de los programas y su control, depuración y ejecución. La propia Ayuda de

Microsoft para VBA está incluida en la Ayuda de AutoCAD VBA y puede accederse a ella desde el IDE de VBA con uno de los siguientes métodos:

- Pulsando la tecla F1
- Eligiendo Ayuda en la barra de menús del IDE de VBA.
- Haciendo clic en el signo de interrogación de la barra de herramientas del IDE de VBA.
- **Uso de Microsoft .NET Framework**
- **Requisitos y restricciones**

Uso de Microsoft .NET Framework

Para tener un acceso completo a los objetos de automatización de AutoCAD desde Microsoft Visual Studio® .NET, cree referencias a los siguientes archivos:

- La biblioteca de tipos de AutoCAD 2008, *acax17enu.tlb*, ubicada en *c:\Archivos de programa\Archivos comunes\Autodesk Shared.*
- La biblioteca de tipos de AutoCAD/ObjectDBX Common 17.0, *axdb17enu.tlb*, ubicada en *c:\Archivos de programa\Archivos comunes\Autodesk Shared.*

Estas referencias le permitirán tener disponibles los siguientes conjuntos primarios de interoperabilidad: Autodesk.AutoCAD.Interop.dll (para tipos específicos de AutoCAD), y Autodesk.AutoCAD.Interop.Common.dll (para tipos compartidos mediante las aplicaciones huesped ObjectDBXTM). Los ensamblajes de interoperabilidad se encuentran en la caché del ensamblaje global y asignan objetos de automatización a homólogos de .NET.

Después de crear las referencias a las bibliotecas de tipos, puede declarar variables basadas en AutoCAD en Microsoft Visual Studio .NET, como en los ejemplos siguientes:

```
Dim objAcad As Autodesk.AutoCAD.Interop.AcadApplication
Dim objLine As Autodesk.AutoCAD.Interop.Common.AcadLine
```

Puede cargar una aplicación .NET utilizando el comando NETLOAD de AutoCAD.

Hay disponible información adicional sobre el uso de una aplicación .NET con AutoCAD en la sección Developer Center del sitio Web de Autodesk

¿Comentarios?

Requisitos y restricciones

Si instala, reinstala, o desinstala Microsoft Office u otras aplicaciones VBA después de instalar AutoCAD, reinstale AutoCAD y reinicie el sistema.

Uso combinado de ActiveX y VBA en AutoCAD

La interfaz de ActiveX/VBA de AutoCAD presenta varias ventajas sobre otros entornos API de AutoCAD:

- **Velocidad**. Cuando se realiza una ejecución en proceso con VBA, las aplicaciones de ActiveX son más rápidas que las de AutoLISP.
- **Facilidad de uso.**El lenguaje de programación y el entorno de desarrollo son fáciles de usar y vienen instalados con AutoCAD.
- **Funcionamiento conjunto con Windows.** ActiveX y VBA están diseñados para su uso con otras aplicaciones de Windows y constituyen una excelente vía para la comunicación de información entre distintas aplicaciones.
- **Rápida creación de prototipos.** La capacidad de VBA para un rápido desarrollo de interfaces ofrece un entorno óptimo para la creación de aplicaciones prototipo, incluso si dichas aplicaciones en última instancia deberán ser desarrolladas en otro lenguaje.
- **Base para programadores.** La tecnología VBA y ActiveX de AutoCAD proporciona a los programadores de Visual Basic 6 la capacidad de personalizar AutoCAD y de desarrollar aplicaciones compatibles.

Organización de este manual

En este manual se ofrece información relativa al desarrollo de aplicaciones de ActiveX y VBA para uso con AutoCAD. En “Para empezar con VBA” y “Desarrollo de aplicaciones con VBA” se ofrece información específica sobre el desarrollo de aplicaciones con VBA.Los programadores que utilicen ActiveX en un entorno de desarrollo que no sea VBA pueden omitir estos dos capítulos. No obstante, hay que tener en cuenta que el código de los ejemplos de este manual está escrito en VBA.

Para obtener más información

En este manual se supone que se tienen conocimientos para trabajar con el lenguaje de programación Visual Basic 6. No se intenta duplicar o remplazar la abundante documentación disponible sobre Visual Basic 6. Si necesita más información acerca del uso del entorno de desarrollo o el lenguaje Visual Basic 6, consulte el archivo de Ayuda de Visual Basic para Aplicaciones desarrollado por Microsoft, disponible en el menú Ayuda del entorno de desarrollo interactivo (IDE).

Código de ejemplo

Este manual y *ActiveX and VBA Reference* juntos contienen más de 800 subrutinas de ejemplo de VBA que constituyen una demostración del uso de los métodos, propiedades y eventos de ActiveX.

AutoCAD también proporciona multitud de aplicaciones de ejemplo en el directorio *Sample* de AutoCAD. Estas aplicaciones demuestran el cometido de una amplia variedad de funciones, desde la extracción de datos de dibujos de AutoCAD a hojas de cálculo Microsoft Excel, hasta el dibujo y el análisis de tensiones en una torre eléctrica de transmisiones.

Estos ejemplos también le permitirán aprender a combinar la flexibilidad del entorno de programación Visual Basic for Applications con la avanzada interfaz de ActiveX de AutoCAD para crear aplicaciones personalizadas.

De manera adicional, el código de ejemplo del *Manual del desarrollador de ActiveX y VBA* y *ActiveX and VBA Reference* se puede copiar desde los archivos de ayuda, pegarla después en el entorno VBA de AutoCAD, y ejecutarla con una condición: el dibujo actual activo en AutoCAD debe ser un dibujo vacío abierto al espacio modelo.

Para ejecutar los ejemplos de los archivos de Ayuda

1. Copie el ejemplo desde el archivo de Ayuda a un módulo de código de VBA vacío.
2. Verifique que AutoCAD tiene abierto un dibujo en blanco en el espacio modelo.
3. Abra el cuadro de diálogo Macros, ejecutando el comando VBARUN.
4. Elija la macro que desee y pulse Ejecutar.

   Hay más información acerca de la ejecución de macros y el cuadro de diálogo Macros disponible en el tema “Ejecutar una macro”.

¿Comentarios?

Migración de proyectos de automatización

Puede utilizar las funciones de AutoCAD mediante los objetos y métodos añadidos a la interfaz de ActiveX Automation. En esta sección se incluyen los cambios que se aplican a los proyectos de automatización creados con Visual Basic para Aplicaciones (VBA), Visual Basic 6 (VB) y otros entornos compatibles con la automatización.

Para obtener más información acerca del uso de funciones en AutoCAD, véase “Utilización de las funciones de AutoCAD”.

- **Objetos nuevos**
- **Elementos modificados**
- **Cómo migrar proyectos**

Objetos nuevos

Los siguientes objetos son nuevos en AutoCAD 2008. Para obtener más información sobre estos objetos, véase *ActiveX and VBA Reference* y el Examinador de objetos en el IDE de VBA.

- **SortentsTable**. Contiene y manipula la información de ordenación de los objetos.
- **Table**. Añade y modifica tablas en un dibujo.
- **TableStyle**. Añade y modifica formato en las tablas, como visibilidad de rejilla, grosor de línea y color.

Además, AutoCAD 2008 contiene objetos para la automatización del Administrador de conjuntos de planos. Para obtener información sobre estos objetos, véase *Referencia a objetos de conjunto de planos*.

Elementos modificados

En esta sección se describen los elementos existentes que han cambiado.

Elementos modificados		
AutoCAD 2004 item	AutoCAD 2008 item	Descripción del cambio
BeginClose (evento)	BeginDocClose (evento)	Puede utilizar el evento BeginDocClose para impedir que un dibujo se cierre.
Layer (objeto)	Layer (objeto)	Adición de la propiedad Description y la propiedad Used.
Colección de capas	Colección de capas	Adición del método GenerateUsageData.
Colección de espacio modelo Colección de espacio papel Block (objeto)	Colección de espacio modelo Colección de espacio papel Block (objeto)	Adición del método AddTable a todas las colecciones y al objeto; adición de la propiedad Path al objeto Block
MText (objeto), Text (objeto)	MText (objeto), Text (objeto)	Adición de la propiedad BackgroundFill al objeto MText; adición del método FieldCode al

		objeto MText y al objeto Text.
Plot (objeto)	Plot (objeto)	La propiedad BatchPlotProgress y la propiedad StartBatchMode han quedado obsoletas. Se recomienda el uso de aplicaciones Microsoft .NET para el trazado por lotes. El método DisplayPlotPreview ya no admite la vista preliminar parcial.
PreferencesFiles (objeto)	PreferencesFiles (objeto)	Adición de la propiedad PlotLogFilePath, la propiedad PageSetupOverridesTemplateFile y la propiedad QNewTemplateFile.
PreferencesOutput (objeto)	PreferencesOutput (objeto)	Adición de la propiedad AutomaticPlotLog, la propiedad DefaultPlotToFilePath y la propiedad ContinuousPlotLog.
Viewport (objeto)	Viewport (objeto)	Adición de la propiedad ModelView, la propiedad SheetView, la propiedad LabelBlockId, la propiedad HasSheetView y el método SyncModelView.
View (objeto)	View (objeto)	Adición de la propiedad CategoryName, la propiedad LayoutId, la propiedad LayerState y la propiedad HasVpAssociation.

¿Comentarios?

Cómo migrar proyectos

En general, un proyecto de automatización de AutoCAD creado en el IDE de VBA o con Visual Basic 6 debería funcionar con AutoCAD 2008.

Los proyectos de automatización de AutoCAD 2008 utilizan la misma biblioteca de tipos (acax17enu.tlb) que los proyectos de automatización de AutoCAD. La biblioteca de tipos se encuentra en *C:\Archivos de programa\Archivos comunes\Autodesk Shared*.

Los proyectos de automatización de AutoCAD 2008 también utilizan el mismo ProgID dependiente de la versión para los métodos CreateObject, GetObject y GetInterfaceObject. Por ejemplo, si utiliza la función CreateObject en un proyecto de automatización de AutoCAD, puede usar CreateObject ("AutoCAD.Application.17"). Si un proyecto de automatización utiliza ProgID independientes de la versión, cambie el proyecto para utilizar ProgID dependientes de la versión.

<$nopage>propiedades. <$nopage>objetos:

Para empezar con VBA

En este capítulo se presentan los proyectos Visual Basic para Aplicaciones (VBA) de AutoCAD y el entorno de desarrollo interactivo (IDE) de VBA. Aunque la mayoría de los entornos de VBA tienen un comportamiento semejante, el IDE de VBA de AutoCAD posee algunas características exclusivas. Pueden utilizarse algunos comandos de AutoCAD para cargar y ejecutar proyectos o abrir el IDE de VBA. En este capítulo se describe el uso de proyectos y comandos de VBA, así como del entorno IDE de VBA en general.

Descripción de los proyectos VBA globales e incrustados

Un proyecto VBA de AutoCAD® consiste en un conjunto de módulos de código, módulos de clase y formularios que se combinan para realizar una función determinada. Los proyectos pueden almacenarse en un dibujo de AutoCAD o en un archivo independiente.

Los proyectos incrustados se almacenan dentro de un dibujo de AutoCAD. Estos proyectos se cargan automáticamente al abrir en AutoCAD el dibujo que los contiene, lo cual facilita extraordinariamente la distribución de los proyectos. Los proyectos incrustados están limitados y no pueden abrir ni cerrar dibujos de AutoCAD ya que sólo funcionan en el documento donde residen. Los usuarios de proyectos incrustados ya no tienen que buscar y cargar los archivos de proyecto antes de ejecutar un programa. Un ejemplo de proyecto incrustado en un dibujo lo constituye un registro de tiempo que se activa al abrir el dibujo. Esta macro permite a los usuarios iniciar una sesión y registrar el tiempo que han trabajado en el dibujo. El usuario no necesita acordarse de cargar el proyecto antes de abrir el dibujo, ya que esto se hace de forma automática.

Los proyectos globales se almacenan en archivos independientes y son más versátiles, ya que permiten abrir y cerrar cualquier dibujo de AutoCAD, así como trabajar con él, pero no se cargan de forma automática cuando se abre el dibujo. Para poder ejecutar la macro que necesitan, los usuarios deben saber en qué archivo de proyecto se encuentra y cargar ese archivo. No obstante, los proyectos globales son mucho más fáciles de compartir con otros usuarios, y generan bibliotecas estupendas de macros comunes. Un ejemplo de proyecto que puede almacenarse en un archivo de proyecto lo constituye una macro que recopila una lista de elementos a partir de varios dibujos. La macro puede ser ejecutada por un administrador al terminar el ciclo de trabajo, a fin de reunir información de diversos dibujos.

En cualquier momento, los usuarios pueden tener cargados en su sesión de AutoCAD tanto proyectos incrustados como globales.

&Los proyectos VBA de AutoCAD no son compatibles en formato binario con los proyectos de Visual Basic 6 independientes. No obstante, es posible intercambiar formularios, módulos y clases entre proyectos mediante los comandos IMPORTAR y EXPORTAR VBA del IDE de VBA. Para obtener más información acerca del IDE de VBA, véase Edición de proyectos con el IDE de VBA.

Se admite el uso de Visual Studio .NET para dirigir y personalizar AutoCAD mediante COM Automation.

<$nopage>propiedades. <$nopage>objetos:

Organización de los proyectos con el Administrador de VBA

Puede utilizar el Administrador de VBA para ver todos los proyectos VBA que estén cargados en la sesión actual de AutoCAD. El Administrador de VBA es una herramienta de AutoCAD que permite cargar, descargar, guardar, crear, incrustar y extraer proyectos VBA.

Para abrir el Administrador de VBA

Puede abrir el Administrador de VBA desde el menú Herr. o, en AutoCAD, ejecutando el comando VBAMAN.

- **Carga de un proyecto existente**
- **Descarga de un proyecto**
- **Incrustación de un proyecto en un dibujo**
- **Extracción de un proyecto de un dibujo**
- **Creación de un proyecto**
- **Guardado del proyecto**

<$nopage>propiedades. <$nopage>objetos:

Carga de un proyecto existente

Cuando se carga un proyecto en AutoCAD, todas las subrutinas públicas, también denominadas macros, quedan habilitadas. Los proyectos incrustados en un dibujo se cargan siempre que el dibujo es abierto. Los proyectos guardados en archivos DVB deben cargarse de forma explícita.

En el momento de cargar un proyecto, todos los proyectos a los que haga referencia se cargan de forma automática. De manera adicional, AutoCAD carga automáticamente durante el inicio cualquier archivo de proyecto cuyo nombre sea *acad.dvb*.

Para cargar el archivo de un proyecto VBA existente

1. En el Administrador de VBA, elija la opción Cargar para abrir el cuadro de diálogo Abrir proyecto VBA.
2. En el cuadro de diálogo Abrir, seleccione el archivo de proyecto que desee utilizar. En el cuadro de diálogo Abrir proyecto VBA sólo pueden abrirse archivos DVB válidos. Si intenta abrir un archivo de otro tipo, recibirá un mensaje de error.
3. Elija Abrir.

También puede cargar un archivo de proyecto mediante uno de los métodos siguientes:

- Introduzca el comando VBALOAD, que abre el cuadro de diálogo Abrir proyecto VBA.
- Arrastre un archivo DVB desde el Explorador de Windows y colóquelo en un dibujo abierto en la ventana de AutoCAD.

- **Alerta de virus**

¿Comentarios?

Alerta de virus

Cada vez que se carga un proyecto, se puede activar o desactivar su código interno como protección contra virus. Si activa el código, los virus que éste pueda contener comienzan a ejecutarse. Si desactiva el código, el proyecto se carga pero todo su código queda sin ejecutar. La alerta de protección antivirus no está visible cuando el proyecto se carga arrastrando un archivo DVB desde el Explorador de Windows y colocándolo en un dibujo abierto en la ventana de AutoCAD.

Para obtener más información acerca de la alerta contra virus, véase de opciones de proyectos.

Descarga de un proyecto

Al descargar un proyecto se aumenta la memoria disponible y se agiliza la utilización de la lista de proyectos cargados pues se reduce su longitud.

No es posible descargar proyectos incrustados o proyectos a los que hacen referencia otros proyectos cargados.

Para descargar un proyecto VBA

Puede descargar un proyecto VBA seleccionándolo y eligiendo Descargar, o bien mediante el comando VBAUNLOAD, el cual solicita el proyecto que se desea descargar.

Incrustación de un proyecto en un dibujo

Cuando se incrusta un proyecto se introduce una copia del mismo en la base de datos de dibujos. El proyecto estará cargado o descargado según esté abierto o cerrado el dibujo que lo contiene.

Un dibujo puede contener sólo un proyecto incrustado. Si un dibujo tiene ya un proyecto incrustado, debe extraerlo antes de poder incrustar otro proyecto.

Para incrustar un proyecto en un dibujo de AutoCAD

1. Abra el Administrador de VBA y seleccione el proyecto que desee incrustar.
2. Elija Incluir.

Extracción de un proyecto de un dibujo

Cuando se extrae un proyecto, éste queda eliminado de la base de datos de dibujos. Tiene la posibilidad de guardarlo en un archivo de proyecto externo; si no lo hace, los datos del proyecto se borrarán.

Para extraer un proyecto de un dibujo de AutoCAD

1. Abra el Administrador de VBA y seleccione el dibujo del que desee extraer el proyecto.
2. Elija Extraer.
3. Si desea guardar la información del proyecto en un archivo de proyecto externo, seleccione Sí en la solicitud "¿Desea exportar el proyecto VBA antes de eliminarlo?" Se mostrará el cuadro de diálogo Guardar como, en el que podrá guardar el archivo.

   Si no desea guardar la información del proyecto en un archivo externo, responda No a la solicitud "¿Desea exportar el proyecto VBA antes de eliminarlo?" La información del proyecto se eliminará del dibujo y no se guardará.

Creación de un proyecto

Los proyectos nuevos se crean como proyectos globales no guardados. Una vez creado, puede incrustar el proyecto en un dibujo o guardarlo en un archivo de proyecto.

Para crear un proyecto VBA

1. Abra el Administrador de VBA.
2. Elija Nuevo.

   Se crea un proyecto nuevo con el nombre por defecto *ACADProject*. Si desea cambiar el nombre del proyecto, debe entrar en el IDE de VBA. Para obtener más información acerca de la asignación de nombre a proyectos en el IDE de VBA, véase de nombre al proyecto.

Guardado del proyecto

Los proyectos incrustados se guardan siempre que se guarda el dibujo. Los proyectos globales deben guardarse desde el Administrador de VBA o desde el IDE de VBA

Para guardar un proyecto con el Administrador de VBA

1. Abra el Administrador de VBA y seleccione el proyecto que desee guardar.
2. Pulse Guardar. Se abrirá el cuadro de diálogo Guardar como.
3. Seleccione el nombre de archivo para el proyecto que desee guardar.
4. Pulse Guardar.

Gestión de macros

Una macro es una subrutina (ejecutable) pública. Por lo general, cada proyecto contiene al menos una macro.

- **Uso del cuadro de diálogo Macros**
- **Ejecución de una macro**
- **Edición de una macro**
- **Revisión de una macro paso a paso**
- **de opciones de proyectos**

Uso del cuadro de diálogo Macros

En el cuadro de diálogo Macros puede ejecutar, modificar, suprimir y crear macros, así como definir opciones de proyectos VBA. En el menú Herr. de AutoCAD, elija Macro ▸ Macros, o escriba VBARUN en la solicitud de comando de AutoCAD.

Los nombres de todas las macros dentro del rango válido se muestran en este cuadro de diálogo. Puede cambiar el rango válido en la lista desplegable Macros en: En esta lista se especifican los proyectos o dibujos de las macros mostradas. También puede elegir que se muestren las macros de:

- Todos los dibujos y proyectos
- Todos los dibujos
- Todos los proyectos
- Cualquier dibujo que actualmente se encuentre abierto en AutoCAD
- Cualquier proyecto que actualmente se encuentre cargado en AutoCAD

Al limitar el rango válido se puede controlar cuantos nombres de macro aparecen en la lista. Esto le ayudará en los casos en los que muchas macros estén disponibles en los dibujos y proyectos cargados.

Para crear una nueva macro

1. Abra el cuadro de diálogo Macros e introduzca el nombre de la nueva macro.
2. En la lista desplegable Macros en:, seleccione el proyecto donde desee crear la macro.
3. Elija Crear.

Si ya existe una macro con el nombre que ha especificado, se le preguntará si desea reemplazar la macro existente.

Si selecciona Sí, se borra el código de la macro existente y se crea una nueva macro vacía con el nombre indicado.

Si selecciona No, se accede al cuadro de diálogo Macros donde debe indicar un nuevo nombre para la macro.

Si selecciona Cancelar, se cierra el cuadro de diálogo Macros y no se crea ninguna macro nueva.

Para borrar una macro

1. Abra el cuadro de diálogo Macros y seleccione la macro que desee borrar.
2. Elija Borrar. Se le solicitará que confirme la supresión.
3. En la solicitud, elija Sí para suprimirla o No para cancelar la operación.

Ejecución de una macro

Cuando se activa una macro, se ejecuta el código de la macro dentro del contexto de la sesión actual de AutoCAD. El dibujo que esté activo en el momento de ejecutar la macro se considera el dibujo abierto en el que se deben desarrollar las acciones de la macro. Todas las referencias de VBA al objeto `ThisDrawing (este dibujo)` se refieren al dibujo activo cuando se trata de macros de proyectos globales. En el caso de las macros de proyectos incrustados, el objeto `ThisDrawing` siempre se refiere al dibujo en el que esté incrustada la macro

Para ejecutar una macro desde el cuadro de diálogo Macros

1. Abra el cuadro de diálogo Macros y seleccione la macro que desee ejecutar.
2. Elija Ejecutar.

Para ejecutar una macro desde el IDE de VBA

- En el menú Ejecutar, utilice la opción de menú Ejecutar macro.

  Si no hay ninguna macro ni ningún formulario activos, aparece un cuadro de diálogo donde se puede elegir la macro que se desea ejecutar.

  Si ya hay una macro activa (el cursor está en un procedimiento), se ejecuta esa macro.

Edición de una macro

Al editar una macro se abre el IDE de VBA con la macro elegida abierta en la ventana del código. Para obtener más información acerca de cómo editar macros en el IDE de VBA, véase Edición de proyectos con el IDE de VBA.

Para editar una macro

1. Abra el cuadro de diálogo Macros y seleccione la macro que desee editar.
2. Pulse Editar.

Revisión de una macro paso a paso

Este método inicia la ejecución de la macro y la detiene en la primera línea de código. El IDE de VBA se abre con la macro elegida abierta en la ventana del código, en la línea de ejecución.

Para revisar una macro paso a paso

1. En el cuadro de diálogo Macros, seleccione la macro que desee revisar.
2. Elija Entrar.

de opciones de proyectos

Hay tres opciones que pueden definirse para los proyectos VBA de AutoCAD:

- Activación de Incrustación automática
- Activación de Interrupción en error
- Activación de la protección contra virus de macro

Activación de Incrustación automática

La función de incrustación automática crea un proyecto VBA incrustado para todos los dibujos cuando se abren.

Activación de Interrupción en error

Esta opción permite que se active el modo de interrupción de VBA cuando se encuentre un error. El modo de interrupción supone una suspensión temporal de la ejecución del programa en el entorno de desarrollo interactivo. En el modo de interrupción es posible examinar, depurar, restablecer, ir paso a paso o continuar la ejecución del programa.

Cuando esta opción está activada, los errores no gestionados que se encuentran durante la ejecución de la macro de VBA provocan que se suspenda la ejecución de la macro y que muestre el IDE de VBA en el punto de error de la macro.

Cuando esta opción está desactivada, los errores no localizados que se encuentran durante la ejecución de la macro de VBA generan un mensaje de aviso del error y finalizan la ejecución de la macro.

Activación de la protección contra virus de macro

El mecanismo de protección antivirus muestra un mensaje de advertencia integrado siempre que se abre un dibujo que pueda tener virus de macros.

Para definir las opciones de un proyecto VBA de AutoCAD

1. En el menú Herr., elija Macro ➤ Macros para acceder al cuadro de diálogo correspondiente.
2. En el cuadro de diálogo Macros, elija Opciones para abrir el cuadro de diálogo correspondiente.
3. En el cuadro de diálogo Opciones, seleccione las opciones que desea activar.
4. Pulse Aceptar.

Edición de proyectos con el IDE de VBA

Una vez cargado un proyecto en AutoCAD, es posible editar el código, los formularios y las referencias correspondientes mediante el entorno de desarrollo interactivo de VBA. Desde el IDE de VBA también se pueden depurar y ejecutar proyectos. Una vez abierto, el IDE de VBA permite acceder a todos los proyectos cargados.

Para abrir el IDE de VBA en cualquier momento

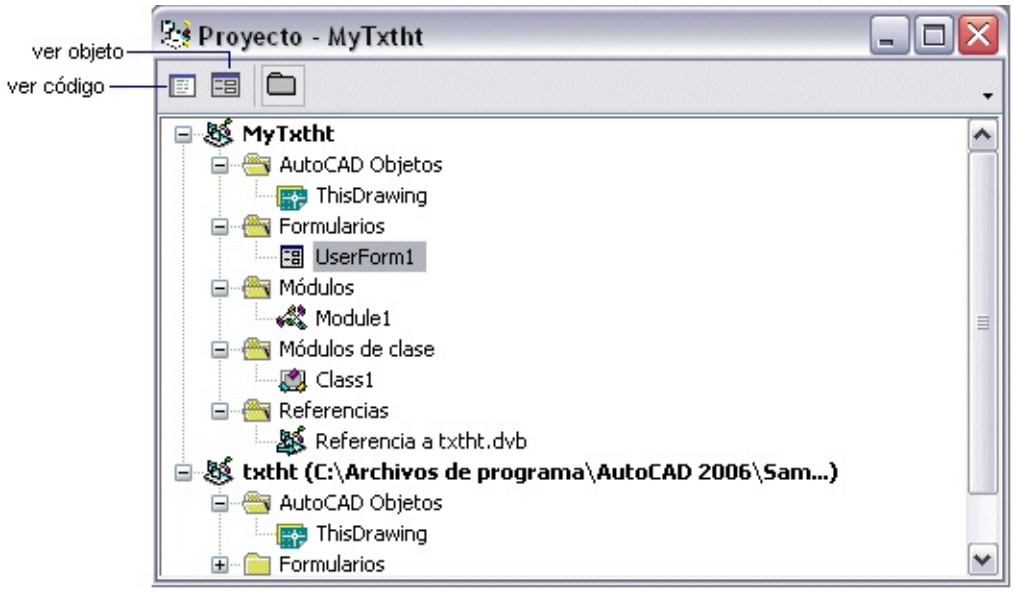
El IDE de VBA puede abrirse desde la línea de comando o desde la barra de menús.

- Introduzca VBAIDE en la línea de comando, o elija, desde el menú Herr., Macro ➤ Editor de Visual Basic.
- **Visualización de información sobre proyectos**
- **Definición de componentes de un proyecto**
- **Importación de componentes existentes**
- **Edición de componentes**
- **de nombre al proyecto**
- **Guardado del proyecto**
- **Referencias a otros proyectos VBA**
- **Definición de opciones del IDE de VBA**

Visualización de información sobre proyectos

El IDE de VBA contiene una ventana, llamada ventana de proyecto, que presenta una lista de todos los proyectos VBA cargados. También presenta los módulos de código, clases y formularios que están incluidos en el proyecto, el documento que tiene asociado, todos los proyectos a los que hace referencia y la ubicación física del proyecto.

La ventana de proyecto tiene su propia barra de herramientas, que puede utilizar para abrir diversos componentes del proyecto y editarlos. Puede abrir el código de un módulo seleccionado pulsando el botón Ver código. Este botón permite mostrar objetos concretos como formularios, por ejemplo.



La ventana de proyecto está visible por defecto. Si no lo está, elija Explorador de proyectos en el menú Ver o pulse CTRL+R.

Definición de componentes de un proyecto

Cada proyecto puede incluir componentes muy distintos. Pueden ser componentes de proyectos los objetos, los formularios, los módulos estándar, los módulos de clases y las referencias.

- **Objetos**
- **Formularios**
- **Módulos estándar**
- **Módulos de clase**
- **Referencias**
- **Adición de componentes nuevos**

Objetos

El componente objeto representa el tipo de objeto, o documento, al que accederá el código de VBA. En los proyectos VBA de AutoCAD, este objeto representa el dibujo actual de AutoCAD.

Formularios

El componente formulario contiene los cuadros de diálogo personalizados que se hayan creado para utilizar con el proyecto.

Módulos estándar

El componente módulo de código contiene los procedimientos y funciones genéricos. Los módulos estándar también se denominan módulos de código, o simplemente módulos.

Módulos de clase

El componente módulo de clase contiene todos los objetos del usuario, que están definidos como clases.

Referencias

El componente referencia contiene todas las referencias a otros proyectos o bibliotecas.

Adición de componentes nuevos

La adición de nuevos componentes crea un componente vacío en el proyecto. Puede añadir nuevos módulos, formularios y módulos de clases al proyecto. El propio usuario debe actualizar todas las propiedades del componente (como el nombre, por ejemplo) y rellenar el código apropiado. Cuando asigne un nombre a un componente nuevo, tenga en cuenta que podrían utilizarlo otros programadores en sus aplicaciones. Siga las convenciones de nomenclatura adoptadas en su equipo de programación.

Para añadir un componente nuevo al proyecto

1. En la ventana de proyecto del IDE de VBA, seleccione el proyecto al que desee añadir el componente.
2. En el menú Insertar, elija UserForm, Módulo o Módulo de clase para añadir el nuevo componente al proyecto.

   El componente nuevo se añade al proyecto y se muestra en la ventana de proyecto.

Importación de componentes existentes

La importación permite añadir al proyecto componentes que ya existen. Puede importar formularios, módulos o módulos de clases. Los formularios se importan como archivos FRM, los módulos como archivos BAS y los módulos de clases como archivos CLS.

Cuando se importa un archivo de componentes, se añade al proyecto una copia del archivo. El archivo original queda intacto. Los cambios que realice en los componentes importados no afectarán al archivo original.

Si se importa un componente con el mismo nombre que uno ya existente, al archivo que se añade al proyecto se le asigna un sufijo numérico.

El componente importado se añade al proyecto y se muestra en la ventana de proyecto. Para modificar las propiedades del componente, selecciónelo en la ventana de proyecto. Las propiedades del componente seleccionado se muestran en una lista y pueden modificarse en la ventana de propiedades.

Para importar un componente existente al proyecto

1. En la ventana de proyecto del IDE de VBA, seleccione el proyecto al que desee añadir el componente.
2. En el menú Archivo, elija Importar archivo para abrir el cuadro de diálogo correspondiente.
3. En este cuadro de diálogo, seleccione el archivo que desee importar y haga clic en Abrir.

Edición de componentes

En el IDE de VBA pueden modificarse los módulos estándar, los módulos de clase y los formularios. Los módulos estándar y de clase se editan en una ventana de código. Los formularios se modifican en la ventana UserForm mediante un cuadro de herramientas especial.

Puede abrir tantas ventanas de código como módulos tenga; de esta forma podrá ver el código de varios formularios o módulos, y copiar y pegar datos entre ellos.

Para editar un componente del proyecto

1. En la ventana de proyecto del IDE de VBA, seleccione el componente que desee editar.
2. Abra una ventana de código pulsando el botón Ver código de la ventana de proyecto.
3. Abra una ventana Userform, con el cuadro de herramientas correspondiente, pulsando el botón Ver objeto de la ventana de proyecto.

Para acceder al código asociado a un formulario

- Para acceder al código asociado a un control, haga doble clic en cualquier control de la ventana de formulario. El código asociado se abre en una ventana de código.
- **Uso de la ventana de código**
- **Uso de la ventana Userform**

Uso de la ventana de código

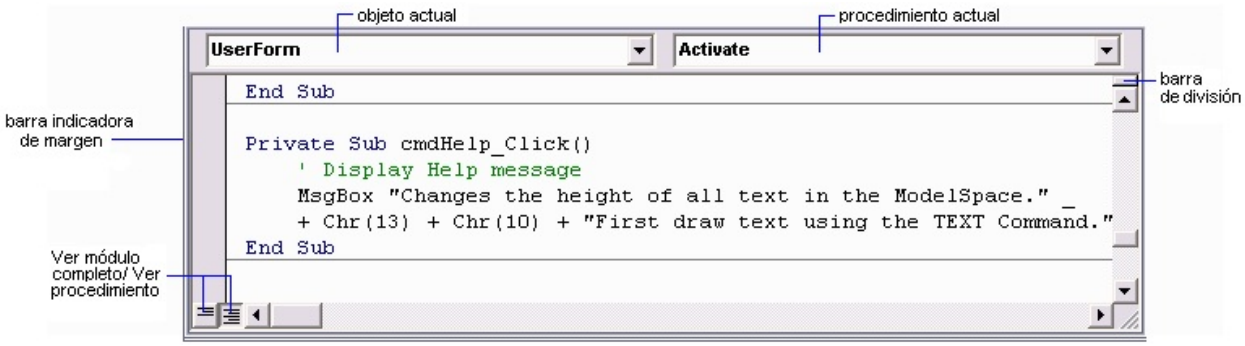
La ventana de código contiene dos listas desplegables, una barra de división, una barra indicadora de margen y los iconos Ver módulo completo y Ver procedimiento.

Las dos listas desplegables de la parte superior de la ventana de código muestran el objeto y el procedimiento actuales. Puede desplazarse por el proyecto si cambia el objeto o el procedimiento en estas listas.

La barra de división situada a la derecha de la ventana de código permite dividir la ventana en sentido horizontal. Sólo tiene que arrastrar la barra hacia abajo para crear otro panel de ventana. Esta función permite ver simultáneamente dos partes del código del mismo módulo. Para cerrar el panel, arrastre la barra de división hasta su posición inicial.

La barra indicadora de margen se encuentra en la parte izquierda inferior de la ventana de código. Puede utilizarla para ver los indicadores de margen que se utilizan durante la modificación y depuración del código.

Los iconos Ver módulo completo y Ver procedimiento se encuentran en la esquina inferior izquierda de la ventana de código y sirven para cambiar la visualización de forma que se muestre un procedimiento cada vez o el módulo completo.



¿Comentarios?

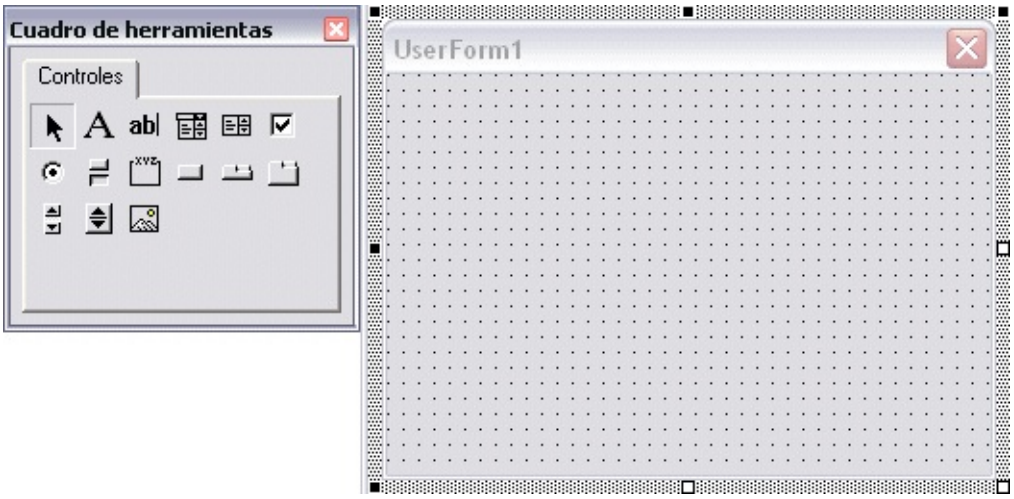
Uso de la ventana Userform

En la ventana Userform puede crear cuadros de diálogo personalizados para un proyecto.

Para añadir un control, basta con arrastrarlo desde el cuadro de herramientas y colocarlo en el formulario. Es posible alinear los controles con la rejilla del formulario desde la ficha General del cuadro de diálogo Opciones. Puede ver la rejilla de formulario y determinar el tamaño de las líneas de la rejilla en la ficha General del cuadro de diálogo Opciones. (Véase Definición de opciones del IDE de VBA para más información sobre el cuadro de diálogo Opciones.)

Cada formulario que se diseña incluye los botones Maximizar, Minimizar y Cerrar de forma automática. Estos botones ya están implementados.

Para añadir un código al control, basta con hacer doble clic en él cuando se haya colocado en el formulario. De esta forma se abre la ventana de código del control.



de nombre al proyecto

El nombre del proyecto y el nombre del archivo *.dvb* donde se guarda el proyecto son distintos. El nombre del archivo *.dvb* se asigna al guardar el proyecto. El nombre del proyecto se define en la ventana de propiedades del IDE de VBA.

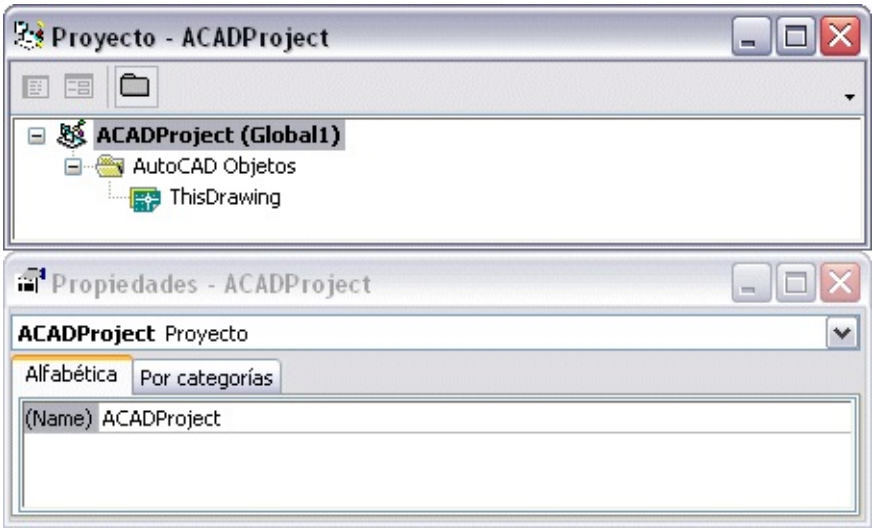
Si no designa un nombre de proyecto y un nombre de archivo, AutoCAD proporciona los siguientes nombres por defecto:

Nombre del proyecto: *ACADProject*

Nombre del archivo: *Project.dvb*

Para cambiar el nombre de un proyecto

1. En la ventana de proyecto del IDE de VBA, seleccione el proyecto que desee cambiar.
2. En la ventana de propiedades, cambie la propiedad Nombre del proyecto.



Para cambiar el nombre de archivo de un proyecto

1. En el IDE de VBA, elija la opción Guardar del menú Archivo.
2. En el cuadro de diálogo Guardar como, escriba el nuevo nombre y la ubicación del archivo de proyecto.

Guardado del proyecto

AutoCAD no tiene un comando GUARDAR específico para los proyectos VBA. En lugar de ello, el comando GUARDAR está incluido en el menú Archivo del IDE y en el Administrador de VBA. Cuando se realicen cambios en un proyecto VBA se abrirá el cuadro de diálogo Guardar proyecto VBA si se produce una de estas situaciones:

- Ha seleccionado el comando GUARDAR en el IDE de VBA.
- Ha elegido la opción Guardar como en el Administrador de VBA.
- La sesión de AutoCAD va a terminar o a cerrarse, pero no se ha guardado el proyecto VBA.

Nota Antes de guardar un proyecto, se le asigna el nombre de archivo por defecto *project.dvb*. Es importante que asigne otro nombre al proyecto cuando lo guarde. Si lo guarda con el nombre de archivo *project.dvb*, ya no podrá crear proyectos nuevos vacíos: cada vez que cree un proyecto nuevo, se cargará el proyecto *project.dvb* guardado.

Referencias a otros proyectos VBA

Las referencias a un proyecto VBA desde otro facilitan a los programadores el uso compartido del código. Los programadores pueden crear bibliotecas con las macros de uso más frecuente y hacer referencia a las bibliotecas cuando las necesiten. Esto permite que el código compartido esté centralizado y revisado y que pueda ser utilizado por un gran número de programadores.

Cuando haya realizado con éxito una referencia a otro proyecto, observará que se crea una carpeta en la ventana de proyecto del IDE de VBA. Esta nueva carpeta se llama *Referencias* y contiene el nombre del proyecto al que se hace referencia.

Una vez creada la referencia a un proyecto, puede utilizar cualquier componente de código o de formulario público en ese proyecto.

Cuando se carga en AutoCAD un proyecto que utiliza referencias a otro proyecto, éste último también se carga en AutoCAD de forma automática. El proyecto de referencia no puede cerrarse hasta que no esté cerrado el proyecto principal.

No pueden establecerse referencias circulares. Es decir, no se puede hacer referencia a un proyecto que a su vez tenga una referencia al primer proyecto. Si se crea una referencia circular por error, VBA lo comunica.

La utilización de referencias a proyectos es una función estándar de Microsoft VBA. En AutoCAD se utiliza tal cual, no se ha realizado ninguna acción para ampliarla. Para obtener más información acerca de la utilización de referencias a proyectos, véase la Ayuda de Microsoft VBA. La Ayuda de Microsoft VBA se puede abrir desde el menú Ayuda del IDE de VBA.

Nota No pueden crearse referencias a proyectos incrustados ni a proyectos VBA de otras aplicaciones.

Para crear una referencia a otro proyecto VBA

1. En la ventana de proyecto del IDE de VBA, seleccione el proyecto al que desee añadir la referencia.
2. En el menú Herramientas, elija la opción Referencias para abrir el cuadro de diálogo correspondiente.
3. En este cuadro de diálogo, haga clic en Examinar para abrir el cuadro de diálogo Agregar referencias.
4. En el cuadro de diálogo, seleccione el archivo de proyecto al que desee hacer referencia y haga clic en el botón Abrir.
5. En este cuadro de diálogo, seleccione el archivo de proyecto al que desee hacer referencia y haga clic en el botón Abrir.

Definición de opciones del IDE de VBA

Es posible cambiar las características del IDE de VBA en el cuadro de diálogo Opciones. Para abrir este cuadro de diálogo, en el menú Herramientas, elija Opciones.

El cuadro de diálogo Opciones incluye cuatro pestañas: Editor, Formato del editor, General y Acoplar.

- **Editor**
- **Formato del editor**
- **General**
- **Acoplar**

Editor

La ficha Editor permite definir los parámetros de las ventanas de código y de proyecto.

Entre los parámetros de código se incluyen:

- Comprobación de sintaxis automática
- Declaración de variable requerida:
- Lista de miembros automática
- Información rápida automática
- Sugerencias de datos automáticas
- Sangría automática
- Ancho de tabulación

Entre los parámetros de ventana se incluyen:

- Arrastrar y colocar en el editor
- Vista completa predeterminada del módulo
- Separador de procedimientos

Formato del editor

La ficha Formato del editor determina el aspecto del código.

Es posible:

- Cambiar el color del código
- Designar los elementos de la lista de texto
- Cambiar el primer plano
- Cambiar el fondo
- Cambiar los indicadores de margen
- Cambiar el tipo de letra del texto y su tamaño
- Mostrar u ocultar el indicador de margen
- Mostrar u ocultar el texto de ejemplo de los parámetros

General

La ficha General precisa los valores, la detección y recuperación de errores y los parámetros de compilación del proyecto VBA actual.

Es posible:

- Cambiar los parámetros de la rejilla del formulario
- Mostrar u ocultar informaciones sobre herramientas
- Definir la ocultación automática de las ventanas
- Elegir la recepción de notificaciones de pérdida de estado
- Determinar el método de gestión de errores
- Definir el proyecto de forma que se compile cuando el usuario lo solicite o se realicen compilaciones en segundo plano

Acoplar

Esta ficha permite elegir las ventanas que pueden acoplarse.

Realización de un ejercicio preliminar

Ahora que ya ha conoce los aspectos básicos de la programación en VBA de AutoCAD, vamos a crear un sencillo ejercicio denominado “Hola a todos”. En este ejercicio va a crear un dibujo de AutoCAD nuevo, va a añadirle una línea de texto y va a guardarlo, todo ello desde VBA.

Para crear el objeto de texto “Hola a todos”

1. Abra el IDE de VBA ejecutando el siguiente comando desde la línea de comando de AutoCAD:

   Comando: **VBAIDE**

2. Abra la ventana de código seleccionando la opción Código del menú Ver en el IDE de VBA.

3. Cree un procedimiento nuevo en el proyecto seleccionando la opción Procedimiento en el menú Insertar en el IDE de VBA.

4. Cuando se le solicite la información del procedimiento, escriba un nombre, por ejemplo, **HolaATodos**. Asegúrese de que estén seleccionados el tipo Procedimiento y el ámbito Público.

5. Pulse Aceptar.

6. Escriba el código siguiente (que abre un dibujo nuevo) entre las líneas `Public Sub Hola a todos()` y `End Sub`.

   ```
   ThisDrawing.Application.Documents.Add
   ```

7. Escriba el código siguiente (que crea la cadena de texto y define el punto donde se inserta) inmediatamente después del código introducido en el paso 6.

```
Dim insPoint(0 To 2) As Double 'Declare insertion point
Dim textHeight As Double 'Declare text height
Dim textStr As String 'Declare text string
Dim textObj As AcadText 'Declare text object
insPoint(0) = 2 'Set insertion point x coordinate
insPoint(1) = 4 'Set insertion point y coordinate
insPoint(2) = 0 'Set insertion point z coordinate
textHeight = 1 'Set text height to 1.0
textString = "Hello, World." 'Set the text string
'Create the Text object
Set textObj = ThisDrawing.ModelSpace.AddText _
                        (textStr, insPoint, textHeight)
```

8. Escriba el siguiente código (que guarda el dibujo) inmediatamente después del código introducido en el paso 7.

```
ThisDrawing.SaveAs("Hello.dwg")
```

9. Ejecute el programa seleccionando la opción Ejecutar Sub/UserForm en el menú Ejecutar del IDE de VBA.

   Cuando termine la ejecución del programa, traiga la aplicación AutoCAD al primer plano. Debería ver el texto “Hola a todos” en el dibujo. El nombre del dibujo debe ser *Hello.dwg*.

Obtención de más información

Puede obtener más información acerca del IDE de VBA y del lenguaje de programación VBA en los archivos de Ayuda que proporciona Microsoft. Para abrir los archivos de Ayuda de Microsoft Visual Basic, elija Ayuda de Microsoft Visual Basic en el menú Ayuda del IDE de VBA.

Términos para proyectos VBA de AutoCAD

Proyecto global

Proyecto VBA guardado en un archivo *.dvb*.

Proyecto incrustado

Proyecto de VBA almacenado en un dibujo de AutoCAD.

Documento normal

Dibujo de AutoCAD que no contiene proyectos VBA incrustados.

Documento informativo

Dibujo de AutoCAD que contiene uno o más proyectos VBA incrustados.

Proyecto actual

Proyecto que se encuentra seleccionado en el IDE de VBA

ThisDrawing

Término de programación de VBA utilizado para representar el dibujo actual. En los proyectos globales, ThisDrawing siempre hace referencia al documento que está activo en AutoCAD. En los proyectos incrustados, ThisDrawing siempre hace referencia al documento que contiene el proyecto.

IDE de VBA

Entorno de desarrollo interactivo de VBA. Esta aplicación permite editar el código y los formularios del proyecto activo, o copiar código y formulario de otros proyectos. También permite establecer referencias a otros modelos de objetos de aplicación.

Administrador de VBA

El Administrador de VBA permite gestionar los proyectos. Es posible crear, borrar, incrustar o extraer proyectos. También permite ver los proyectos

incrustados en un dibujo abierto si los hay.

Cuadro de diálogo Macros

En el cuadro de diálogo Macros puede ejecutar, suprimir y crear macros, así como acceder a las opciones de proyectos VBA.

¿Comentarios?

Comandos VBA de AutoCAD

VBAIDE

Abre el IDE de VBA.

El IDE de VBA permite editar, ejecutar y depurar programas de forma interactiva. Aunque sólo se puede activar el IDE de VBA mientras se ejecuta AutoCAD, es posible minimizarlo, abrirlo y cerrarlo con independencia de la ventana de aplicación de AutoCAD.

VBALOAD

Carga un proyecto VBA en la sesión actual de AutoCAD.

VBARUN

Ejecuta una macro de VBA desde el cuadro de diálogo Macros o desde la línea de comando de AutoCAD.

VBADESCARGAR

Descarga un proyecto VBA de la sesión de AutoCAD actual.

Si el proyecto VBA se ha modificado pero no se ha guardado, se pregunta al usuario si desea guardarlo mediante el cuadro de diálogo Guardar proyecto (o mediante el equivalente de la línea de comando).

VBAMAN

Muestra el Administrador de VBA, donde puede ver, crear, cargar, cerrar, incrustar y extraer proyectos.

VBAENUN

Ejecuta una secuencia VBA desde la línea de comando de AutoCAD.

<$nopage>datosex. <$nopage>raíz (objeto):

Conceptos básicos de ActiveX Automation

Para hacer un uso eficaz de ActiveX Automation de AutoCAD, debe estar familiarizado con los objetos, entidades y funciones de AutoCAD relativos al tipo de aplicación que esté desarrollando. Cuanto más conocimiento se tenga de las propiedades gráficas y no gráficas de un objeto, más fácil será manipular los objetos en ActiveX Automation de AutoCAD.

Si desea consultar un archivo de ayuda de ActiveX Automation de AutoCAD, pulse F1. Si tiene algún problema con un objeto, método o propiedad concretos, resalte el elemento en cuestión en el IDE de VBA y pulse F1.

- **Modelo de objetos de AutoCAD**
- **Acceso a la jerarquía de objetos**
- **Los objetos de colección**
- **Conceptos básicos sobre propiedades y métodos**
- **Definición de objetos anteriores (padre)**
- **Localización de la biblioteca de tipos**
- **Uso de variantes en métodos y propiedades**
- **Otros lenguajes de programación**

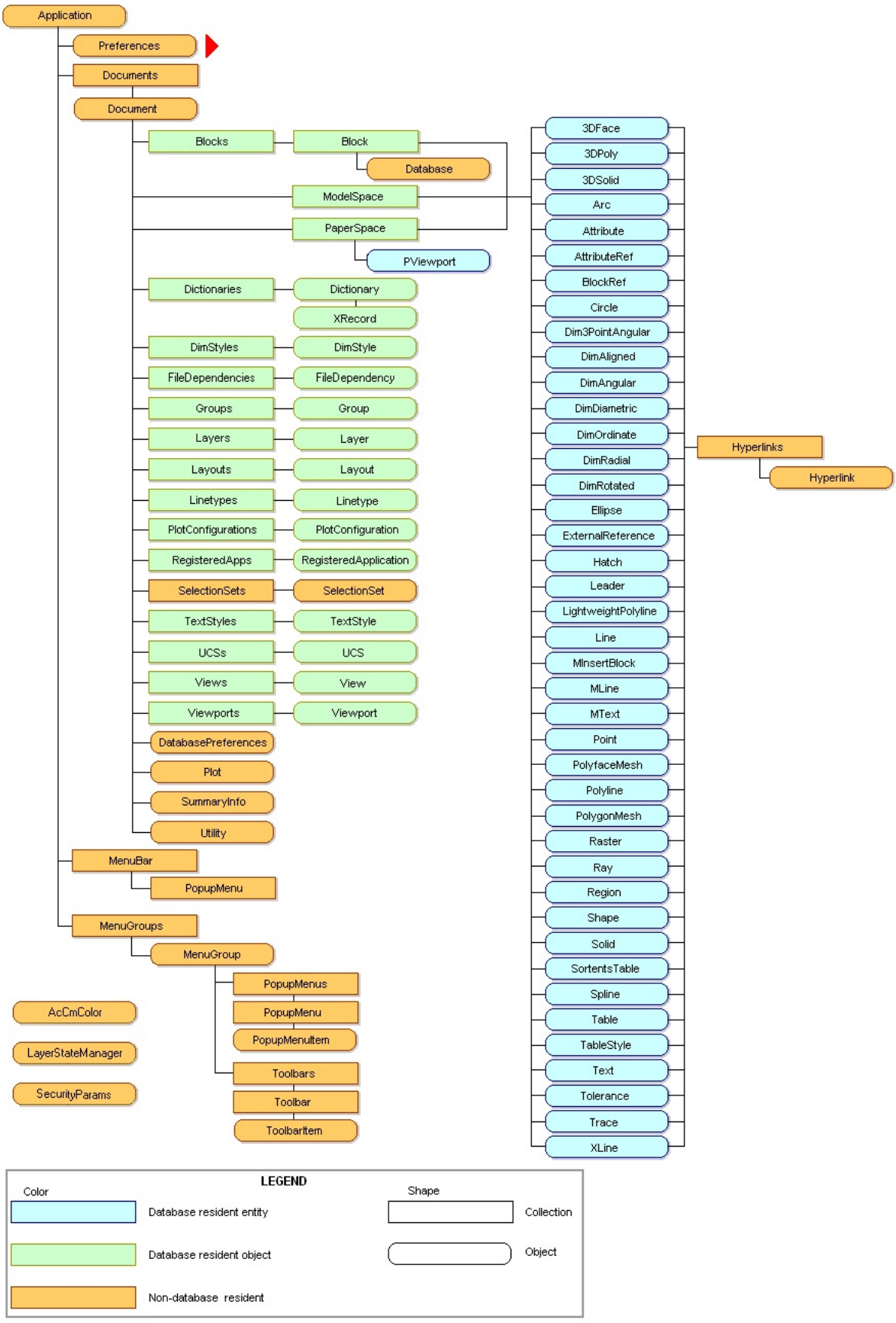
<$nopage>datosex.

Modelo de objetos de AutoCAD

Los objetos constituyen el bloque constructor principal de la interfaz de ActiveX® de AutoCAD®. Cada objeto expuesto representa un componente concreto de AutoCAD. Existen multitud de tipos de objetos diferentes en la interfaz de ActiveX de AutoCAD. Por ejemplo:

- Los elementos gráficos como las líneas, los arcos, el texto y las cotas son objetos.
- Los parámetros de estilo como el tipo de línea y el estilo de cota son objetos.
- Las estructuras de organización como las capas, los grupos y los bloques son objetos.
- Los tipos de visualización de dibujos como las vistas y las ventanas gráficas son objetos.
- Se consideran objetos incluso el dibujo y la aplicación AutoCAD.

Los objetos se estructuran de forma jerárquica, siendo la raíz el objeto Application. A la presentación de esta estructura jerárquica se le denomina Modelo de objetos. El Modelo de objetos permite ver el objeto que proporciona acceso al siguiente nivel de objetos.


Application
Preferences
Documents
Document
Blocks
Block
Database
ModelSpace
PaperSpace
PViewport
Dictionaries
Dictionary
XRecord
DimStyles
DimStyle
FileDependencies
FileDependency
Groups
Group
Layers
Layer
Layouts
Layout
Linetypes
Linetype
PlotConfigurations
PlotConfiguration
RegisteredApps
RegisteredApplication
SelectionSets
SelectionSet
TextStyles
TextStyle
UCSs
UCS
Views
View
Viewports
Viewport
DatabasePreferences
Plot
SummaryInfo
Utility
MenuBar
PopupMenu
MenuGroups
MenuGroup
PopupMenus
PopupMenu
PopupMenuItem
Toolbars
Toolbar
ToolbarItem
AcCmColor
LayerStateManager
SecurityParams
3DFace
3DPoly
3DSolid
Arc
Attribute
AttributeRef
BlockRef
Circle
Dim3PointAngular
DimAligned
DimAngular
DimDiametric
DimOrdinate
DimRadial
DimRotated
Ellipse
ExternalReference
Hatch
Leader
LightweightPolyline
Line
MInsertBlock
MLine
MText
Point
PolyfaceMesh
Polyline
PolygonMesh
Raster
Ray
Region
Shape
Solid
SortentsTable
Spline
Table
TableStyle
Text
Tolerance
Trace
XLine
Hyperlinks
Hyperlink
LEGEND
Color
Database resident entity
Database resident object
Non-database resident
Shape
Collection
Object

- **El objeto Application**
- **El objeto Document**
- **Objetos de colección**
- **Objetos gráficos y no gráficos**
- **Objetos Preferences, Plot y Utility**
- **Uso de las funciones de AutoCAD nuevas**

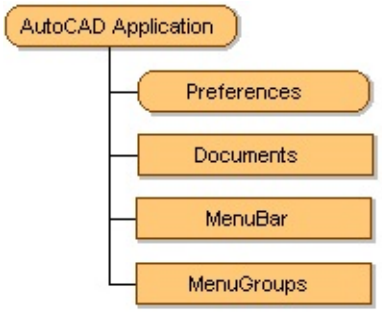
¿Comentarios?

El objeto Application

El objeto Application es la raíz del Modelo de objetos de ActiveX Automation de AutoCAD. Desde él puede obtener acceso a cualquiera de los demás objetos o a los métodos y propiedades que tengan asignados.

Por ejemplo, el objeto Application tiene una propiedad Preferences que devuelve el objeto Preferences. Estos objetos juntos proporcionan acceso a todas las opciones almacenadas en el registro mediante el cuadro de diálogo Opciones. (Los parámetros almacenados en el dibujo están contenidos en el objeto DatabasePreferences que se describe más adelante.) Otras propiedades del objeto Application ofrecen acceso a datos específicos de la aplicación, como su nombre y versión, así como el tamaño, ubicación y visibilidad de AutoCAD. Los métodos del objeto Application permiten realizar acciones específicas de la aplicación, como enumeración, carga y descarga de aplicaciones ADS y ARX, así como salir de AutoCAD.

El objeto Application también proporciona vínculos con los dibujos de AutoCAD a través de la colección Documents, con los menús y las barras de herramientas de AutoCAD a través de las colecciones MenuBar y MenuGroups, y con el IDE de VBA a través de una propiedad llamada VBE.



El objeto Application también es el objeto global de la interfaz de ActiveX. Es decir, todos los métodos y propiedades del objeto Application están disponibles

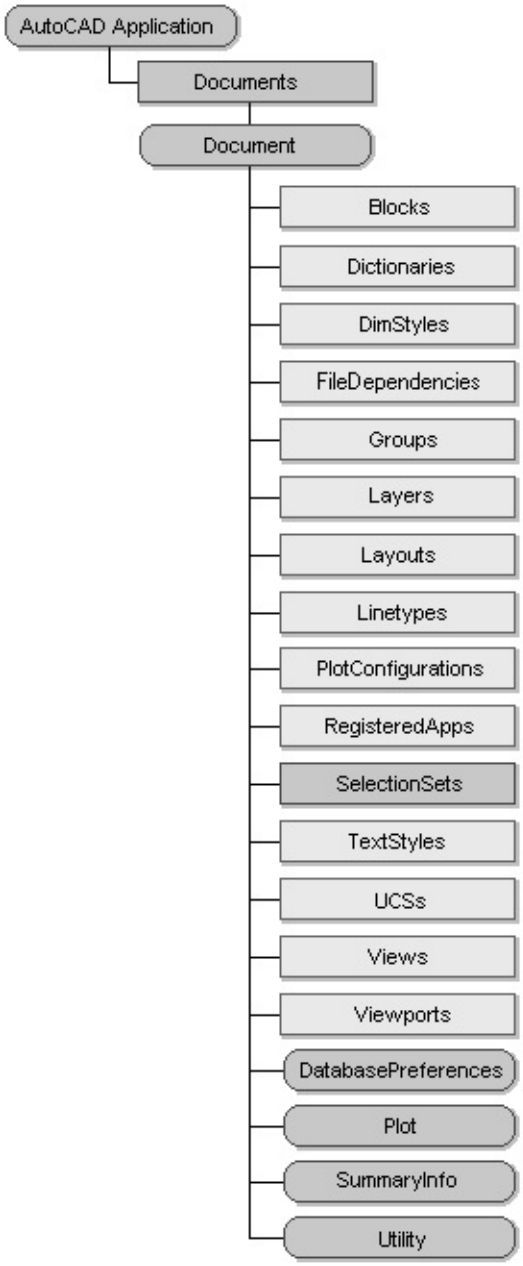
en el espacio de nombres global.

¿Comentarios?

El objeto Document

El objeto Document, que en realidad es un dibujo de AutoCAD, se encuentra en la colección Documents y proporciona acceso a todos los objetos de AutoCAD gráficos y a la mayoría de los que no son gráficos. El acceso a los objetos gráficos (líneas, círculos, arcos, etc.) se realiza a través de las colecciones ModelSpace y PaperSpace, mientras que el acceso a los objetos no gráficos (capas, tipos de línea, estilos de texto, etc.) se realiza a través de colecciones del mismo nombre, como as Layers, Linetypes y TextStyles. El objeto Document también proporciona acceso a los objetos Plot y Utility.

Para acceder a las propiedades del dibujo, utilice la propiedad SummaryInfo del objeto Document.



¿Comentarios?

Objetos de colección

AutoCAD agrupa la mayoría de los objetos en colecciones. Aunque éstas contienen tipos de datos distintos, pueden procesarse aplicando técnicas similares. Cada colección tiene su método de incorporación de nuevos objetos. La mayoría utiliza el método Add. No obstante, los objetos entidad normalmente se añaden mediante un método llamado Add<nombre_entidad>. Por ejemplo, para agregar una línea se utilizaría el método AddLine.

Las colecciones también comparten algunos otros métodos y propiedades. Puede utilizarse la propiedad Count para obtener el total a partir de cero de los objetos de una colección. Con el método Item puede obtenerse cualquier objeto de una colección.

<$nopage>datosex.

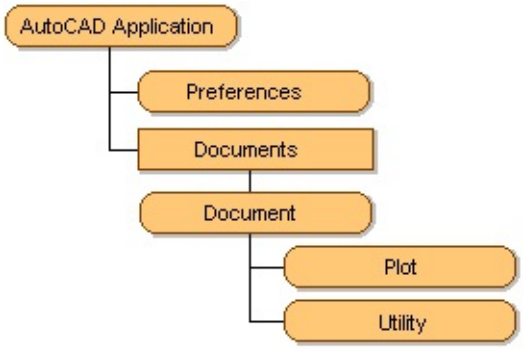
Objetos gráficos y no gráficos

Los objetos gráficos, también conocidos como entidades, son los objetos visibles (líneas, círculos, imágenes ráster, etc.) que componen un dibujo. Para crearlos, se utiliza el método Add<Entidad> apropiado. Para modificar o consultar estos objetos, aplique los métodos o propiedades del propio objeto. Los objetos gráficos tienen métodos que permiten que una aplicación ejecute la mayoría de los comandos de edición de AutoCAD como Copiar, Borrar, Desplazar, Simetría, etc. Estos objetos también tienen métodos para la configuración y recuperación de datos extendidos (datosex), el resaltado y la actualización, y la recuperación del cuadro delimitador del objeto. Los objetos gráficos tienen propiedades básicas como Layer, Linetype, Color y Handle. También tienen propiedades específicas que dependen del tipo de objeto, como Center, Radius y Area.

Los objetos no gráficos son los objetos invisibles (informativos) que forman parte de un dibujo, como Layers, Linetypes, DimStyles, SelectionSets, etc. Para crear estos objetos, aplique el método Add del objeto Collection superior. Para modificar o consultar estos objetos, aplique los métodos o propiedades del propio objeto. Los objetos no gráficos tienen métodos y propiedades específicos a su propósito; todos cuentan con métodos que permiten establecer y recuperar datos extendidos (datosex) y suprimirse a sí mismos.

Objetos Preferences, Plot y Utility

Bajo el objeto Preferences existe un conjunto de objetos, cada uno de los cuales corresponde a una ficha del cuadro de diálogo Opciones. Estos objetos juntos proporcionan acceso a todas las opciones almacenadas en el registro mediante el cuadro de diálogo Opciones. Los parámetros almacenados en el dibujo están contenidos en el objeto DatabasePreferences. Las opciones (y las variables de sistema que no formen parte del cuadro de diálogo Opciones) también pueden establecerse y modificarse con los métodos SetVariable y GetVariable. Para obtener más información acerca de cómo definir opciones, véase Definición de preferencias de AutoCAD .



El objeto Plot proporciona acceso a los parámetros del cuadro de diálogo Imprimir y dota a una aplicación de la capacidad de trazar el dibujo con diversos métodos. Para obtener más información acerca del trazado, véase Impresión de dibujos.

El objeto Utility proporciona al usuario funciones de entrada y conversión de datos. Las funciones de entrada del usuario son métodos que solicitan al usuario la introducción de distintos tipos de datos como cadenas, enteros, reales, puntos y otros, en la línea de comando de AutoCAD. Las funciones de conversión son métodos que utilizan determinados tipos de datos de AutoCAD, como los puntos y ángulos, además de gestionar cadenas y números. Para obtener más

información acerca de las funciones de entrada, véase Solicitud de datos de usuario.

¿Comentarios?

Uso de las funciones de AutoCAD nuevas

Si el proyecto de automatización utiliza una función que no se encontraba en una versión anterior de AutoCAD, debe declararse de forma explícita la interfaz de AutoCAD que se utiliza en el proyecto.

Si un proyecto de automatización contiene declaraciones explícitas de las interfaces nuevas de una determinada versión de AutoCAD, no utilice ese proyecto con versiones anteriores de AutoCAD.

<$nopage>raíz (objeto):

Acceso a la jerarquía de objetos

Es fácil acceder a la jerarquía de objetos desde VBA. El acceso a la jerarquía de objetos es muy sencillo desde VBA, ya que VBA se ejecuta con la sesión actual de AutoCAD en curso y no es preciso por tanto realizar ningún paso adicional para conectar con la aplicación.

VBA proporciona un vínculo al dibujo activo en la sesión actual de AutoCAD a través del objeto `ThisDrawing`. La utilización de `ThisDrawing` permite el acceso inmediato al objeto Document actual y a todos sus métodos y propiedades, así como a todos los demás objetos de la jerarquía.

Cuando se utiliza en un proyecto global, `ThisDrawing` siempre hace referencia al documento que está activo en AutoCAD. En los proyectos incrustados, sin embargo, `ThisDrawing` siempre hace referencia al documento que contiene el proyecto. Por ejemplo, la siguiente línea de código de un proyecto global guarda el dibujo que se encuentre activo en AutoCAD, sea cual sea:

```
ThisDrawing.Save
```

- **Referencias a objetos de la jerarquía de objetos**
- **Acceso al objeto Application**

Referencias a objetos de la jerarquía de objetos

Se puede hacer referencia a los objetos directamente o a través de una variable definida por el usuario. Para utilizar una referencia directa a un objeto, inclúyalo en la jerarquía de la llamada. Por ejemplo, la siguiente instrucción añade una línea en el espacio modelo. Observe que la jerarquía comienza con `ThisDrawing`, va al objeto ModelSpace y a continuación llama al método AddLine:

```
Dim startPoint(0 To 2) As Double, endPoint(0 To 2) As Double
Dim LineObj as AcadLine
startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
endPoint(0) = 30: endPoint(1) = 20: endPoint(2) = 0
Set LineObj = ThisDrawing.ModelSpace.AddLine(startPoint,endPoint)
```

Para hacer referencia a los objetos mediante una variable definida por el usuario, defina la variable con el tipo deseado y, a continuación, establézcala como el objeto adecuado. Por ejemplo, el código siguiente define una variable (`moSpace`) de tipo `AcadModelSpace` y la establece como igual al espacio modelo actual:

```
Dim moSpace As AcadModelSpace
Set moSpace = ThisDrawing.ModelSpace
```

La siguiente instrucción agrega una línea al espacio modelo mediante la variable definida por el usuario:

```
Dim startPoint(0 To 2) As Double, endPoint(0 To 2) As Double
Dim LineObj as AcadLine
startPoint(0) = 0: startPoint(1) = 0: startPoint(2) = 0
endPoint(0) = 30: endPoint(1) = 20: endPoint(2) = 0
Set LineObj = moSpace.AddLine(startPoint,endPoint)
```

Recuperación del primer objeto entidad del espacio modelo

El ejemplo siguiente devuelve el primer objeto entidad del espacio modelo. Este código tendría el mismo efecto en entidades del espacio papel. Tenga en cuenta que todos los objetos del dibujo se pueden definir como objetos AcadEntity:

```
Sub Ch2_FindFirstEntity()
    ' This example returns the first entity in model space
    On Error Resume Next
    Dim entity As AcadEntity
    If ThisDrawing.ModelSpace.count <> 0 Then
        Set entity = ThisDrawing.ModelSpace.Item(0)
        MsgBox entity.ObjectName + _
            " is the first entity in model space."
    Else
        MsgBox "There are no objects in model space."
    End If
End Sub
```

<$nopage>raíz (objeto):

Acceso al objeto Application

La propiedad Application del objeto Document proporciona acceso al objeto Application. El objeto Application está por encima del objeto Document en la jerarquía de objetos.

El objeto `ThisDrawing` proporciona acceso al objeto Document. Por ejemplo, la siguiente línea de código actualiza la aplicación:

```
ThisDrawing.Application.Update
```

Los objetos de colección

Un objeto de colección es un objeto predefinido que contiene (es un objeto propietario de) todas las instancias de un objeto similar. A continuación se muestra una lista de objetos de colección:

Colección

Contiene todos los documentos abiertos en la sesión actual de AutoCAD.

Colección de espacio modelo

Contiene todos los objetos gráficos (entidades) del espacio modelo.

Colección de espacio papel

Contiene todos los objetos gráficos (entidades) de la disposición activa del espacio papel.

Objeto de bloque

Contiene todas las entidades de una definición de bloque concreta.

Colección de bloques

Contiene todos los bloques del dibujo.

Colección de diccionarios

Contiene todos los diccionarios del dibujo.

Colección de estilos de cotas

Contiene todos los estilos de acotación del dibujo.

Colección de dependencias de archivos

Contiene todos los elementos en la lista de dependencias de archivos.

Colección de grupos

Contiene todos los grupos del dibujo.

Colección de hipervínculos

Contiene todos los hipervínculos de una entidad dada.

Colección de capas

Contiene todas las capas del dibujo.

Colección de presentaciones

Contiene todas las disposiciones del dibujo.

Colección de tipos de línea

Contiene todos los tipos de línea del dibujo.

Colección de barras de menús

Contiene todos los menús que actualmente se muestran en AutoCAD.

Colección de grupos de menús

Contiene todos los menús y barras de herramientas que actualmente están cargados en AutoCAD.

Colección de configuraciones de trazado

Contiene parámetros de trazado guardados en el dibujo.

Colección de aplicaciones registradas

Contiene todas las aplicaciones registradas del dibujo.

Colección de conjuntos de selección

Contiene todos los conjuntos de selección del dibujo.

Colección de estilos de texto

Contiene todos los estilos de texto del dibujo.

Colección de coordenadas SCP

Contiene todos los sistemas de coordenadas personales (SCP) del dibujo.

Colección de vistas

Contiene todas las vistas del dibujo.

Colección de ventanas gráficas

Contiene todas las ventanas gráficas del dibujo.

- **Acceso a una colección**
- **Adición de miembros a una colección**
- **Iteración en un objeto de colección**
- **Supresión de miembros de un objeto de colección**

Acceso a una colección

Puede acceder a la mayoría de los objetos de colecciones a través del objeto Document. Dicho objeto contiene una propiedad por cada uno de los objetos Collection. Por ejemplo, el código siguiente define una variable y la define como la colección Layers del dibujo actual:

```
Dim layerCollection as AcadLayers
Set layerCollection = ThisDrawing.Layers
```

Puede acceder a las colecciones Documents, MenuBar y MenuGroups a través del objeto Application. Dicho objeto contiene una propiedad por cada una de estas colecciones. Por ejemplo, el código siguiente define una variable y la define como la colección MenuGroups de la aplicación:

```
Dim MenuGroupsCollection as AcadMenuGroups
Set MenuGroupsCollection = ThisDrawing.Application.MenuGroups
```

Adición de miembros a una colección

Para agregar un miembro nuevo a la colección, utilice el método Add. Por ejemplo, el siguiente código crea una capa nueva y la añade a la colección de capas (Layers).

```
Dim newLayer as AcadLayer
Set newLayer = ThisDrawing.Layers.Add("MyNewLayer")
```

Iteración en un objeto de colección

Utilice el método Item para seleccionar un miembro concreto de un objeto de colección. Dicho método requiere un identificador como número de índice que especifique la ubicación del elemento en la colección o bien como una cadena que represente al nombre del elemento.

Item es el método predeterminado por defecto para las colecciones: cuando el usuario no indica un nombre de método al mencionar una colección, se utiliza el método Item. Las siguientes instrucciones son equivalentes:

```
ThisDrawing.Layers.Item("ABC")
ThisDrawing.Layers("ABC")
```

Nota No utilice los métodos de edición de la entidad (Copy, Array, Mirror, etc.) en un objeto a la vez que itera en una colección empleando el mecanismo For Each. Termine la iteración antes de intentar editar un objeto de la colección o bien cree una matriz temporal y defínala como igual a la colección. A continuación puede iterar en la matriz copiada y efectuar cambios.

Iterar en la colección de capas

El ejemplo siguiente efectúa iteraciones en una colección y presenta los nombres de todas las capas de la colección:

```
Sub Ch2_IterateLayer()
    ' Iterate through the collection
    On Error Resume Next
    Dim I As Integer
    Dim msg As String
    msg = ""
    For I = 0 To ThisDrawing.Layers.count - 1
        msg = msg + ThisDrawing.Layers.Item(I).Name + vbCrLf
    Next
```

```
    MsgBox msg
End Sub
```

Búsqueda de la capa "MiCapa"

El siguiente ejemplo se refiere a una capa denominada "MiCapa" y presenta un mensaje si la capa no existe:

```
Sub Ch2_FindLayer()
    ' Use the Item method to find a layer named "MiCapa"
    On Error Resume Next
    Dim ABCLayer As AcadLayer
    Set ABCLayer = ThisDrawing.Layers("MiCapa")
    If Err <> 0 Then
        MsgBox "The layer 'MiCapa' does not exist."
    End If
End Sub
```

Supresión de miembros de un objeto de colección

Para suprimir un estilo de cota concreto, utilice el método Delete que se encuentra en el objeto miembro. Por ejemplo, la siguiente línea de código suprime la capa ABC:

```
Dim ABCLayer as AcadLayer
Set ABCLayer = ThisDrawing.Layers.Item("ABC")
ABCLayer.Delete
```

Una vez que se suprime un objeto, no vuelva a intentar acceder a él a lo largo del programa.

Conceptos básicos sobre propiedades y métodos

Cada objeto lleva asociados sus propios métodos y propiedades. Las propiedades describen aspectos individuales del objeto; los métodos son acciones que pueden realizarse con el objeto concreto. Una vez creado el objeto, lo puede consultar y modificar a través de sus propiedades y métodos.

Por ejemplo, un objeto Circle tiene la propiedad Center. Esta propiedad representa las coordenadas 3D del SCU en el centro del círculo. Para cambiar el centro del círculo, solo tiene que definir esta propiedad con unas nuevas coordenadas. El objeto Circle también tiene un método denominado Offset. Este método crea un objeto nuevo a una distancia de desfase especificada con respecto al círculo original. Para ver una lista de todos los métodos y propiedades del objeto Circle, véase el objeto Circle en *ActiveX and VBA Reference* de AutoCAD.

Definición de objetos anteriores (padre)

Todos los objetos tienen un objeto anterior al que están vinculados permanentemente. Todos los objetos tienen como origen un mismo objeto anterior llamado objeto raíz. Puede acceder a todos los objetos de la interfaz siguiendo los vínculos que unen al objeto raíz con los objetos subordinados. Asimismo, los objetos tienen una propiedad llamada Application que los une directamente con el objeto raíz.

El objeto raíz de la interfaz de AutoCAD es la aplicación AutoCAD.

Localización de la biblioteca de tipos

Los objetos, métodos y propiedades expuestos por objetos de automatización están incluidos en una biblioteca de tipos. Una biblioteca de tipos es un archivo o parte de un archivo que describe el tipo de uno o más objetos.

Las bibliotecas de tipos no almacenan objetos, sino información. El acceso a una biblioteca de tipos permite a aplicaciones y exploradores determinar características de un objeto tales como las interfaces que admite o los nombres y direcciones de los miembros de cada interfaz.

Para poder utilizar el objeto de automatización expuesto por una aplicación, debe hacer referencia a su biblioteca de tipos. La referencia se define automáticamente en el IDE de VBA. En otros entornos de desarrollo interactivo, el usuario debe crear una referencia al archivo de biblioteca de tipos de AutoCAD, *acax17enu.tlb*, que se encuentra en *c:\Archivos de programa\Archivos comunes\Autodesk Shared*. Para acceder a objetos del Administrador de conjuntos de planos en el IDE de VBA o en otros entornos, el usuario debe crear una referencia al archivo de biblioteca de tipos de AcSmComponents17 1.0, *AcSmComponents17.tlb*, que se encuentra en *c:\Archivos de programa\Archivos comunes\Autodesk Shared*.

Puede utilizar los objetos de una aplicación sin hacer referencia a la biblioteca de tipos de la misma. No obstante, es preferible añadir la referencia a la biblioteca de tipos, por los siguientes motivos:

- El acceso a las funciones disponibles globalmente es directo, sin necesidad de calificación.
- Durante la compilación puede comprobarse si la activación de funciones, propiedades y métodos es correcta y, por lo tanto, la ejecución es más rápida en tiempo de ejecución.
- Pueden declararse variables de los tipos definidos en la biblioteca, y

aumentar la fiabilidad y legibilidad en tiempo de ejecución.

¿Comentarios?

Uso de variantes en métodos y propiedades

ActiveX Automation utiliza variantes para transferir matrices de datos. Aunque esto puede ser confuso para un usuario sin experiencia, es muy fácil una vez aprendidos los conceptos básicos. ActiveX Automation de AutoCAD también proporciona utilidades para facilitar la conversión de los tipos de datos.

- **¿Qué es un Variant?**
- **Uso de Variant (variantes) en matrices de datos**
- **Conversión de matrices en variantes**
- **Interpretación de matrices de variantes**

¿Qué es un Variant?

Un Variant (variante) es un tipo de datos especial que puede contener cualquier clase de datos excepto cadenas de longitud fija y tipos definidos por el usuario. Un variante también puede contener los valores especiales `Empty`, `Error`, `Nothing` y `NULL`. Es posible determinar el tratamiento que se aplica a los datos de un variante utilizando las funciones `VarType` o `TypeName` de VBA.

El tipo de datos Variant se puede utilizar en sustitución de la mayoría de tipos de datos, lo que aporta gran flexibilidad.

Uso de Variant (variantes) en matrices de datos

Las variantes se utilizan para la entrada y salida de matrices de datos de ActiveX Automation de AutoCAD. La matriz, por tanto, debe ser una variante aceptada por los métodos y propiedades de ActiveX Automation de AutoCAD. Asimismo, la salida de los datos matriciales de ActiveX Automation de AutoCAD debe tratarse como un Variant.

Nota En AutoCAD, las matrices de entrada de VBA se convierten en variantes de forma automática. Es decir, que no es necesario proporcionar una matriz de variantes como entrada a los métodos y propiedades de ActiveX Automation si se están utilizando desde VBA. No obstante, todas las matrices de salida estarán en forma de variantes, por lo que no debe olvidar tratarlas como corresponde.

Conversión de matrices en variantes

AutoCAD ActiveX Automation proporciona un método para convertir una matriz de datos en un variante. Este método se llama CreateTypedArray y crea un variante que contiene una matriz de enteros, números flotantes, dobles, etc. Puede transferir el variante resultante a cualquier método o propiedad de AutoCAD que acepte como variante una matriz de números.

El método CreateTypedArray utiliza como entradas el tipo de valores que están en la matriz, y la matriz de datos que debe convertirse. Devuelve como variante la matriz de valores.

Creación de una curva spline con el método CreateTypedArray

El siguiente código convierte tres matrices mediante CreateTypedArray: coordenadas de los puntos de ajuste de la spline, y la tangente inicial y final de la spline. A continuación transfiere la variante al método AddSpline para crear la spline.

```
Sub Ch2_CreateSplineUsingTypedArray()
    ' This example creates a spline object in model space
    ' using the CreateTypedArray method.
    Dim splineObj As AcadSpline
    Dim startTan As Variant
    Dim endTan As Variant
    Dim fitPoints As Variant
    Dim utilObj As Object ' late bind the Utility object
    Set utilObj = ThisDrawing.Utility
    ' Define the Spline Object
    utilObj.CreateTypedArray _
                startTan, vbDouble, 0.5, 0.5, 0
    utilObj.CreateTypedArray _
                endTan, vbDouble, 0.5, 0.5, 0
    utilObj.CreateTypedArray _
                fitPoints, vbDouble, 0, 0, 0, 5, 5, 0, 10, 0, 0
```

```
    Set splineObj = ThisDrawing.ModelSpace.AddSpline _
                (fitPoints, startTan, endTan)
    ' Zoom in on the newly created spline
    ZoomAll
End Sub
```

¿Comentarios?

Interpretación de matrices de variantes

La información matricial se transfiere de vuelta desde ActiveX Automation de AutoCAD como variante. Si conoce el tipo de datos de la matriz, puede acceder a la variante como a una matriz. Si no conoce el tipo de datos que contiene la variante, utilice las funciones `VarType` o `Typename de VBA`. Estas funciones devuelven el tipo de datos del variante. Si necesita iterar en la matriz, puede utilizar la declaración de VBA `For Each`.

Cálculo de la distancia entre dos puntos

El código siguiente muestra el método de cálculo de la distancia entre dos puntos especificados por el usuario. En este ejemplo, el tipo de datos se conoce porque todas las coordenadas son dobles. Las coordenadas 3D consisten en una matriz de tres elementos de números dobles, y las coordenadas 2D en una matriz de dos elementos de números dobles.

```
Sub Ch2_CalculateDistance()
    Dim point1 As Variant
    Dim point2 As Variant
    ' Get the points from the user
    point1 = ThisDrawing.Utility.GetPoint _
                (, vbCrLf & "First point: ")
    point2 = ThisDrawing.Utility.GetPoint _
                (point1, vbCrLf & "Second point: ")
    ' Calculate the distance between point1 and point2
    Dim x As Double, y As Double, z As Double
    Dim dist As Double
    x = point1(0) - point2(0)
    y = point1(1) - point2(1)
    z = point1(2) - point2(2)
    dist = Sqr((Sqr((x ^ 2) + (y ^ 2)) ^ 2) + (z ^ 2))
    'Display the resulting distance
    MsgBox "The distance between the points is: " _
                  & dist, , "Calculate Distance"
```

```
End Sub
```

¿Comentarios?

Otros lenguajes de programación

Este manual se ha redactado para el lenguaje de programación VBA. Los ejemplos de programación y las aplicaciones de ejemplo están escritas en VBA. Para poder utilizar este código en otros entornos de programación debe actualizarse para dicho entorno.

Para obtener más información acerca de la conversión del código de los ejemplos, véase la documentación del entorno de desarrollo.

Nota La clave de registro para el acceso a la aplicación COM de AutoCAD 2008 es AutoCAD.Application.17.

- **Conversión del código VBA a VB**

Conversión del código VBA a VB

Para actualizar un ejemplo de código y utilizarlo con VB, primero hay que hacer referencia a la biblioteca de tipos de AutoCAD. Si desea hacerlo en VB, elija la opción Referencias en el menú Proyecto para abrir el cuadro de diálogo correspondiente. En el cuadro de diálogo Referencias, elija la biblioteca de tipos de AutoCAD y haga clic en Aceptar.

A continuación, en el ejemplo de código, reemplace todas las referencias a `ThisDrawing` con una variable definida por el usuario que haga referencia al documento activo. Con este propósito, defina una variable para la aplicación de AutoCAD(`acadApp`) y para el documento actual (`acadDoc`). A continuación, establezca como variable de aplicación la aplicación actual de AutoCAD.

Si se está ejecutando AutoCAD, la función GetObject de VB recupera el objeto Application de AutoCAD cuando usted especifica el número de versión de AutoCAD. Si no se está ejecutando AutoCAD, ocurre un error que (en el ejemplo) se detecta y después se despeja. A continuación, la función CreateObject intenta crear un objeto Application de AutoCAD. Si lo consigue, se inicia AutoCAD; en caso contrario, se muestra un cuadro de mensaje, con una descripción del error.

Durante la ejecución de varias sesiones simultáneas de AutoCAD, la función GetObject devuelve la primera instancia de AutoCADen la tabla de objetos en ejecución en Windows. Para obtener más información acerca de la verificación de la sesión devuelta por GetObject, consulte la documentación de Microsoft VBA relativa a la tabla de objetos en ejecución (ROT) y a la función GetObject.

Para que aparezca la ventana de dibujo de AutoCAD debe establecer en `TRUE` la propiedad Visible de la aplicación AutoCAD.

Si GetObject crea una sesión nueva de AutoCAD (es decir, AutoCAD aún no estaba ejecutándose cuando se emitió GetObject), al no poder asignar `TRUE` a

Visible la aplicación AutoCAD será invisible y ni siquiera aparecerá en la barra de tareas de Windows.

Nota Utilice ProgIDs dependientes de la versión. Si una función CreateObject o GetObject utiliza un ProgID dependiente de la versión, cambie la función para que utilice un ProgID dependiente de la versión. Por ejemplo, si utiliza CreateObject, sustituya CreateObject ("AutoCAD.Application") por CreateObject ("AutoCAD.Application.17"). Además, si un método GetInterfaceObject usa un ProgID independiente de la versión, el método debe cambiarse para que utilice un ProgID dependiente de la versión.

Conexión con AutoCAD desde Visual Basic 6

El siguiente ejemplo de código utiliza las propiedades Clear y Description de Err. Si el entorno de programación no admite estas propiedades, deberá modificar el ejemplo en consecuencia:

```
Sub Ch2_ConnectToAcad()
    Dim acadApp As AcadApplication
    On Error Resume Next
    Set acadApp = GetObject(, "AutoCAD.Application.17")
    If Err Then
        Err.Clear
        Set acadApp = CreateObject("AutoCAD.Application.17")
        If Err Then
            MsgBox Err.Description
            Exit Sub
        End If
    End If
    MsgBox "Now running " + acadApp.Name + _
           " version " + acadApp.Version
End Sub
```

A continuación, defina la variable del documento como el objeto Document de la aplicación de AutoCAD. Dicho objeto es el que devuelve la propiedad ActiveDocument del objeto Application.

```
Dim acadDoc as AcadDocument
Set acadDoc = acadApp.ActiveDocument
```

En adelante, utilice la variable `acadDoc` para hacer referencia al dibujo actual de AutoCAD.

Ejemplo de código comparado entre VBA y VB

El siguiente código del ejemplo muestra cómo se crea una línea en VBA y en VB:

Creación de una línea con VBA:

```
Sub Ch2_AddLineVBA()
    ' This example adds a line
    ' in model space
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    ' Define the start and end
    ' points for the line
    startPoint(0) = 1
    startPoint(1) = 1
    startPoint(2) = 0
    endPoint(0) = 5
    endPoint(1) = 5
    endPoint(2) = 0
    ' Create the line in model space
    Set lineObj = ThisDrawing. _
        ModelSpace.AddLine _
        (startPoint, endPoint)
    ' Zoom in on the newly created line
    ZoomAll
End Sub
```

Creación de una línea con VB:

```
Sub Ch2_AddLineVB()
    On Error Resume Next
    ' Connect to the AutoCAD application
    Dim acadApp As AcadApplication
    Set acadApp = GetObject _
                  (, "AutoCAD.Application.17")
    If Err Then
        Err.Clear
        Set acadApp = CreateObject _
                  ("AutoCAD.Application.17")
        If Err Then
            MsgBox Err.Description
            Exit Sub
        End If
    End If
    ' Connect to the AutoCAD drawing
    Dim acadDoc As AcadDocument
```

```
    Set acadDoc = acadApp.ActiveDocument
    ' Establish the endpoints of the line
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    startPoint(0) = 1
    startPoint(1) = 1
    startPoint(2) = 0
    endPoint(0) = 5
    endPoint(1) = 5
    endPoint(2) = 0
    ' Create a Line object in model space
    Set lineObj = acadDoc.ModelSpace.AddLine _
                          (startPoint, endPoint)
    ZoomAll
    acadApp.visible = True
End Sub
```

¿Comentarios?

<$nopage>métodos.

Control del entorno de AutoCAD

En este capítulo se describen los conceptos básicos necesarios para desarrollar aplicaciones en AutoCAD. También se explica cómo controlar el entorno de AutoCAD y cómo trabajar eficazmente en dicho entorno.

Apertura, guardado y cierre de dibujos

La colección Documents y el objeto Document proporcionan acceso a las funciones de archivos de AutoCAD®.

Para crear un nuevo dibujo o abrir uno ya existente, utilice los métodos de la colección Documents. El método Add crea un dibujo nuevo y lo añade a la colección Documents. El método Open abre un dibujo existente. La colección de documentos también cuenta con un método Close que cierra todos los dibujos abiertos en la sesión de AutoCAD.

Para guardar los dibujos, utilice cualquiera de los métodos Save o SaveAs Ocasionalmente deseará comprobar si el dibujo activo tiene cambios sin guardar. Es conveniente hacer esto antes de salir de la sesión de AutoCAD o comenzar un nuevo dibujo. Utilice la propiedad Saved para asegurarse de que el dibujo actual no contiene cambios sin guardar.

Para importar y exportar dibujos, utilice los métodos Import y Export del objeto Document.

Apertura de un dibujo existente

En este ejemplo se utiliza el método Open para abrir un dibujo existente. Se usa la función `Dir` de VBA para comprobar si existe el archivo antes de intentar abrirlo. Cambie el nombre del archivo de dibujo o su ruta de acceso para especificar un archivo de dibujo de AutoCAD existente en el sistema.

```
Sub Ch3_OpenDrawing()
    Dim dwgName As String
    dwgName = "c:\campus.dwg"
    If Dir(dwgName) <> "" Then
        ThisDrawing.Application.Documents.Open dwgName
    Else
        MsgBox "File " & dwgName & " does not exist."
```

```
    End If
End Sub
```

Creación de un dibujo nuevo

En este ejemplo se utiliza el método Add para crear un dibujo basado en la plantilla por defecto.

```
Sub Ch3_NewDrawing()
    Dim docObj As AcadDocument
    Set docObj = ThisDrawing.Application.Documents.Add
End Sub
```

Guardado del dibujo activo

En este ejemplo se guarda el dibujo activo con el nombre actual y una segunda vez con otro nombre.

```
Sub Ch3_SaveActiveDrawing()
    ' Save the active drawing under the current name
    ThisDrawing.Save
    ' Save the active drawing under a new name
    ThisDrawing.SaveAs "MyDrawing.dwg"
End Sub
```

Comprobación de posibles cambios sin guardar en un dibujo

En este ejemplo se comprueba si el dibujo contiene cambios pendientes de guardar y se pregunta al usuario si desea guardarlos (si la respuesta es negativa, omita estos pasos hasta el final). Si la respuesta es afirmativa, utilice el método Save para guardar el dibujo actual, como se muestra a continuación:

```
Sub Ch3_TestIfSaved()
    If Not (ThisDrawing.Saved) Then
        If MsgBox("Do you wish to save this drawing?", _
                  vbYesNo) = vbYes Then
            ThisDrawing.Save
        End If
    End If
End Sub
```

Definición de preferencias de AutoCAD

Hay nueve objetos relacionados con las opciones, cada uno de ellos representado en una ficha del cuadro de diálogo Opciones. Estos objetos proporcionan acceso a todas las opciones almacenadas en el registro mediante el cuadro de diálogo Opciones. Muchos de los parámetros de AutoCAD pueden adaptarse a necesidades personales a través de las propiedades de estos objetos, que son los siguientes:

- PreferencesDisplay
- PreferencesDrafting
- PreferencesFiles
- PreferencesOpenSave
- PreferencesOutput
- PreferencesProfiles
- PreferencesSelection
- PreferencesSystem
- PreferencesUser

Puede acceder a estos objetos a través del objeto Preferences Para acceder al objeto Preferences, utilice la propiedad Preference del objeto Application:

```
Dim acadPref as AcadPreferences
Set acadPref = ThisDrawing.Application.Preferences
```

Puede acceder a cualquiera de los objetos Preferences mediante las propiedades Display, Drafting, Files, OpenSave, Output, Profile, Selection, System y User.

Configuración del cursor en cruz para pantalla completa

```
Sub Ch2_PrefsSetCursor()
    ' This example sets the crosshairs of the AutoCAD drawing cursor
    ' to full screen.
    ' Access the Preferences object
    Dim acadPref As AcadPreferences
    Set acadPref = ThisDrawing.Application.Preferences
    ' Use the CursorSize property to set the size of the crosshairs
    acadPref.Display.CursorSize = 100
End Sub
```

Visualización del menú de pantalla y las barras de desplazamiento

```
Sub Ch2_PrefsSetCursor()
    ' This example enables the screen menu and disables the scroll
    ' bars with the DisplayScreenMenu and DisplayScrollBars
    ' properties.
    ' Access the Preferences object
    Dim acadPref As AcadPreferences
    Set acadPref = ThisDrawing.Application.Preferences
    ' Display the screen menu and disable scroll bars
    acadPref.Display.DisplayScreenMenu = True
    acadPref.Display.DisplayScrollBars = False
End Sub
```

- **Preferencias de bases de datos**

Preferencias de bases de datos

Además de los nueve objetos de preferencias, existe un objeto llamado DatabasePreferences que contiene todas las opciones guardadas en el dibujo. Este otro objeto se ha ideado para poner las opciones almacenadas en los dibujos a disposición de las aplicaciones que accedan a los dibujos de AutoCAD sin iniciar antes el programa (aplicaciones ObjectDBX).

El objeto DatabasePreferences se encuentra bajo la jerarquía del objeto Document.

Control de la ventana de la aplicación

La capacidad para controlar la ventana de la aplicación ofrece a los programadores la flexibilidad que necesitan para crear aplicaciones eficaces e inteligentes. A veces, por ejemplo mientras se está procesando código en otra aplicación como Excel, conviene minimizar la ventana de AutoCAD. Además, podría ser necesario verificar el estado de la ventana de AutoCAD antes de realizar tareas como solicitar información al usuario.

Los métodos y las propiedades del objeto de aplicación permiten cambiar la posición, el tamaño y la visibilidad de la ventana de la aplicación. También puede utilizar la propiedad WindowState para minimizar, maximizar y comprobar el estado actual de la ventana de la aplicación.

Colocación y ajuste de tamaño de la ventana de la aplicación

En este ejemplo se utilizan las propiedades WindowTop, WindowLeft, Width y Height para situar la ventana de la aplicación AutoCAD en la esquina superior izquierda de la pantalla y fijar su tamaño en 400 píxeles de ancho por 400 de alto.

```
Sub Ch3_PositionApplicationWindow()
    ThisDrawing.Application.WindowTop = 0
    ThisDrawing.Application.WindowLeft = 0
    ThisDrawing.Application.width = 400
    ThisDrawing.Application.height = 400
End Sub
```

Para maximizar la ventana de la aplicación

```
Sub Ch3_MaximizeApplicationWindow()
    ThisDrawing.Application.WindowState = acMax
End Sub
```

Para minimizar la ventana de la aplicación

```
Sub Ch3_PositionApplicationWindow()
    ThisDrawing.Application.WindowState = acMin
End Sub
```

Para conocer el estado actual de la ventana de AutoCAD

Este ejemplo consulta el estado de la ventana de aplicación y lo presenta en un cuadro de mensaje.

```
Sub Ch3_CurrentWindowState()
    Dim CurrWindowState As Integer
    Dim msg As String
    CurrWindowState = ThisDrawing.Application.WindowState
    msg = Choose(CurrWindowState, "normal", _
                 "minimized", "maximized")
    MsgBox "The application window is " + msg
End Sub
```

Para ocultar la ventana de la aplicación

El código siguiente utiliza la propiedad Visible para ocultar la aplicación AutoCAD al usuario final.

```
Sub Ch3_HideWindowState()
    ThisDrawing.Application.Visible = False
End Sub
```

Control de las ventanas del dibujo

Como en la ventana de la aplicación de AutoCAD, las ventanas de documento también se pueden minimizar, maximizar, cambiar de posición y comprobar su estado. Pero también es posible cambiar la presentación de un dibujo en una ventana mediante vistas, ventanas gráficas y ajustes de zoom.

ActiveX de AutoCAD ofrece varios modos para mostrar distintas vistas del dibujo. Es posible controlar la presentación en pantalla del dibujo y desplazarse rápidamente a sus diferentes áreas al tiempo que se supervisa el efecto general de los cambios. Puede utilizar el zoom para cambiar la ampliación o el encuadre y volver a colocar la vista en el área gráfica, guardar una vista y recuperarla cuando necesite trazarla o consultar detalles concretos, o bien, visualizar varias vistas a la vez dividiendo la pantalla en ventanas en mosaico.

- **Colocación y ajuste de tamaño de la ventana del documento**
- **Utilización de zoom**
- **Vistas guardadas**
- **Ventanas en mosaico**
- **Actualización de la geometría en la ventana de documento**

Colocación y ajuste de tamaño de la ventana del documento

Utilice el objeto Document para modificar la posición y el tamaño de cualquier ventana de documento. La ventana de documento se puede minimizar o maximizar a través de la propiedad WindowState, que también permite conocer su estado actual..

Colocación de una ventana de documento

En este ejemplo se utilizan las propiedades Width y Height para definir la ventana del documento activo con una anchura y altura de 400 x 400 píxeles.

```
Sub Ch3_SizeDocumentWindow()
ThisDrawing.Width = 400
ThisDrawing.Height = 400
End Sub
```

Para maximizar la ventana del documento activo

```
Sub Ch3_MaximizeDocumentWindow()
ThisDrawing.WindowState = acMax
End Sub
```

Para minimizar la ventana del documento activo

```
Sub Ch3_MinimizeDocumentWindow()
ThisDrawing.WindowState = acMin
End Sub
```

Para conocer el estado actual de la ventana del documento activo

```
Sub Ch3_CurrentWindowState()
    Dim CurrWindowState As Integer
    Dim msg As String
    CurrWindowState = ThisDrawing.WindowState
    msg = Choose(CurrWindowState, "normal", _
                 "minimized", "maximized")
    MsgBox "The document window is " + msg
End Sub
```

¿Comentarios?

Utilización de zoom

Una vista es una ampliación, posición y orientación específica de un dibujo. AutoCAD pone a su disposición numerosas opciones de Zoom que aumentan o reducen el tamaño de la imagen que aparece en el área gráfica. Para obtener más información acerca de ampliar el zoom en AutoCAD, véase “Ampliación de una vista (hacer zoom)” en el *Manual del usuario*.

- **Definición de una ventana de zoom**
- **Atribución de escala a una vista**
- **Centrado de objetos**
- **Visualización de límites y extensiones del dibujo**

Definición de una ventana de zoom

Se puede ampliar con suma rapidez un área especificando sus esquinas. Para ampliar un área especificando su contorno, utilice uno de los métodos ZoomWindow o ZoomPickWindow El primero permite definir mediante programación dos puntos que representan la ventana zoom. El método ZoomPickWindow requiere que el usuario designe dos puntos. Estos dos puntos de designación conforman la ventana de zoom.

Ampliación del dibujo activo al tamaño de una ventana definida por dos puntos

```
Sub Ch3_ZoomWindow()
    ' ZoomWindow
    MsgBox "Perform a ZoomWindow with:" & vbCrLf & _
           "1.3, 7.8, 0" & vbCrLf & _
           "13.7, -2.6, 0", , "ZoomWindow"
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    point1(0) = 1.3: point1(1) = 7.8: point1(2) = 0
    point2(0) = 13.7: point2(1) = -2.6: point2(2) = 0
    ThisDrawing.Application.ZoomWindow point1, point2
    ' ZoomPickWindow
    MsgBox "Perform a ZoomPickWindow", , "ZoomPickWindow"
    ThisDrawing.Application.ZoomPickWindow
End Sub
```

Atribución de escala a una vista

Cuando desee aumentar o reducir la ampliación en pantalla de una imagen conforme a una escala determinada, puede indicar la escala de tres formas distintas:

- En relación con los límites del dibujo
- En relación con la vista actual
- En relación con las unidades del espacio papel

Para ampliar o reducir una vista, utilice el método ZoomScaled Este método espera dos parámetros de entrada: la escala y el tipo de escala. La escala es simplemente un número. La interpretación de este número por parte de AutoCAD depende del tipo de escala elegida.

El tipo de la escala determina si su valor se crea con respecto a los límites del dibujo, a la vista actual o a las unidades del espacio papel. Para asignar una escala relativa a los límites del dibujo, use la constante `acZoomScaledAbsolute`. Para ajustar la escala de una vista en relación con la vista actual, use la constante `acZoomScaledAbsolute`. Para asignar una escala relativa a las unidades del espacio papel, use la constante `acZoomScaledRelativePSpace`.

Ampliación del dibujo activo a una escala específica

```
Sub Ch3_ZoomScaled()
    MsgBox "Perform a ZoomScaled using:" & vbCrLf & _
           "Scale Type: acZoomScaledRelative" & vbCrLf & _
           "Scale Factor: 2", , "ZoomScaled"
    Dim scalefactor As Double
    Dim scaletype As Integer
    scalefactor = 2
```

```
    scaletype = acZoomScaledRelative
    ThisDrawing.Application.ZoomScaled scalefactor, scaletype
End Sub
```

¿Comentarios?

Centrado de objetos

Ampliación del dibujo activo a un centro especifico Puede mover un punto específico de su dibujo para centrarlo en el área gráfica. El método ZoomCenter resulta especialmente útil a la hora de reajustar el tamaño de un objeto y situarlo en el centro de la ventana gráfica. ZoomCenter permite especificar la escala mediante una ampliación relativa a la vista actual

Ampliación del dibujo activo a un centro específico

El siguiente ejemplo ilustra el efecto del método ZoomCenter cuando se aplica para mostrar una vista en su tamaño real o al doble de su tamaño:

```
Sub Ch3_ZoomCenter()
    MsgBox "Perform a ZoomCenter using:" & vbCrLf & _
           "Center 3, 3, 0" & vbCrLf & _
           "Magnification: 10", , "ZoomCenter"
    Dim Center(0 To 2) As Double
    Dim magnification As Double
    Center(0) = 3: Center(1) = 3: Center(2) = 0
    magnification = 10
    ThisDrawing.Application.ZoomCenter Center, magnification
End Sub
```

Visualización de límites y extensiones del dibujo

Los métodos ZoomAll, ZoomExtents o ZoomPrevious permiten la presentación en pantalla de una vista en función de los contornos o de la extensión de los objetos del dibujo.

ZoomAll muestra el dibujo completo. Si los objetos se extienden más allá de los límites, ZoomAll muestra la extensión de los objetos. Si los objetos están dibujados dentro de los límites, ZoomAll muestra los límites.

ZoomExtens calcula el factor de ampliación tomando como referencia la extensión de la ventana gráfica actual, no de la vista actual. Por lo general, la ventana gráfica activa está completamente visible, por lo que el resultado es obvio y previsible. Por lo general, la ventana gráfica activa está completamente visible, por lo que el resultado es obvio y previsible. Sin embargo, cuando se utilizan los métodos Zoom en el espacio modelo mientras se trabaja en una ventana gráfica en el espacio papel, si se amplía sobrepasando los contornos de la ventana gráfica en el espacio papel, puede que no se vea parte del área ampliada.

ZoomExtents cambia la vista para ajustar las extensiones de entidad en el dibujo actual. En algunos casos (tanto para ZoomAll como para ZoomExtents), esto puede provocar la regeneración. La regeneración no tendrá lugar en capas inutilizadas o desactivadas. Si el dibujo no contiene objetos, ZoomExtents muestra los límites del dibujo.

En las vistas 3D, ZoomAll y ZoomExtents tienen el mismo resultado. Las líneas auxiliares infinitas (líneasx) y los rayos no influyen en ninguna de las opciones.

ZoomPrevious amplía la ventana gráfica hasta la extensión anterior.

Para obtener más información acerca del funcionamiento del zoom, véase “Ampliación de una vista (hacer zoom)” en el *Manual del usuario*.

Ampliación del dibujo activo a todo el contenido y a la extensión del dibujo

```
Sub Ch3_ZoomAll()
' ZoomAll
    MsgBox "Perform a ZoomAll", , "ZoomAll"
    ThisDrawing.Application.ZoomAll
' ZoomExtents
    MsgBox "Perform a ZoomExtents", , "ZoomExtents"
    ThisDrawing.Application.ZoomExtents
End Sub
```

¿Comentarios?

Vistas guardadas

Es posible asignar un nombre y guardar cada vista que desee volver a utilizar. Cuando ya no la necesite, puede suprimirla.

Si desea crear una vista nueva, utilice el método Add para añadir una nueva vista a la colección Views Al guardar el dibujo, se guarda también la posición y la escala de la vista.

El nombre de la vista se le asigna al crearla. Puede constar de un total de 255 caracteres y contener letras, dígitos y los caracteres especiales dólar ($), guión (-) y subrayado (_).

Para suprimir una vista guardada, utilice el método Delete. El método Delete del objeto View se encuentra en dicho objeto, no en el superior.

Adición de un objeto de vista

En el ejemplo siguiente se añade un objeto de vista (`viewObj`).

```
Sub Ch3_AddView()
    ' Add a named view to the views collection
    Dim viewObj As AcadView
    Set viewObj = ThisDrawing.Views.Add("View1")
End Sub
```

Supresión de un objeto de vista

En el ejemplo siguiente se borra un objeto de vista (objeto `viewObj`).

```
Sub Ch3_DeleteView()
    Dim viewObj As AcadView
    Set viewObj = ThisDrawing.Views("View1")
   ' Delete the view
```

```
    viewObj.Delete
End Sub
```

Eliminación de una vista de la colección de vistas

En este ejemplo se borra una vista guardada de la colección Views.

```
Sub Ch3_DeleteViewFromCollection()
   ThisDrawing.Views("View1").Delete
End Sub
```

Ventanas en mosaico

AutoCAD suele iniciar cada nuevo dibujo en una sola ventana gráfica que ocupa toda el área gráfica. Si lo desea, puede dividir el área de dibujo para mostrar varias ventanas de forma simultánea. Por ejemplo, si mantiene visibles tanto las vistas completas como las detalladas, resultará más fácil apreciar los efectos de los cambios sutiles introducidos en todo el dibujo. Las ventanas en mosaico ofrecen las siguientes posibilidades:

- ampliar, reducir, activar los modos Forzcursor, Rejilla y el modo de icono SCP, así como restituir vistas guardada en ventanas gráficas individuales
- dibujar en una ventana gráfica y desplazarse a otra durante la ejecución de un comando
- asignar nombre a una disposición de ventanas determinada para poderla utilizar en el futuro.

Es posible mostrar ventanas en mosaico en varias disposiciones. La forma de mostrar las ventanas depende en gran medida del número y del tamaño de las vistas que desee utilizar.

Para obtener más información e ilustraciones de las ventanas gráficas, véase “Definición de las ventanas gráficas del espacio modelo” en el *Manual del usuario*.

- **División de la ventana gráfica activa**
- **Activación de una ventana en mosaico**

División de la ventana gráfica activa

La ventana gráfica activa se divide con el método Split Este método utiliza un parámetro, el tipo de configuración, para establecer cómo se desea dividir la ventana gráfica. Para especificar la configuración de la ventana, utilice una de las siguientes constantes que corresponden a las configuraciones por defecto anteriormente mostradas: `acViewport2Horizontal`, `acViewport2Vertical`, `acViewport3Left`, `acViewport3Right`, `acViewport3Horizontal`, `acViewport3Vertical`, `acViewport3Above`, `acViewport3Below`, o `acViewport4`.

Para obtener más información acerca de la modificación de la configuración de las ventanas gráficas, véase “Definición de las ventanas gráficas del espacio modelo” en el *Manual del usuario*.

División de una ventana gráfica en dos ventanas horizontales

En el ejemplo siguiente se crea una ventana gráfica nueva que, a continuación, se divide en dos ventanas horizontales.

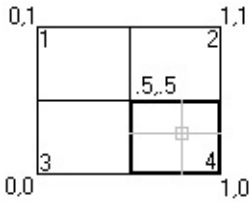
```
Sub Ch3_SplitAViewport()
    ' Create a new viewport
    Dim vportObj As AcadViewport
    Set vportObj = ThisDrawing.Viewports.Add("TEST_VIEWPORT")
    ' Split vportObj into 2 horizontal windows
    vportObj.Split acViewport2Horizontal
    ' Now set vportObj to be the active viewport
    ThisDrawing.ActiveViewport = vportObj
End Sub
```

Activación de una ventana en mosaico

En la ventana gráfica activa se introducen puntos y se seleccionan objetos. Para convertir una ventana gráfica en la ventana actual, utilice la propiedad ActiveViewport

Puede repetir la operación en las ventanas gráficas existentes hasta encontrar la que busca. Para ello, primero debe identificar el nombre de la disposición de ventanas a la que pertenece la ventana que desea utilizar, mediante la propiedad Name. Por otra parte, si se trata de una disposición de ventanas dividida, las ventanas individuales pueden identificarse mediante las propiedades LowerLeftCorner y UpperRightCorner

Las propiedades LowerLeftCorner y UpperRightCorner representan la ubicación gráfica de la ventana en la pantalla. Estas propiedades se definen según se indica a continuación (con una división de cuatro ventanas a modo de ejemplo):



Equivalencias del ejemplo:

- Ventana 1-LowerLeftCorner = (0, .5), UpperRightCorner = (.5, 1)
- Ventana 2-LowerLeftCorner = (.5, .5), UpperRightCorner = (1, 1)
- Ventana 3-LowerLeftCorner = (.5, 0), UpperRightCorner = (1, .5)
- Ventana 4-LowerLeftCorner = (.5, 0), UpperRightCorner = (1, .5)

División de una ventana gráfica e iteración en todas las ventanas

En este ejemplo se divide una ventana gráfica en cuatro ventanas. A continuación se efectúan iteraciones en todas las ventanas gráficas del dibujo y se presenta el nombre de la ventana y las esquinas inferior izquierda y superior derecha de cada una.

```
Sub Ch3_IteratingViewportWindows()
    ' Create a new viewport and make it active
    Dim vportObj As AcadViewport
    Set vportObj = ThisDrawing.Viewports.Add("TEST_VIEWPORT")
    ThisDrawing.ActiveViewport = vportObj
    ' Split vport into 4 windows
    vportObj.Split acViewport4
    ' Iterate through the viewports,
    ' highlighting each viewport and displaying
    ' the upper right and lower left corners
    ' for each.
    Dim vport As AcadViewport
    Dim LLCorner As Variant
    Dim URCorner As Variant
    For Each vport In ThisDrawing.Viewports
        ThisDrawing.ActiveViewport = vport
        LLCorner = vport.LowerLeftCorner
        URCorner = vport.UpperRightCorner
        MsgBox "Viewport: " & vport.Name & " is now active." & _
               vbCrLf & "Lower left corner: " & _
               LLCorner(0) & ", " & LLCorner(1) & vbCrLf & _
               "Esquina superior derecha: " & _
               URCorner(0) & ", " & URCorner(1)
    Next vport
End Sub
```

¿Comentarios?

Actualización de la geometría en la ventana de documento

Muchas de las acciones que lleva a cabo a través de AutoCAD ActiveX Automation modifican lo que se muestra en el dibujo AutoCAD. No todas estas acciones actualizan inmediatamente la visualización del dibujo. De esta forma es posible efectuar varios cambios en el dibujo sin tener que esperar a que se actualice la pantalla después de cada acción. En su lugar, puede efectuar varios cambios consecutivos y, cuando termine, realizar una sola llamada para actualizar la pantalla.

Los métodos que actualizan la pantalla son Update y Regen.

El método Update actualiza la presentación en pantalla de un solo objeto. El método Regen actualiza todo el dibujo y vuelve a calcular las coordenadas de pantalla de todos los objetos. También regenera el índice de la base de datos de dibujo para mejorar la visualización y la designación de objetos.

Actualización de la presentación en pantalla de un solo objeto

Este ejemplo crea un círculo. A continuación, actualiza el círculo con el método Update para que sea visible en AutoCAD.

```
Sub Ch3_UpdateDisplay()
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 1: center(1) = 1: center(2) = 0
    radius = 1
    ' Create the circle
    Set circleObj = ThisDrawing.ModelSpace.AddCircle(center, radius)
    ' Update the circle
    circleObj.Update
End Sub
```

¿Comentarios?

Restablecimiento de objetos activos

Los cambios que se realizan en la mayoría de los objetos activos, como la capa activa o el tipo de línea activo, se ven en pantalla de inmediato. Sin embargo, hay algunos objetos activos que se deben restablecer para reflejar los cambios realizados en ellos. Se trata de los objetos correspondientes al estilo de texto activo, al sistema de coordenadas personales (SCP) activo y a la ventana gráfica activa. Si se realizan cambios en alguno de estos objetos, es preciso restablecer el objeto en cuestión y llamar al método Regen para que los cambios sean visibles.

Para restablecer los objetos, sólo es necesario definir la propiedad ActiveTextStyle, ActiveUCS o ActiveViewport utilizando el objeto actualizado.

Restablecimiento de la ventana gráfica activa

En el ejemplo siguiente se cambia la presentación de la rejilla en la ventana gráfica activa y después se restablece como ventana gráfica activa para mostrar el cambio.

```
Sub Ch3_ResetActiveViewport()
    ' Toggle the setting of the grid display
    ' for the active viewport
    ThisDrawing.ActiveViewport.GridOn = _
              Not (ThisDrawing.ActiveViewport.GridOn)
    ' Reset the active viewport
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
End Sub
```

Definición y devolución de variables de sistema

El objeto Document proporciona los métodos SetVariable y GetVariable para la configuración y recuperación de variables de sistema de AutoCAD. Por ejemplo, para asignar un entero a la variable de sistema MAXSORT, utilice el siguiente código:

```
ThisDrawing.SetVariable "MAXSORT", 100
```

<$nopage>métodos.

Dibujo con precisión

Con AutoCAD puede crear sus dibujos con una geometría precisa sin realizar largos cálculos. A menudo se pueden indicar puntos concretos sin conocer las coordenadas. También se pueden realizar cálculos sobre el dibujo y presentar en pantalla varios tipos de información de estado sin salir de la pantalla de dibujo.

En la actualidad, ActiveX Automation de AutoCAD no dispone de métodos para realizar las siguientes funciones de AutoCAD:

- Definición de referencias a objetos
- Especificación de intervalos graduados en objetos o división de objetos en segmentos.
- **Alineación de la resolución y la rejilla**
- **Utilización del modo Orto**
- **Trazado de líneas auxiliares**
- **Cálculo de puntos y valores**
- **Cálculo de áreas**

Alineación de la resolución y la rejilla

Puede utilizar la rejilla como guía visual y activar el modo Forzcursor para limitar el movimiento del cursor. Además de determinar el intervalo, puede ajustar la alineación de la rejilla y la malla. Puede girar la alineación o bien definirla para su utilización con dibujos isométricos

Si es necesario dibujar con una alineación específica o en un cierto ángulo, se puede rotar el ángulo de la malla. El punto central de rotación del ángulo de la malla es el punto base de la malla. Si necesita alinear un patrón de sombreado, puede cambiar este punto, que habitualmente se define como 0,0.

Para rotar el ángulo de la malla, utilice la propiedad SnapRotationAngle Si necesita cambiar el punto base de rotación del ángulo de la malla, utilice la propiedad SnapBasePoint

Nota Ambas propiedades requieren una llamada al método Update para actualizar la pantalla de AutoCAD.

Para obtener más información acerca del uso y establecimiento de mallas y rejillas, véase “Ajuste de la rejilla y la referencia rejilla” en el *Manual del usuario*.

Modificación del punto base de la malla de resolución y del ángulo de rotación

En este ejemplo se cambia el punto base de la malla de resolución a (1,1) y el ángulo de rotación de la malla a 30 grados. La rejilla se activa para que los cambios sean visibles.

```
Sub Ch3_ChangeSnapBasePoint()
    ' Turn on the grid for the active viewport
    ThisDrawing.ActiveViewport.GridOn = True
    ' Change the snap base point to 1, 1
```

```
    Dim newBasePoint(0 To 1) As Double
    newBasePoint(0) = 1: newBasePoint(1) = 1
    ThisDrawing.ActiveViewport.SnapBasePoint = newBasePoint
    ' Change the snap rotation angle to 30 degrees (0.575 radians)
    Dim rotationAngle As Double
    rotationAngle = 0.575
    ThisDrawing.ActiveViewport.SnapRotationAngle = rotationAngle
    ' reset the viewport
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
End Sub
```

¿Comentarios?

Utilización del modo Orto

Cuando dibuja líneas o mueve objetos, puede utilizar el modo Orto para restringir el cursor al eje horizontal o vertical. La alineación ortogonal depende del ángulo de forzado del cursor o del SCP. El modo Ortho funciona con actividades que requieren que se especifique un segundo punto. No sólo se puede utilizar el modo Ortho para establecer la alineación vertical u horizontal, sino también para asegurar el paralelismo o para crear desfases regulares.

Al permitir a AutoCAD imponer limitaciones ortogonales, se consigue dibujar con mayor rapidez. Por ejemplo, puede crear una serie de líneas perpendiculares si activa el modo Orto antes de empezar a dibujar. Dado que las líneas se limitarán a los planos horizontal y vertical, se pueden dibujar más rápido con la seguridad de que serán perfectamente perpendiculares.

A medida que desplaza el cursor, una línea elástica que determina el desplazamiento verá forzada su trayectoria al eje horizontal o al vertical, dependiendo de cuál esté más próximo al cursor. El modo Orto no tiene validez en las vistas en perspectiva en AutoCAD, cuando se introducen coordenadas en la línea de comando o cuando se especifica una referencia a un objeto.

Para activar o desactivar el modo Orto, utilice la propiedad OrthoOn Esta propiedad necesita una entrada Booleana. Defínala como TRUE para activar el modo Orto y como FALSE para desactivarlo. Por ejemplo, la siguiente instrucción activa el modo Orto en la ventana gráfica activa:

```
ThisDrawing.ActiveViewport.OrthoOn = True
```

¿Comentarios?

<$nopage>métodos.

Trazado de líneas auxiliares

Puede crear líneas auxiliares que se prolonguen hasta el infinito en una o ambas direcciones: las líneas auxiliares que se extienden en una dirección se conocen como rayos. Las líneas auxiliares que se extienden en ambas direcciones se conocen también como líneasX. Estas líneas auxiliares pueden utilizarse como referencia para crear otros objetos. Por ejemplo, puede utilizar líneas auxiliares para encontrar el centro de un triángulo, preparar vistas múltiples del mismo elemento, o crear intersecciones temporales que pueden utilizarse para las referencias a objetos.

- **Creación de líneas auxiliares**
- **Consulta de líneas auxiliares**
- **Creación de rayos**
- **Consulta de rayos**

<$nopage>métodos.

Creación de líneas auxiliares

La líneas auxiliares se pueden colocar en cualquier lugar del espacio 3D y se extienden hasta el infinito en ambas direcciones. Para crear una línea auxiliar, utilice el método AddXLine Con este método se utiliza la técnica de dos puntos para precisar la línea: el usuario introduce o designa dos puntos para definir la orientación. El primer punto, el punto raíz, se considerará como el punto medio de la línea auxiliar.

Adición de una línea auxiliar

El siguiente código del ejemplo crea un objeto XLine mediante los dos puntos (5, 0, 0) y (1, 1, 0).

```
Sub Ch3_AddXLine()
    Dim xlineObj As AcadXline
    Dim basePoint(0 To 2) As Double
    Dim directionVec(0 To 2) As Double
    ' Define the xline
    basePoint(0) = 2#: basePoint(1) = 2#: basePoint(2) = 0#
    directionVec(0) = 1#: directionVec(1) = 1#: directionVec(2) = 0#
    ' Create the xline in model space
    Set xlineObj = ThisDrawing.ModelSpace.AddXLine _
                   (basePoint, directionVec)
    ThisDrawing.Application.ZoomAll
End Sub
```

Consulta de líneas auxiliares

Una vez creada una línea auxiliar, puede utilizar la propiedad BasePoint para consultar su primer punto. El segundo punto utilizado en la creación de la línea auxiliar no se almacena con el objeto. En su lugar, utilice la propiedad DirectionVector para obtener el vector de dirección de la línea auxiliar.

Consulta de una línea auxiliar

En este ejemplo se buscan el punto de base y el vector de dirección de la línea auxiliar creada en Adición de una línea auxiliar

```
Dim BPoint As Variant
Dim Vector As Variant
BPoint = xlineObj.basePoint
Vector = xlineObj.DirectionVector
```

Creación de rayos

Un rayo es una línea en espacio 3D que comienza en un punto dado y se extiende hasta el infinito. A diferencia de las líneas auxiliares X, que se extienden en dos direcciones, los rayos sólo se extienden en una dirección. En consecuencia, los rayos ayudan a reducir la confusión visual que se produce cuando hay muchas líneas auxiliares.

Al igual que ocurre con las líneas auxiliares, los comandos que muestran la extensión del dibujo no tienen en cuenta los rayos.

Consulta de rayos

Una vez creado el rayo, puede utilizar la propiedad BasePoint para consultar su primer punto. El segundo punto utilizado en la creación del rayo no se almacena con el objeto. En su lugar, utilice la propiedad DirectionVector para obtener el vector de dirección del rayo.

Adición, consulta y edición de un objeto Ray

El código del ejemplo siguiente crea un objeto Ray utilizando los puntos (5, 0, 0) y (1, 1, 0). A continuación, consulta el punto base y el vector de dirección activos y presenta los resultados en un cuadro de mensajes. Por último, cambia el vector de dirección y consulta y muestra en pantalla el nuevo vector y el punto de base.

```
Sub Ch3_EditRay()
    Dim rayObj As AcadRay
    Dim basePoint(0 To 2) As Double
    Dim secondPoint(0 To 2) As Double
    ' Define the ray
    basePoint(0) = 3#: basePoint(1) = 3#: basePoint(2) = 0#
    secondPoint(0) = 4#: secondPoint(1) = 4#: secondPoint(2) = 0#
    ' Creates a Ray object in model space
    Set rayObj = ThisDrawing.ModelSpace.AddRay _
                 (basePoint, secondPoint)
    ThisDrawing.Application.ZoomAll
    ' Find the current status of the Ray
    MsgBox "The base point of the ray is: " & _
           rayObj.basePoint(0) & ", " & _
           rayObj.basePoint(1) & ", " & _
           rayObj.basePoint(2) & vbCrLf & _
           "The directional vector for the ray is: " & _
          rayObj.DirectionVector(0) & ", " & _
          rayObj.DirectionVector(1) & ", " & _
          rayObj.DirectionVector(2), , "Edit Ray"
    ' Change the directional vector for the ray
```

```
    Dim newVector(0 To 2) As Double
    newVector(0) = -1
    newVector(1) = 1
    newVector(2) = 0
    rayObj.DirectionVector = newVector
    ThisDrawing.Regen False
    MsgBox "The base point of the ray is: " & _
           rayObj.basePoint(0) & ", " & _
           rayObj.basePoint(1) & ", " & _
           rayObj.basePoint(2) & vbCrLf & _
           "The directional vector for the ray is: " & _
          rayObj.DirectionVector(0) & ", " & _
          rayObj.DirectionVector(1) & ", " & _
          rayObj.DirectionVector(2), , "Edit Ray"
End Sub
```

¿Comentarios?

Cálculo de puntos y valores

Los métodos que proporciona el objeto Utility permiten resolver rápidamente un problema matemático o localizar puntos en el dibujo. Con los métodos del objeto Utility puede realizar lo siguiente:

- calcular el ángulo que forma una línea con el eje *X*, mediante el método AngleFromXAxis
- convertir un ángulo en forma de cadena en un valor real (doble), mediante el método AngleToReal
- convertir un ángulo en forma de valor real (doble) en una cadena, mediante el método AngleToString
- convertir una distancia en forma de cadena en un valor real (doble), mediante el método DistanceToReal
- crear un variante que contiene una matriz de enteros, números flotantes, dobles, etc. con el método CreateTypedArray
- calcular el punto situado en un ángulo y a una distancia concretos de un punto determinado, con el método PolarPoint
- convertir un punto de un sistema de coordenadas a otro, mediante el método TranslateCoordinates
- calcular la distancia entre dos puntos especificados por el usuario, con el método GetDistance

Cálculo de la distancia entre dos puntos con el método GetDistance

En este ejemplo se utiliza el método GetDistance para obtener las coordenadas del punto y la función `MsgBox` para mostrar la distancia calculada.

```
Sub Ch3_GetDistanceBetweenTwoPoints()
    Dim returnDist As Double
    ' Return the value entered by user. A prompt is provided.
    returnDist = ThisDrawing.Utility.GetDistance _
                 (, "Pick two points.")
    MsgBox "The distance between the two points is: " & returnDist
End Sub
```

¿Comentarios?

Cálculo de áreas

Cálculo del área definida por los puntos introducidos por el usuario Puede conocer el área de un arco, círculo, elipse, polilínea optimizada, polilínea, región o curvas spline planas-cerradas mediante la propiedad Área.

Si necesita calcular el área combinada de más de un objeto, puede mantener un total cuando añade o usa el método Boolean en una serie de regiones para obtener una sola región que represente el área deseada. Desde esta única región puede utilizar la propiedad Area para obtener su área.

El área del cálculo varía en función del tipo de objeto designado. Para obtener información acerca de cómo se calcula el área de cada tipo de objeto, véase “Obtención de información de área” en el *Manual del usuario*.

- **Cálculo de un área definida**

Cálculo de un área definida

Cálculo del área definida por los puntos introducidos por el usuario Obtiene el área especificada por los puntos introducidos por el usuario. Puede medir una región cerrada arbitraria delimitada por los puntos 2D o 3D especificados por el usuario. Los puntos deben ser coplanares.

Para obtener el área designada por puntos del usuario

1. Utilice el método GetPoint en un bucle para obtener los puntos del usuario.
2. Cree una polilínea optimizada a partir de los puntos facilitados por el usuario. Utilice el método AddLightweightPolyline para crear la polilínea.
3. Utilice la propiedad Area para obtener el área de la polilínea recién creada.
4. Borre la polilínea con el método Erase.

Cálculo del área definida por los puntos introducidos por el usuario

En este ejemplo, se pide al usuario que designe cinco puntos. Después se crea una polilínea a partir de los puntos. La polilínea es cerrada y su área se muestra en un cuadro de mensaje.

```
Sub Ch3_CalculateDefinedArea()
    Dim p1 As Variant
    Dim p2 As Variant
    Dim p3 As Variant
    Dim p4 As Variant
    Dim p5 As Variant
    ' Get the points from the user
```

```
    p1 = ThisDrawing.Utility.GetPoint(, vbCrLf & "First point: ")
    p2 = ThisDrawing.Utility.GetPoint(p1, vbCrLf & "Second point: ")
    p3 = ThisDrawing.Utility.GetPoint(p2, vbCrLf & "Third point: ")
    p4 = ThisDrawing.Utility.GetPoint(p3, vbCrLf & "Fourth point: ")
    p5 = ThisDrawing.Utility.GetPoint(p4, vbCrLf & "Fifth point: ")
    ' Create the 2D polyline from the points
    Dim polyObj As AcadLWPolyline
    Dim vertices(0 To 9) As Double
    vertices(0) = p1(0): vertices(1) = p1(1)
    vertices(2) = p2(0): vertices(3) = p2(1)
    vertices(4) = p3(0): vertices(5) = p3(1)
    vertices(6) = p4(0): vertices(7) = p4(1)
    vertices(8) = p5(0): vertices(9) = p5(1)
    Set polyObj = ThisDrawing.ModelSpace.AddLightWeightPolyline _
                  (vertices)
    polyObj.Closed = True
    ThisDrawing.Application.ZoomAll
    ' Display the area for the polyline
    MsgBox "The area defined by the points is " & _
           polyObj.Area, , "Calculate Defined Area"
End Sub
```

¿Comentarios?

Solicitud de datos de usuario

El objeto Utility, un subordinado del objeto Document, define los métodos de entrada de información por parte del usuario. Dichos métodos presentan una solicitud en la línea de comando de AutoCAD y piden entradas de diversos tipos. Este tipo de entrada del usuario es muy útil para la introducción interactiva de coordenadas de pantalla, selección de entidades y valores de cadena corta o numéricos. Si una aplicación requiere la introducción de varios valores u opciones, puede ser más práctico proporcionar un cuadro de diálogo que presentar solicitudes individuales.

Cada método presenta una solicitud en la línea de comando de AutoCAD y da como resultado un valor específico del tipo de entrada solicitada. Por ejemplo, GetString devuelve una cadena, GetPoint devuelve una variante (que contiene una matriz de dobles de tres elementos) y GetInteger devuelve un valor entero. También puede controlar la entrada del usuario con el método InitializeUserInput Por ejemplo, este método permite controlar la entrada `NULL` (pulsando la tecla INTRO), la introducción de cero o números negativos y la de valores de texto arbitrarios.

Si desea que la solicitud se muestre en una línea aparte, incluya la constante de retorno de carro/salto de línea `(vbCrLf)` al principio de la cadena de la solicitud.

- **Método GetString**
- **Método GetPoint**
- **Método GetKeyword**
- **Control de entradas del usuario**

Método GetString

El método GetString pide al usuario la introducción de una cadena en la solicitud de comando de AutoCAD. Este método admite dos parámetros. El primer parámetro controla la inserción de espacios en la cadena de entrada. Si se establece en 0, los espacios no se permiten (BARRA ESPACIADORA termina la entrada); si se le da el valor 1, la cadena puede contener espacios (debe pulsarse INTRO para terminar la entrada). El segundo parámetro es la cadena de la solicitud.

Obtención del valor de una cadena del usuario en la línea de comando de AutoCAD

En el ejemplo siguiente se muestra la solicitud Enter Your Name y se indica al usuario que pulse ENTER para terminar la entrada (la cadena de entrada admite la inclusión de espacios). El valor de la cadena se guarda en la variable `retVal` y se muestra en un cuadro de mensaje.

```
Sub Ch3_GetStringFromUser()
    Dim retVal As String
    retVal = ThisDrawing.Utility.GetString _
             (1, vbCrLf & "Enter your name: ")
    MsgBox "The name entered Wad: " & retal
End Sub
```

El método GetString no respeta una llamada anterior al método InitializeUserInput

Método GetPoint

El método GetString solitita al usuario que designe un punto en la solicitud de comando de AutoCAD. Este método admite dos parámetros, un punto From opcional y la cadena de solicitud. Si se proporciona el punto From, AutoCAD traza una línea elástica desde dicho punto. Para controlar la entrada del usuario, puede anteponerse a este método una llamada al método InitializeUserInput

Obtención de un punto elegido por el usuario

En el ejemplo siguiente se solicita al usuario la selección de dos puntos, inicial y final, para dibujar una línea entre ellos.

```
Sub Ch3_GetPointsFromUser()
    Dim startPnt As Variant
    Dim endPnt As Variant
    Dim prompt1 As String
    Dim prompt2 As String
    prompt1 = vbCrLf & "Enter the start point of the line: "
    prompt2 = vbCrLf & "Enter the end point of the line: "
    ' Get the first point without entering a base point
    startPnt = ThisDrawing.Utility.GetPoint(, prompt1)
    ' Use the point entered above as the base point
    endPnt = ThisDrawing.Utility.GetPoint(startPnt, prompt2)
    ' Create a line using the two points entered
    ThisDrawing.ModelSpace.AddLine startPnt, endPnt
    ThisDrawing.Application.ZoomAll
End Sub
```

Método GetKeyword

El método GetKeyword pide al usuario la introducción de una palabra clave en la solicitud de comando de AutoCAD. Este método admite sólo un parámetro: la cadena de la solicitud. Las palabras clave y los parámetros de entrada se definen con una llamada al método InitializeUserInput

Obtención de una palabra clave introducida por el usuario en la línea de comando de AutoCAD

En el ejemplo siguiente se insta al usuario a escribir una palabra clave, estableciendo el primer parámetro de InitializeUserInput en 1, lo que desactiva la entrada `NULL` (al pulsar INTRO). El segundo parámetro establece la lista de palabras clave válidas.

```
Sub Ch3_KeyWord()
    Dim keyWord As String
    ThisDrawing.Utility.InitializeUserInput 1, "Line Circle Arc"
    keyWord = ThisDrawing.Utility.GetKeyword _
              (vbCrLf & "Enter an option (Line/Circle/Arc): ")
    MsgBox keyWord, , "GetKeyword Example"
End Sub
```

Una solicitud de palabra clave más sencilla para los usuarios proporciona un valor por defecto cuando el usuario pulse `INTRO` (entrada NULL). Observe las pequeñas modificaciones realizadas en el siguiente ejemplo:

```
Sub Ch3_KeyWord2()
    Dim keyWord As String
    ThisDrawing.Utility.InitializeUserInput 0, "Line Circle Arc"
    keyWord = ThisDrawing.Utility.GetKeyword _
              (vbCrLf & "Enter an option (Line/Circle/<Arc>): ")
    If keyWord = "" Then keyWord = "Arc"
    MsgBox keyWord, , "GetKeyword Example"
```

```
End Sub
```

¿Comentarios?

Control de entradas del usuario

Puede utilizar el método InitializeUserInput para definir palabras clave o restringir el tipo de entradas en el método de introducción de información por parte del usuario. El uso y los valores de los parámetros son similares a la función de AutoLISP **`initget`**. InitializeUserInput puede utilizarse con los siguientes métodos: GetAngle, GetCorner, GetDistance, GetInteger, GetKeyword, GetOrientation, GetPoint, y GetReal. InitializeUserInput no puede utilizarse con el método GetString. Utilice el método GetInput para recuperar el valor de cadena (palabra clave o entrada al azar) cuando el método de entradas del usuario no devuelva un valor de cadena.

El método InitializeUserInput admite dos parámetros. El primer parámetro es un valor entero, expresado en bits, que determina las opciones de entrada del método de introducción de información por parte del usuario. El segundo parámetro es una cadena que define las palabras clave válidas.

Obtención de un valor entero o de una palabra clave en la línea de comando de AutoCAD

En el ejemplo siguiente se solicita al usuario la designación de un valor entero positivo o una palabra clave:

```
Sub Ch3_UserInput()
    ' The first parameter of InitializeUserInput (6)
    ' restricts input to positive and non-negative
    ' values. The second parameter is the list of
    ' valid keywords.
    ThisDrawing.Utility.InitializeUserInput 6, "Big Small Regular"
    ' Set the prompt string variable
    Dim promptStr As String
    promptStr = vbCrLf & "Enter the size or (Big/Small/<Regular>):"
    ' At the GetInteger prompt, entering a keyword or pressing
    ' ENTER without entering a value results in an error. To allow
```

```
    ' your application to continue and check for the error
    ' description, you must set the error handler to resume on error
    On Error Resume Next
    ' Get the value entered by the user
    Dim returnInteger As Integer
    returnInteger = ThisDrawing.Utility.GetInteger(promptStr)
    ' Check for an error. If the error number matches the
    ' one shown below, then use GetInput to get the returned
    ' string; otherwise, use the value of returnInteger.
    If Err.Number = -2145320928 Then
        Dim returnString As String
        Debug.Print Err.Description
        returnString = ThisDrawing.Utility.GetInput()
        If returnString = "" Then 'ENTER returns null string
            returnString = "Regular" 'Set to default
        End If
        Err.Clear
    Else 'Otherwise,
        returnString = returnInteger 'Use the value entered
    End If
    ' Display the result
    MsgBox returnString, , "InitializeUserInput Example"
End Sub
```

Acceso a la línea de comando de AutoCAD

El método SendCommand permite enviar comandos directamente a la línea de comando de AutoCAD. Este método envía una cadena simple a la línea de comando. Es necesario que los argumentos del comando en la cadena estén en el orden que espera la secuencia de solicitud del comando ejecutado. Un espacio en blanco o el equivalente ASCII de un retorno de carro en la cadena equivale a pulsar la tecla INTRO. A diferencia del entorno AutoLISP, aquí no se puede llamar al método SendCommand sin argumentos.

Envío de un comando a la línea de comando de AutoCAD

En el ejemplo siguiente se crea un círculo de radio 4 con centro en (2, 2, 0). Después se aplica el zoom a toda la geometría del dibujo. Observe el espacio al final de la cadena que representa el INTRO final para iniciar la ejecución del comando.

```
Sub Ch3_SendACommandToAutoCAD()
   ThisDrawing.SendCommand "_Circle 2,2,0 4 "
   ThisDrawing.SendCommand "_zoom a "
End Sub
```

Trabajo sin documentos abiertos

AutoCAD siempre arranca con un documento nuevo o uno existente abiertos. Durante la sesión, no obstante, se pueden cerrar todos los documentos.

Si cierra todos los documentos en la interfaz del usuario de AutoCAD observará algunos cambios en la ventana de la aplicación. Los menús disponibles quedan reducidos a Archivo, Ver, Ventana y Ayuda. Las opciones de estos menús también ven reducido su número. Asimismo, la línea de comando desaparece.

De forma similar, cuando no hay documentos abiertos la interfaz de ActiveX sólo permite las acciones siguientes:

- abrir un documento.
- crear un documento.
- importar un documento.
- salir de AutoCAD.

Todas estas acciones se encuentran disponibles en la colección Documents. Los métodos y las propiedades de la colección Documents, junto con un conjunto limitado de métodos y propiedades del objeto Application, constituyen la única interfaz válida cuando no hay documentos abiertos. Si se realiza cualquier otra acción, por ejemplo, intentar acceder a opciones del usuario, el resultado será fallido.

Utilice la propiedad Count de la colección Documents para determinar si AutoCAD está en un estado de cero documentos. Si `Documents.Count = 0`, significa que AutoCAD tiene un estado de cero documentos. Si `Documents.Count > 0`, indica que hay un documento abierto como mínimo.

También es importante tener en cuenta que en VBA el objeto `ThisDrawing` no

se define cuando AutoCAD está en estado de documento cero. Esto es lógico, puesto que `ThisDrawing` se refiere generalmente al dibujo activo y en estado de documento cero no hay documentos abiertos. Si se intenta ejecutar una macro que utiliza `ThisDrawing` el resultado será un error en tiempo de ejecución. Puede evitar este error si utiliza la función `GetObject` de VBA y especifica la versión de AutoCAD para conectarse con AutoCAD cuando no hay documentos abiertos.

Importación de otros formatos de archivo

Se pueden emplear dibujos o imágenes de otras aplicaciones abriéndolos en formatos específicos. AutoCAD es capaz de realizar cierta conversión para archivos DXF™ de intercambio de dibujos, SAT y WMF. Independientemente de la versión que esté empleando, puede importar el archivo utilizando el método Import. Este método requiere tres valores de entrada: el nombre del archivo que se debe importar, el punto de inserción en el dibujo donde se colocará el archivo y el factor de escala que debe emplearse al colocar el dibujo importado.

Exportación a otros formatos de archivo

Si desea utilizar un dibujo de AutoCAD en otra aplicación, puede convertirlo a un formato específico por medio del método Export. Este método permite exportar los dibujos de AutoCAD al formato WMF, SAT, EPS, DXF o BMP. El método Export requiere tres valores de entrada: el nombre del archivo que se creará, la extensión del nuevo archivo y el conjunto de selección de objetos para exportar.

Cuando realice una exportación a formatos WMF, SAT o BMP, debe proporcionar un conjunto de selección que no esté vacío. El conjunto de selección designa los objetos del dibujo que se desea exportar. Si no se precisa un conjunto de selección no se exporta nada, y el resultado es un error interceptable de argumento no válido.

Cuando se exporta a los formatos EPS o DXF, Export no considera, aunque lo requiere, el argumento del conjunto de selección. Se exporta automáticamente todo el dibujo.

Para exportar un dibujo como archivo DXF y volverlo a importar

En este ejemplo se crea un círculo en el dibujo actual. Después se exporta el dibujo a un archivo llamado *DXFExprt.DXF*. A continuación, se abre un dibujo nuevo y se importa el archivo. Tenga presente que a Export se le proporciona como argumento un conjunto de selección vacío. Aunque el método Export omite la información del conjunto de selección al exportar un archivo DXF, produciría un error si no se incluyera el argumento.

```
Sub Ch3_ImportingAndExporting()
    ' Create the circle for visual representation
    Dim circleObj As AcadCircle
    Dim centerPt(0 To 2) As Double
    Dim radius As Double
```

```
    centerPt(0) = 2: centerPt(1) = 2: centerPt(2) = 0
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace.AddCircle _
                    (centerPt, radius)
    ThisDrawing.Application.ZoomAll
    ' Create an empty selection set
    Dim sset As AcadSelectionSet
    Set sset = ThisDrawing.SelectionSets.Add("NEWSSET")
    'Export the current drawing to a DXF file in the
    ' AutoCAD temporary file directory
    Dim tempPath As String
    Dim exportFile As String
    Const dxfname As String = "DXFExprt"
    tempPath = _
       ThisDrawing.Application.preferences.Files.TempFilePath
    exportFile = tempPath & dxfname
    ThisDrawing.Export exportFile, "DXF", sset
    ' Delete the empty selection set
    ThisDrawing.SelectionSets.Item("NEWSSET").Delete
    ' Open a new drawing
    ThisDrawing.Application.Documents.Add "acad.dwt"
    ' Define the import
    Dim importFile As String
    Dim insertPoint(0 To 2) As Double
    Dim scalefactor As Double
    importFile = tempPath & dxfname & ".dxf"
    insertPoint(0) = 0: insertPoint(1) = 0: insertPoint(2) = 0
    scalefactor = 2#
    ' Import the file
    ThisDrawing.Import importFile, insertPoint, scalefactor
    ThisDrawing.Application.ZoomAll
End Sub
```

<$nopage>relleno sólido (áreas):<$startrange>conjuntos de selección:filtros (listas), <$startrange>filtros (listas), <$startrange>filtrar:conjuntos de selección, DXF (códigos), y tipos de filtros (tabla), filtros (tipos), y códigos DXF (tabla), <$startrange>filtros (listas):código de ejemplo, SelectionSet (objeto):código de ejemplo, filtrar:código de ejemplo, <$endrange>conjuntos de selección:filtros (listas), <$endrange>filtros (listas), <$endrange>filtrar:conjuntos de selección, <$endrange>filtros (listas):código de ejemplo, conjuntos de selección:eliminar objetos, objetos:eliminar de conjuntos de selección, RemoveItems (método):en conjuntos de selección, Clear (método):en conjuntos de selección, Erase (método):en conjuntos de selección, Delete (método):en conjuntos de selección, objetos:existentes (modificar), Update (método):volver a dibujar objetos, objetos 2D:editar, editar:objetos 2D, no gráficos (objetos), editar, editar:objetos no gráficos, objetos guardados:especificar, objetos:guardados (especificar) , objetos guardados:limpiar, limpiar (objetos guardados), PurgeAll (método):código de ejemplo, objetos guardados:cambiar nombre, objetos guardados:longitud de caracteres, Name (propiedad):código de ejemplo, Add (método):capas, código de ejemplo, Layer (objeto):código de ejemplo, copiar:desfasar, copiar:reflejar, copiar:usar matrices, desfasar, objetos, reflejar:objetos, disponer en matrices, patrones, Copy (método), copiar:objeto único, CopyObjects (método), copiar:varios objetos, copiar:de un dibujo a otro, copiar:objetos a otros dibujos, CopyObjects (método):código de ejemplo, Circle (objeto):código de ejemplo, CopyObjects (método):código de ejemplo, copiar:de un dibujo a otro, copiar:objetos a otros dibujos, objetos:desfasar, Offset (método), AddLightweightPolyline (método):código de ejemplo, LightweightPolyline (objeto):código de ejemplo, Offset (método):código de ejemplo, Offset (método):código de ejemplo, Mirror (método), reflejar:con dos coordenadas, Mirror (método):ilustración, Erase (método), MIRRTEXT (variable de sistema), Text (objeto):reflejar texto, reflejar:objetos Text, reflejar:código de ejemplo, LightweightPolyline (objeto):código de ejemplo, Mirror (método):código de ejemplo, matrices:matrices polares, matrices:matrices rectangulares, ArrayPolar (método):crear matrices, matrices polares:crear, matrices polares:punto central (especificar), matrices polares:referencia (puntos), referencia (puntos en matrices polares), matrices polares:código de ejemplo, ArrayRectangular (método), matrices:rectangulares, matrices rectangulares, ángulo de rotación de resolución:matrices rectangulares, matrices rectangulares:ángulo de rotación de resolución, SnapRotationAngle (propiedad), matrices rectangulares:código de ejemplo, vectores:desplazar objetos, objetos:desplazar a lo largo de un vector, Move (método):vectores, vector de desplazamiento, Move (objeto):ilustración, Move (objeto):código de ejemplo, girar objetos:ilustración, punto base, girar

(propiedad), TextGenerationFlag (propiedad):código de ejemplo, Text (objeto):código de ejemplo, Text (objeto):texto de una línea (crear), Text (objeto):aplicar formato, StyleName (propiedad), Text (objeto):propiedades (lista), Alignment (propiedad), InsertionPoint (propiedad), ObliqueAngle (propiedad), Rotation (propiedad), ScaleFactor (propiedad), TextAlignmentPoint (propiedad), TextGenerationFlag (propiedad), TextString (propiedad), Update (método):Text (objeto), Alignment (propiedad):en texto, Text (objeto):alinear en dibujos (ilustración), Text (objeto):código de ejemplo, SetVariable (método):código de ejemplo, Alignment (propiedad):código de ejemplo, TextAlignmentPoint (propiedad):código de ejemplo, Text (objeto):modificar, MIRRTEXT (variable de sistema), Text (objeto):métodos (lista), ArrayPolar (método):Text (objeto), ArrayRectangular (método):Text (objeto), Copy (método), Erase (método), Mirror (método), Move (método), Rotate (método), textoM:usos, textoM:modificar, <$nopage>texto de líneas múltiples.

Creación y edición de entidades de AutoCAD

Con AutoCAD podrá crear una amplia variedad de objetos, desde líneas y círculos hasta curvas spline, elipses y áreas de sombreado asociativo. Por lo general, se añaden objetos al espacio modelo utilizando uno de los métodos Add. También se pueden crear objetos en espacio papel o en un bloque.

Una vez creado un objeto, se puede cambiar la capa, el color y el tipo de línea del mismo. También se puede añadir texto para incluir notas en el dibujo.

¿Comentarios?

<$nopage>relleno sólido (áreas):

Creación de objetos

Aunque existen diferentes formas de crear el mismo objeto gráfico en AutoCAD®, ActiveX Automation sólo ofrece un método de creación para cada objeto. Por ejemplo, existen en AutoCAD cuatro maneras diferentes de crear un círculo: (1) especificando el punto central y el radio, (2) mediante dos puntos que definen el diámetro, (3) mediante tres puntos que definan la circunferencia, o (4) mediante dos tangentes y el radio. Sin embargo, en ActiveX Automation sólo se crean círculos con el método que utiliza el centro y el radio.

Nota Los métodos de creación de objetos de VB y VBA que utilizan CreateObject o Dim con la palabra clave New sólo permiten crear el objeto Application de AutoCAD. Todos los demás objetos de AutoCAD deben crearse mediante los métodos Add o Add<*nombreobjeto*> que proporciona la interfaz de AutoCAD.

Determine Determinación del objeto contenedor

Los objetos gráficos se crean en las colecciones ModelSpace y PaperSpace, y en los objetos Block.

La colección ModelSpace la devuelve la propiedad ModelSpace, mientras que la propiedad PaperSpace devuelve la colección PaperSpace.

La referencia a estos objetos se puede realizar directamente o a través de una variable definida por el usuario. Para utilizar una referencia directa a un objeto, inclúyalo en la jerarquía de la llamada. Por ejemplo, la siguiente instrucción agrega una línea al espacio modelo:

```
Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint,endPoint)
```

Para hacer referencia a los objetos mediante una variable definida por el usuario, asigne a la variable el tipo `AcadModelSpace` o `AcadPaperSpace` y después defínala como la propiedad apropiada del documento activo. En el siguiente ejemplo se definen dos variables, que se establecen igual que los espacios modelo y papel activos, respectivamente:

```
Dim moSpace As AcadModelSpace
Dim paSpace As AcadPaperSpace
Set moSpace = ThisDrawing.ModelSpace
Set paSpace = ThisDrawing.PaperSpace
```

La siguiente instrucción agrega una línea al espacio modelo mediante la variable definida por el usuario:

```
Set lineObj = moSpace.AddLine(startPoint,endPoint)
```

Creación de líneas

La línea es el objeto más sencillo de AutoCAD. Pueden crearse diversas líneas, líneas individuales y varios segmentos de línea con o sin arcos. En general, las líneas se dibujan designando puntos de coordenadas. El tipo de línea por defecto es CONTINUOUS (línea continua), pero hay varios tipos de línea posibles que utilizan puntos y rayas.

Para crear una línea, utilice uno de los métodos siguientes:

AddLine

Crea una línea que pasa por dos puntos.

AddLightweightPolyline

Crea una polilínea optimizada 2D a partir de una lista de vértices.

AddMLine

Crea una línea múltiple.

AddPolyline

Crea una polilínea 2D o 3D.

Las líneas estándar y las polilíneas se crean en el plano *XY* del sistema de coordenadas universales. Las polilíneas y las polilíneas optimizadas se crean en el Sistema de coordenadas de objeto (SCO). Para obtener información acerca de la conversión de coordenadas SCO, véase Conversión de coordenadas.

Creación de un objeto Polyline

Este ejemplo aplica el método AddLightweightPolyline para crear una polilínea sencilla de dos segmentos utilizando las coordenadas 2D (2,4), (4,2) y (6,4).

```
Sub Ch4_AddLightWeightPolyline()
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 5) As Double
    ' Define the 2D polyline points
    points(0) = 2: points(1) = 4
    points(2) = 4: points(3) = 2
    points(4) = 6: points(5) = 4
    ' Create a light weight Polyline object in model space
    Set plineObj = ThisDrawing.ModelSpace. _
                   AddLightWeightPolyline(points)
    ThisDrawing.Application.ZoomAll
End Sub
```

¿Comentarios?

Creación de objetos curvos

Con AutoCAD podrá crear una amplia variedad de objetos curvos, incluidos círculos, arcos, elipses y curvas spline. Todas las curvas se crean en el plano *XY* del SCU actual.

Para crear una curva, utilice uno de los métodos siguientes:

AddArc

Crea un arco contando con el centro, el radio y los ángulos inicial y final.

AddCircle

Crea un círculo con el radio y centro dados.

AddEllipse

Crea una elipse contando con el punto central, un punto en el eje mayor y la proporción del radio.

AddSpline

Crea una curva NURBS (B-spline racional no uniforme) cuadrática o cúbica.

Creación de un objeto Spline

En este ejemplo se crea una curva spline en espacio modelo a partir de tres puntos (0, 0, 0), (5, 5, 0) y (10, 0, 0). La curva tiene las tangentes inicial y final de (0,5, 0,5, 0,0).

```
Sub Ch4_CreateSpline()
    ' This example creates a spline object in model space.
    ' Declare the variables needed
    Dim splineObj As AcadSpline
    Dim startTan(0 To 2) As Double
    Dim endTan(0 To 2) As Double
```

```
    Dim fitPoints(0 To 8) As Double
    ' Define the variables
    startTan(0) = 0.5: startTan(1) = 0.5: startTan(2) = 0
    endTan(0) = 0.5: endTan(1) = 0.5: endTan(2) = 0
    fitPoints(0) = 1: fitPoints(1) = 1: fitPoints(2) = 0
    fitPoints(3) = 5: fitPoints(4) = 5: fitPoints(5) = 0
    fitPoints(6) = 10: fitPoints(7) = 0: fitPoints(8) = 0
    ' Create the spline
    Set splineObj = ThisDrawing.ModelSpace.AddSpline _
                        (fitPoints, startTan, endTan)
    ZoomAll
End Sub
```

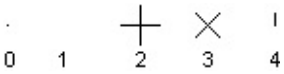
Para obtener más información acerca de las curvas spline, véase la documentación del objeto Spline y el método AddSpline en *ActiveX and VBA Reference* de AutoCAD.

¿Comentarios?

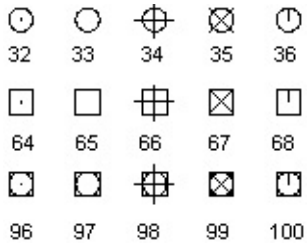
Creación de objetos Point

Los objetos de punto pueden ser de mucha utilidad, por ejemplo, como puntos de referencia o de nodo hacia los cuales podrá forzar el cursor o desfasar los objetos. Si lo desea, podrá especificar el estilo del punto, así como su tamaño, en relación con la pantalla o en unidades absolutas.

Las variables de sistema PDMODE y PDSIZE controlan el aspecto de los objetos de punto. Los valores 0, 2, 3 y 4 de PDMODE seleccionan una figura que debe dibujarse a través del punto. El valor 1 establece que no se visualice nada.



Añada 32, 64 o 96 al valor anterior para seleccionar una forma que debe dibujarse alrededor del punto además de la que se dibuja para atravesarlo:



PDSIZE controla el tamaño de las figuras de punto, salvo en los valores 0 y 1 de PDMODE. Al establecer PDSIZE en 0 se genera el punto al 5% de la altura del área gráfica. Un valor positivo de PDSIZE especifica un tamaño absoluto para las figuras de punto. Un valor negativo se interpreta como un porcentaje del tamaño de la ventana gráfica. El tamaño de todos los puntos vuelve a calcularse al regenerar el dibujo.

Después de cambiar PDMODE y PDSIZE, la próxima vez que se regenere el dibujo cambiará el aspecto de los puntos existentes.

Para definir PDMODE y PDSIZE, utilice el método SetVariable.

Creación de un objeto Point y modificación de su aspecto

El código siguiente crea un objeto Point en las coordenadas (5, 5, 0) del espacio modelo. Después se actualizan las variables de sistema PDMODE y PDSIZE.

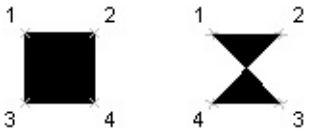
```
Sub Ch4_CreatePoint()
    Dim pointObj As AcadPoint
    Dim location(0 To 2) As Double
    ' Define the location of the point
    location(0) = 5#: location(1) = 5#: location(2) = 0#
    ' Create the point
    Set pointObj = ThisDrawing.ModelSpace.AddPoint(location)
    ThisDrawing.SetVariable "PDMODE", 34
    ThisDrawing.SetVariable "PDSIZE", 1
    ZoomAll
End Sub
```

<$nopage>relleno sólido (áreas):

Creación de áreas con relleno sólido

Es posible crear áreas triangulares y cuadriláteras rellenas de un color. Para obtener resultados más rápidos, estas áreas deben crearse con la variable de sistema FILLMODE desactivada, y activar de nuevo FILLMODE para rellenar el área terminada.

Cuando se crea un área de relleno sólido cuadrangular, la secuencia de los puntos tercero y cuarto determina su forma. Compare las figuras siguientes:



Los dos primeros puntos definen un lado del polígono. El tercer punto se define diagonalmente contrario al segundo. Si el cuarto punto se define igual que el tercero, se crea un triángulo relleno.

Para crear un área de relleno sólido, utilice el método AddSolid.

Para obtener más información acerca del relleno de sólidos, véase “Creación de áreas de relleno sólido” en el *Manual del usuario*.

Creación de un objeto con relleno sólido

El código del ejemplo siguiente crea un cuadrilátero sólido en las coordenadas (0, 0, 0), (5, 0, 0), (5, 8, 0) y (8, 8, 0) del espacio modelo.

```
Sub Ch4_CreateSolid()
    Dim solidObj As AcadSolid
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    Dim point3(0 To 2) As Double
```

```
    Dim point4(0 To 2) As Double
    ' Define the solid
    point1(0) = 0#: point1(1) = 0#: point1(2) = 0#
    point2(0) = 5#: point2(1) = 0#: point2(2) = 0#
    point3(0) = 5#: point3(1) = 8#: point3(2) = 0#
    point4(0) = 0#: point4(1) = 8#: point4(2) = 0#
    ' Create the solid object in model space
    Set solidObj = ThisDrawing.ModelSpace.AddSolid _
                   (point1, point2, point3, point4)
    ZoomAll
End Sub
```

¿Comentarios?

Trabajo con regiones

Las regiones son áreas cerradas de dos dimensiones que el usuario crea a partir de formas cerradas o bucles. Un bucle es una curva o una secuencia de curvas conectadas que define un área de un plano con un contorno que no intersecta consigo mismo. Los bucles pueden estar constituidos por una combinación de líneas, polilíneas optimizadas, círculos, arcos, elipses, arcos elípticos, splines, caras 3D, trazos y sólidos. Los objetos que conforman los bucles deben ser objetos cerrados o formar áreas cerradas cuyos puntos finales compartan con otros objetos. Los objetos han de ser además coplanarios (situados en el mismo plano). Los bucles que conforman una región han de estar definidos como matriz de objetos.

Para obtener más información acerca de las regiones, véase “Creación y combinación de áreas (regiones)” en el *Manual del usuario*.

- **Creación de regiones**
- **Creación de regiones compuestas**
- **Unión de regiones**
- **Cálculo de la intersección de dos regiones**

Creación de regiones

Para crear una región, utilice el método AddRegion Este método crea una región a partir de todos los bucles cerrados formados con la matriz de entrada de curvas. AutoCAD convierte las polilíneas 2D cerradas y las 3D planas en regiones distintas y, a continuación, convierte las polilíneas, líneas y curvas que forman bucles planos cerrados. Si más de dos curvas comparten un punto final, puede que la región resultante sea arbitraria. Por esta razón, es posible que algunas regiones en realidad se creen cuando se utilice el método AddRegion. Utilice una variante que contenga la recién creada matriz de regiones.

Puede calcular el total de objetos de región creados mediante las funciones `UBound` y `LBound` de VBA, como ilustra el siguiente ejemplo:

```
UBound(objRegions) - LBound(objRegions) + 1
```

donde `objRegions` es un variante que contiene el valor de retorno de AddRegion. Esta instrucción calcula el número total de regiones creadas.

Creación de una región simple

El código del ejemplo siguiente crea una región a partir de un círculo.

```
Sub Ch4_CreateRegion()
    ' Define an array to hold the
    ' boundaries of the region.
    Dim curves(0 To 0) As AcadCircle
    ' Create a circle to become a
    ' boundary for the region.
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2
    center(1) = 2
    center(2) = 0
```

```
    radius = 5#
    Set curves(0) = ThisDrawing.ModelSpace.AddCircle _
                    (center, radius)
    ' Create the region
    Dim regionObj As Variant
    regionObj = ThisDrawing.ModelSpace.AddRegion(curves)
    ZoomAll
End Sub
```

¿Comentarios?

Creación de regiones compuestas

Se pueden crear regiones compuestas mediante la sustracción, combinación o localización de la intersección de regiones o sólidos 3D. A continuación, se pueden extruir o girar las regiones compuestas para crear sólidos complejos. Para crear una región compuesta, utilice el método Boolean.

Cuando se sustrae una región de otra, se llama al método Boolean desde la región primera. Esta es la región de la que debe realizar la sustracción. Por ejemplo, si desea calcular los metros de alfombrado que necesita para un suelo, llame al método Boolean desde el contorno exterior del suelo y utilice las zonas que no irán cubiertas con moqueta, como es el caso del espacio que ocupan las columnas o los mostradores, como objeto de la lista de parámetros de Boolean.

Creación de una región compuesta

```
Sub Ch4_CreateCompositeRegions()
    ' Create two circles, one representing a room,
    ' the other a pillar in the center of the room
    Dim RoomObjects(0 To 1) As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 4
    center(1) = 4
    center(2) = 0
    radius = 2#
    Set RoomObjects(0) = ThisDrawing.ModelSpace. _
                         AddCircle(center, radius)
    radius = 1#
    Set RoomObjects(1) = ThisDrawing.ModelSpace. _
                         AddCircle(center, radius)
    ' Create a region from the two circles
    Dim regions As Variant
    regions = ThisDrawing.ModelSpace.AddRegion(RoomObjects)
    ' Copy the regions into the region variables for ease of use
```

```
    Dim RoundRoomObj As AcadRegion
    Dim PillarObj As AcadRegion
    If regions(0).Area > regions(1).Area Then
        ' The first region is the room
        Set RoundRoomObj = regions(0)
        Set PillarObj = regions(1)
    Else
        ' The first region is the pillar
        Set PillarObj = regions(0)
        Set RoundRoomObj = regions(1)
    End If
' Subtract the pillar space from the floor space to
    ' get a region that represents the total carpet area.
    RoundRoomObj.Boolean acSubtraction, PillarObj
    ' Use the Area property to determine the total carpet area
    MsgBox "The carpet area is: " & RoundRoomObj.Area
End Sub
```

Calcule el área de la región resultante con la propiedad Area.

Unión de regiones

Para unificar regiones, llame al método Boolean e introduzca en la operación la constante `acUnion` en lugar de `acSubtraction`. Puede combinar en cualquier orden las regiones que desee unir.

Cálculo de la intersección de dos regiones

Para localizar la intersección de dos regiones, se utiliza la constante `acIntersection`. Puede combinar en cualquier orden las regiones que desee intersectar.

Creación de sombreados

El sombreado rellena un área precisa de un dibujo con un patrón.

Al crear un sombreado, el área que debe rellenarse no se especifica al principio. Primero debe crear el objeto Hatch (de sombreado). Una vez hecho esto, puede especificar el bucle exterior, que es el contorno más externo del sombreado. Puede continuar especificando cualquier bucle interno que exista en el sombreado.

Para obtener más información acerca de los sombreados, véase “Introducción a los patrones de sombreado y los rellenos” en el *Manual del usuario*.

- **Creación de objetos Hatch**
- **Asociación de sombreados**
- **Asignación de nombre y tipo del patrón de sombreado**
- **Definición de contornos de sombreado**

Creación de objetos Hatch

Cuando cree un objeto Hatch, especifique el tipo del patrón de sombreado, el nombre del patrón de sombreado y la asociatividad. Una vez creado el objeto Hatch, no podrá cambiar su asociatividad.

Para crear un objeto Hatch, utilice el método AddHatch.

Asociación de sombreados

Se pueden crear sombreados asociativos y no asociativos. Los sombreados asociativos se vinculan a sus contornos y se actualizan al modificar éstos. Los sombreados no asociativos son independientes de sus contornos.

La asociatividad sólo puede establecerse al crear el sombreado. Una vez creado el sombreado, su asociatividad puede romperse, pero no puede volverse a asociar.

Para hacer que un sombreado sea asociativo, defina el parámetro `Associativity` del método AddHatch como `TRUE`. Para que no sea asociativo, defina el parámetro `Associativity` del método AddHatch como `FALSE`.

Asignación de nombre y tipo del patrón de sombreado

AutoCAD proporciona un relleno sólido y más de cincuenta patrones de sombreado estándar. Los patrones de sombreado resaltan una característica particular o un área determinada del dibujo. Los patrones pueden, por ejemplo, ayudar a diferenciar los componentes de un objeto 3D o representar los materiales de los que está hecho.

Puede aplicar un patrón suministrado por AutoCAD o tomar uno de una biblioteca externa de patrones. Para ver una tabla de los patrones de sombreado suministrados con AutoCAD, véase la *Lista de comandos* de AutoCAD

Para especificar un patrón concreto, al crear el objeto Hatch indique un tipo y un nombre de patrón. El tipo de patrón indica dónde localizar el nombre del patrón. Cuando introduzca el tipo de patrón, utilice una de las siguientes constantes:

acHatchPatternTypePredefined

Selecciona el nombre del patrón entre los definidos en el archivo *acad.pat*.

acHatchPatternTypeUserDefined

Define un patrón de líneas que utiliza el tipo de línea actual.

acHatchPatternTypeCustomDefined

Selecciona el nombre del patrón en un archivo PAT que no es *acad.pat*.

Cuando indique el nombre del patrón, utilice un nombre válido para el archivo especificado por el tipo de patrón.

Definición de contornos de sombreado

Una vez creado el objeto Hatch, puede añadirle los contornos. Un contorno puede ser cualquier combinación de líneas, arcos, círculos, polilíneas 2D, elipses, splines y regiones.

El primer contorno añadido debe ser el exterior, que define los límites externos que se rellenarán con el sombreado. Para agregar el contorno exterior, utilice el método AppendOuterLoop.

Una vez definido el contorno exterior, puede añadir los interiores. Agregue contornos interiores con el método AppendInnerLoop.

Los contornos interiores definen islas dentro del sombreado, que el objeto Hatch maneja según los parámetros de la propiedad HatchStyle. La propiedad HatchStyle puede definirse con una de las siguientes condiciones:

Definiciones de estilos de sombreado

HatchStyle	Condición	Descripción
	Normal	Especifica el estilo estándar o normal. Esta opción sombrea el interior desde el contorno del área más exterior. Si AutoCAD encuentra un contorno interno, desactiva el sombreado hasta que encuentra otro contorno. Este es el valor

		por defecto de la propiedad HatchStyle.
	Exterior	Rellena solamente las áreas más externas. Este estilo sombrea también el interior desde el contorno del área, pero desactiva el sombreado si encuentra un contorno interno y no lo vuelve a activar.
	Ignorar	Ignora la estructura interna. Esta opción sombrea todos los objetos internos.

Una vez terminada la definición del sombreado, debe evaluarse antes de que pueda verse. Utilice para ello el método Evaluate.

Creación de un objeto Hatch

Este ejemplo crea un sombreado asociado en espacio modelo. Una vez creado, puede cambiar el tamaño del círculo con el que está asociado el sombreado. El sombreado cambia para adaptarse al tamaño del círculo actual.

```
Sub Ch4_CreateHatch()
    Dim hatchObj As AcadHatch
    Dim patternName As String
    Dim PatternType As Long
    Dim bAssociativity As Boolean
    ' Define the hatch
    patternName = "ANSI31"
    PatternType = 0
    bAssociativity = True
    ' Create the associative Hatch object
    Set hatchObj = ThisDrawing.ModelSpace.AddHatch _
                   (PatternType, patternName, bAssociativity)
    ' Create the outer boundary for the hatch. (a circle)
```

```
    Dim outerLoop(0 To 0) As AcadEntity
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 3: center(1) = 3: center(2) = 0
    radius = 1
    Set outerLoop(0) = ThisDrawing.ModelSpace. _
                       AddCircle(center, radius)
    ' Append the outerboundary to the hatch
    ' object, and display the hatch
    hatchObj.AppendOuterLoop (outerLoop)
    hatchObj.Evaluate
    ThisDrawing.Regen True
End Sub
```

¿Comentarios?

<$startrange>conjuntos de selección:filtros (listas), <$startrange>filtros (listas), <$startrange>filtrar:conjuntos de selección, DXF (códigos), y tipos de filtros (tabla), filtros (tipos), y códigos DXF (tabla), <$startrange>filtros (listas):código de ejemplo, SelectionSet (objeto):código de ejemplo, filtrar:código de ejemplo, <$endrange>conjuntos de selección:filtros (listas), <$endrange>filtros (listas), <$endrange>filtrar:conjuntos de selección, <$endrange>filtros (listas):código de ejemplo, conjuntos de selección:eliminar objetos, objetos:eliminar de conjuntos de selección, RemoveItems (método):en conjuntos de selección, Clear (método):en conjuntos de selección, Erase (método):en conjuntos de selección, Delete (método):en conjuntos de selección,">

Trabajo con conjuntos de selección

Un conjunto de selección puede ser un único objeto, o puede contener un agrupamiento complejo: por ejemplo, el conjunto de objetos de una capa determinada.

El proceso de definición de un conjunto de selección consta de dos pasos. En primer lugar, debe crear un conjunto de selección nuevo y añadirlo a la colección SelectionSets. A continuación, debe llenar el conjunto de selección con los objetos que desee procesar

- **Creación de conjuntos de selección**
- **Adición de objetos a conjuntos de selección**
- **Definición de normas para conjuntos de selección**
- **Presentación de información sobre un conjunto de selección**
- **Eliminación de objetos de un conjunto de selección**

Creación de conjuntos de selección

Para crear un conjunto de selección con nombre, utilice el método Add. Este método requiere la entrada de un solo parámetro: el nombre del conjunto de selección.

Si ya existe un conjunto de selección del mismo nombre, AutoCAD presenta un mensaje de error. Es una buena práctica de programación borrar los conjuntos de selección en cuanto dejen de ser necesarios. Utilice el método Delete, como en el siguiente ejemplo, para borrarlos:

```
ThisDrawing.SelectionSets.Item("NewSelectionSet").Delete
```

Creación de un conjunto de selección vacío

Este ejemplo crea un conjunto de selección nuevo.

```
Sub Ch4_CreateSelectionSet()
    Dim selectionSet1 As AcadSelectionSet
    Set selectionSet1 = ThisDrawing.SelectionSets. _
                        Add("NewSelectionSet")
End Sub
```

Adición de objetos a conjuntos de selección

Se pueden añadir objetos al conjunto de selección activo mediante uno de los siguientes métodos:

AddItems

Añade uno o más objetos al conjunto de selección indicado.

Select

Selecciona objetos y los coloca en el conjunto de selección activo. Se pueden seleccionar todos los objetos, los objetos incluidos o que atraviesan un área rectangular o poligonal, todos los objetos que atraviesan un borde de selección, el objeto creado en último lugar, los objetos del conjunto de selección más reciente, los objetos de una ventana o los que están incluidos en un polígono de ventana.

SelectAtPoint

Selecciona los objetos que pasan más allá de un punto determinado y los coloca en el conjunto de selección activo.

SelectByPolygon

Selecciona los objetos que encierra un borde y los incluye en el conjunto de selección activo.

SelectOnScreen

Solicita al usuario que designe objetos de la pantalla y los añada al conjunto de selección activo.

Adición de objetos designados a un conjunto de selección

Este ejemplo solicita al usuario que seleccione objetos, para después añadirlos al

conjunto de selección.

```
Sub Ch4_AddToASelectionSet()
    ' Create a new selection set
    Dim sset As AcadSelectionSet
    Set sset = ThisDrawing.SelectionSets.Add("SS1")
    ' Prompt the user to select objects
    ' and add them to the selection set.
    ' To finish selecting, press INTRO.
    sset.SelectOnScreen
End Sub
```

¿Comentarios?

<$startrange>conjuntos de selección:filtros (listas), <$startrange>filtros (listas), <$startrange>filtrar:conjuntos de selección, DXF (códigos), y tipos de filtros (tabla), filtros (tipos), y códigos DXF (tabla), <$startrange>filtros (listas):código de ejemplo, SelectionSet (objeto):código de ejemplo, filtrar:código de ejemplo, <$endrange>conjuntos de selección:filtros (listas), <$endrange>filtros (listas), <$endrange>filtrar:conjuntos de selección, <$endrange>filtros (listas):código de ejemplo,">

Definición de normas para conjuntos de selección

Puede limitar los conjuntos de selección por propiedad o por tipo de objeto utilizando listas de filtros. Por ejemplo, puede copiar solamente los objetos azules en un dibujo de una placa de circuitos, o solamente los objetos de una capa determinada. También puede combinar los criterios de selección de una lista de filtros. Por ejemplo, puede indicar a AutoCAD que incluya un objeto en una selección solo si se trata de un círculo azul en una capa específica. Se pueden especificar listas de filtros para los métodos Select, SelectAtPoint, SelectByPolygon, y SelectOnScreen.

Nota El proceso de filtrado identifica tan sólo los tipos de línea que hayan sido asignados explícitamente a objetos, no aquéllos que hayan sido heredados por la capa.

- **Uso de listas de filtros para definir normas de conjuntos de selección**
- **Especificación de varios criterios en una lista de filtros de un conjunto de selección**
- **Adición de condiciones a la lista de filtros**
- **Uso de patrones comodín en los criterios de filtros de conjuntos de selección**
- **Filtro para datos extendidos**

¿Comentarios?

<$startrange>filtros (listas):código de ejemplo, SelectionSet (objeto):código de ejemplo, filtrar:código de ejemplo,">

Uso de listas de filtros para definir normas de conjuntos de selección

Las listas de filtros se componen de parejas de argumentos. El primer argumento representa el tipo de filtro (un objeto, por ejemplo), y el segundo argumento especifica el valor que se está filtrando (círculos, por ejemplo). El tipo de filtro consiste en un código de grupo DXF que indica el tipo de filtro que se aplicará. A continuación se presentan los tipos de filtro que más se utilizan.

Códigos DXF para filtros comunes

Código DXF	Tipo de filtro
0	Tipo de objeto (cadena) Como “Line,” “Circle,” “Arc” etc.
2	Nombre de objeto (cadena) El nombre de tabla (asignado) de un objeto guardado.
8	Nombre de capa (cadena) Como “Layer 0.”.
60	Visibilidad del objeto (entero)

	Use 0 = visible, 1 = invisible.
62	Número de color (entero) Valores de índice numéricos desde el 0 al 256. Cero indica BAYBLOCK. 256 indica BALAYER. Un valor negativo indica que la capa está desactivada.
67	Indicador de espacio modelo/papel (entero) 0 u omisión = espacio modelo, 1 = espacio papel.

Para obtener una lista completa de los códigos de grupo DXF, véase Tipos de valores de códigos de grupo en el *DXF Reference*.

Los argumentos de los filtros se declaran como matrices. El tipo de filtro se declara como entero y el valor de filtro es una variante. Cada tipo de filtro debe tener emparejado un valor de filtro. Por ejemplo:

```
FilterType(0) = 0 'Indicates filter refers to an object type
FilterData(0) = "Circle" 'Indicates the object type is "Circle"
```

Especificación de un solo criterio de selección para un conjunto de selección

El siguiente código solicita al usuario que seleccione los objetos que debe incluir un conjunto de selección, pero sólo los añade si se trata de un objeto Circle:

```
Sub Ch4_FilterMtext()
   Dim sstext As AcadSelectionSet
   Dim FilterType(0) As Integer
   Dim FilterData(0) As Variant
   Set sstext = ThisDrawing.SelectionSets.Add("SS2")
   FilterType(0) = 0
   FilterData(0) = "Circle"
   sstext.SelectOnScreen FilterType, FilterData
End Sub
```

¿Comentarios?

Especificación de varios criterios en una lista de filtros de un conjunto de selección

Para especificar varios criterios de selección, declare una matriz que contenga elementos suficientes para representar cada criterio, y asigne un criterio a cada elemento.

Selección de objetos que cumplen tres criterios

El código siguiente especifica dos criterios: el objeto debe ser un círculo, y debe residir en la capa 0. El código declara FilterType y FilterData como matrices de dos elementos y asigna cada criterio a un elemento:

```
Sub Ch4_FilterBlueCircleOnLayer0()
   Dim sstext As AcadSelectionSet
   Dim FilterType(1) As Integer
   Dim FilterData(1) As Variant
   Set sstext = ThisDrawing.SelectionSets.Add("SS4")
   FilterType(0) = 0
   FilterData(0) = "Circle"
   FilterType(1) = 8
   FilterData(1) = "0"
   sstext.SelectOnScreen FilterType, FilterData
End Sub
```

Adición de condiciones a la lista de filtros

Cuando se especifican varios criterios de selección, AutoCAD supone que el objeto seleccionado debe cumplir todos ellos. Los criterios, no obstante, pueden calificarse de otras formas. En cuanto a los elementos numéricos, se pueden especificar operaciones relacionales (por ejemplo, que el radio de un círculo sea *mayor o igual que* 5.0). Para todos los elementos, se pueden especificar operaciones lógicas (por ejemplo, Text *o* Mtext).

Para indicar un operador relacional en la definición del filtro, utilice un código DXF de -4. Especifique el operador como cadena. En la siguiente tabla se muestran los operadores relacionales que pueden utilizarse.

Operadores relacionales para las listas de filtros de conjuntos de selección

Operador	Descripción
"*"	Comodín genérico (siempre verdadero)
"="	Igual a
"!="	Distinto de
"/="	Distinto de
"<>"	Distinto de

"<"	Menor que
"<="	Menor o igual que
">"	Mayor que
">="	Mayor o igual que
"&"	AND binario (solo grupos de enteros)
"&="	Igual a, con máscara binaria (solo grupos de enteros)

Los operadores lógicos de las listas de filtros también se indican con un código de grupo –4 y deben ser una cadena, pero tienen que estar emparejados. El operador de apertura va precedido de un símbolo de menor que (<), y el de cierre de un símbolo de mayor que (>). En la siguiente tabla se muestran los operadores lógicos que pueden utilizarse.

Operadores de agrupamiento lógicos para las listas de filtros de conjuntos de selección

Operador inicial operador	Encierra	Ending operador
"<AND"	Uno o más operandos	"AND>"
"<OR"	Uno o más operandos	"OR>"
"<XOR"	Dos operandos	"XOR>"

"<NOT" Un operando "NOT>"

Selección de un círculo cuyo radio sea mayor o igual que 5.0.

El siguiente código indica que el objeto seleccionado debe ser un círculo cuyo radio sea mayor o igual que 5.0:

```
Sub Ch4_FilterRelational()
   Dim sstext As AcadSelectionSet
   Dim FilterType(2) As Integer
   Dim FilterData(2) As Variant
   Set sstext = ThisDrawing.SelectionSets.Add("SS5")
   FilterType(0) = 0
   FilterData(0) = "Circle"
   FilterType(1) = -4
   FilterData(1) = ">="
   FilterType(2) = 40
   FilterData(2) = 5#
   sstext.SelectOnScreen FilterType, FilterData
End Sub
```

Selección de Text o Mtext

En el siguiente ejemplo se especifica que se puedan seleccionar objetosText o Mtext:

```
Sub Ch4_FilterOrTest()
   Dim sstext As AcadSelectionSet
   Dim FilterType(3) As Integer
   Dim FilterData(3) As Variant
   Set sstext = ThisDrawing.SelectionSets.Add("SS6")
   FilterType(0) = -4
   FilterData(0) = "<or"
   FilterType(1) = 0
   FilterData(1) = "TEXT"
   FilterType(2) = 0
   FilterData(2) = "MTEXT"
   FilterType(3) = -4
   FilterData(3) = "or>"
   sstext.SelectOnScreen FilterType, FilterData
End Sub
```

Uso de patrones comodín en los criterios de filtros de conjuntos de selección

Los nombres de símbolos y cadenas de las listas de filtros pueden incluir patrones comodín.

En la siguiente tabla se incluyen los caracteres comodín que reconoce AutoCAD y su función en el contexto de una cadena:

Caracteres comodín

Carácter	Descripción
# (almohadilla)	Equivale a cualquier número de una sola cifra.
@ (arroba)	Equivale a cualquier carácter alfabético (solo uno).
. (punto)	Equivale a cualquier carácter no alfanumérico.
* (asterisco)	Equivale a cualquier secuencia de caracteres, incluidas las secuencias vacías, y puede utilizarse en cualquier lugar del patrón de búsqueda: al principio, medio o final

? (signo de interrogación)	Equivale a cualquier carácter (solo uno).
~ (tilde)	Cuando es el primer carácter del patrón, equivale a cualquier carácter distinto del patrón.
[...]	Equivale a cualquiera de los caracteres incluidos.
[~...]	Equivale a cualquier carácter no incluido (uno solo).
- (guión)	Se utiliza entre paréntesis para especificar un rango de un solo carácter.
, (coma)	Separa dos patrones.
` (comilla inversa)	Omite los caracteres especiales (lee el carácter siguiente).

Utilice la comilla simple (`) para indicar que un carácter no se debe tratar como comodín, sino como carácter literal. Por ejemplo, para establecer que el conjunto de selección solo incluya un bloque anónimo denominado “*U2”, utilice los siguientes argumentos de filtro:

```
FilterType(0) = 2
FilterData(0) = "`*U2"
```

Selección de Mtext donde aparezca una palabra específica en el texto

El siguiente código define los criterios de selección como cualquier Mtext cuya cadena de texto contenga la palabra “La”. En este ejemplo se muestra también el uso del método de selección SelectByPolygon:

```
Sub Ch4_FilterPolygonWildcard()
   Dim sstext As AcadSelectionSet
```

```
   Dim FilterType(1) As Integer
   Dim FilterData(1) As Variant
   Dim pointsArray(0 To 11) As Double
   Dim mode As Integer
   mode = acSelectionSetWindowPolygon
   pointsArray(0) = -12#: pointsArray(1) = -7#: pointsArray(2) = 0
   pointsArray(3) = -12#: pointsArray(4) = 10#: pointsArray(5) = 0
   pointsArray(6) = 10#: pointsArray(7) = 10#: pointsArray(8) = 0
   pointsArray(9) = 10#: pointsArray(10) = -7#: pointsArray(11) = 0
   Set sstext = ThisDrawing.SelectionSets.Add("SS10")
   FilterType(0) = 0
   FilterData(0) = "MTEXT"
   FilterType(1) = 1
   FilterData(1) = "*La*"
   sstext.SelectByPolygon mode, pointsArray, FilterType, FilterData
End Sub
```

¿Comentarios?

<$endrange>conjuntos de selección:filtros (listas), <$endrange>filtros (listas), <$endrange>filtrar:conjuntos de selección, <$endrange>filtros (listas):código de ejemplo,">

Filtro para datos extendidos

Se pueden utilizar aplicaciones externas para asociar datos tales como cadenas de texto, valores numéricos, puntos 3D, distancias y nombres de capa a objetos de AutoCAD. Estos son los denominados datos extendidos o datoseX. Se pueden crear filtros de entidades que contengan datos extendidos de una aplicación específica.

Para obtener más información acerca de los datos extendidos, véase Filtro para datos extendidos.

Selección de círculos que contienen datos extendidos

En el siguiente ejemplo, se filtran los círculos que contienen datos extendidos procedentes de la aplicación “MI_APL”.

```
Sub Ch4_FilterXdata()
   Dim sstext As AcadSelectionSet
   Dim mode As Integer
   Dim pointsArray(0 To 11) As Double
   mode = acSelectionSetWindowPolygon
   pointsArray(0) = -12#: pointsArray(1) = -7#: pointsArray(2) = 0
   pointsArray(3) = -12#: pointsArray(4) = 10#: pointsArray(5) = 0
   pointsArray(6) = 10#: pointsArray(7) = 10#: pointsArray(8) = 0
   pointsArray(9) = 10#: pointsArray(10) = -7#: pointsArray(11) = 0
   Dim FilterType(1) As Integer
   Dim FilterData(1) As Variant
   Set sstext = ThisDrawing.SelectionSets.Add("SS9")
   FilterType(0) = 0
   FilterData(0) = "Circle"
   FilterType(1) = 1001
```

```
   FilterData(1) = "MI_APL"
   sstext.SelectByPolygon mode, pointsArray, FilterType, FilterData
End Sub
```

¿Comentarios?

Presentación de información sobre un conjunto de selección

Si desea referirse a un conjunto de selección cuyo nombre conoce, utilice su nombre. En el siguiente ejemplo, se hace referencia a un conjunto de selección denominado “SS10”:

```
Sub GetObjInSet()
  Dim selset As AcadSelectionSet
  Set selset = ThisDrawing.SelectionSets("SS10")
  MsgBox ("Selection set " & selset.Name & " contains " & _
    selset.Count & " items")
End Sub
```

Los conjuntos de selección de un dibujo son miembros de la colección SelectionSets. Se puede utilizar la instrucción For Each para iterar en toda la colección SelectionSets de un dibujo y recoger información acerca de cada conjunto.

Presentación del nombre de los conjuntos de selección de un dibujo

El siguiente código muestra el nombre de todos los conjuntos de selección de un dibujo, junto con el tipo de los objetos incluidos en cada uno.

```
Sub ListSelectionSets()
  Dim selsetCollection As AcadSelectionSets
  Dim selset As AcadSelectionSet
  Dim ent As Object
  Dim i, j As Integer
  Set selsetCollection = ThisDrawing.SelectionSets
  ' Find each selection set in the drawing
  i = 0
  For Each selset In selsetCollection
    MsgBox "Selection set " & CStr(i) & " is: " & selset.Name
    ' Now find each object in the selection set, and say what it is
```

```
    j = 0
    For Each ent In selset
       MsgBox "Item " & CStr(j + 1) & " in " & selset.Name _
         & "is: " & ent.EntityName
       j = j + 1
    Next
    i = i + 1
  Next
End Sub
```

¿Comentarios?

Eliminación de objetos de un conjunto de selección

Una vez creado un conjunto de selección, puede optar por eliminar algunos de los objetos del conjunto o todos los que lo componen. Por ejemplo, puede seleccionar un grupo completo de objetos torpemente agrupados y eliminar algunos objetos concretos del grupo, dejando sólo aquéllos que desea que formen parte del conjunto.

Para eliminar elementos del conjunto de selección, utilice los siguientes métodos:

RemoveItems

El método RemoveItems elimina uno o más elementos del conjunto de selección. Los elementos eliminados siguen existiendo, aunque ya no residan en el conjunto de selección.

Clear

El método Clear vacía el conjunto de selección. El conjunto de selección sigue existiendo, pero sin elementos. Los elementos que residían en el conjunto de selección siguen existiendo, pero ya no residen en él.

Erase

El método Erase borra todos los elementos de un conjunto de selección. El conjunto de selección sigue existiendo, pero sin elementos. Los elementos que residían en el conjunto de selección ya no existen.

Delete

El método Delete suprime un conjunto de selección y todos los elementos que lo componen. Tras una llamada al método Delete, tanto el conjunto de selección como los elementos que incluía dejan de existir.

¿Comentarios?

<$nopage>matriz:

Modificación de objetos

Para modificar un objeto existente, puede utilizar los métodos y las propiedades que tiene asociados. Si modifica una propiedad visible de un objeto gráfico, utilice el método Update para volver a dibujarlo en la pantalla.

En esta sección se describe cómo editar objetos 2D.

Objetos guardados

Además de los objetos gráficos que utiliza AutoCAD, en los archivos de dibujo se almacenan también otros tipos de objetos no gráficos. Estos objetos tienen designaciones descriptivas asociadas, como son los bloques, capas, grupos y estilos de cotas. En la mayoría de los casos, a los objetos se les asigna un nombre cuando se crean, y se les cambia más adelante. Los nombres se almacenan en tablas de símbolos. Cuando se especifica un objeto guardado, se hace referencia al nombre y a los datos asociados del objeto que figura en la tabla de símbolos.

- **Limpieza de objetos guardados**
- **Cambio de nombre de los objetos**

Limpieza de objetos guardados

Puede limpiar el dibujo de objetos no utilizados ni referenciados en cualquier momento de una sesión de edición. La limpieza reduce el tamaño del dibujo. Ahora bien, no es posible limpiar los objetos a los que se hace referencia desde otros objetos. Por ejemplo, un archivo de tipos de letra puede ser referenciado por un estilo de texto. A las capas hacen referencia sus propios objetos.

Para limpiar un dibujo, utilice el método PurgeAll, como se indica a continuación:

```
ThisDrawing.PurgeAll
```

Cambio de nombre de los objetos

A medida que los dibujos se hacen más complicados, es aconsejable cambiar el nombre de los objetos para facilitar su identificación y evitar conflictos con los nombres utilizados en dibujos insertados.

Se puede cambiar el nombre de cualquier objeto guardado excepto de aquellos que AutoCAD denomina por defecto, como la capa 0 o el tipo de línea CONTINUOUS.

Los nombres pueden tener una longitud de hasta 255 caracteres. Además de letras y números, pueden contener espacios (aunque AutoCAD elimina los espacios al principio y al final de los nombres) y cualquier carácter especial que no se utilice en Microsoft Windows o AutoCAD para otros fines. Entre los caracteres especiales que no pueden utilizarse se incluyen los símbolos de mayor y menor que (< >), barras oblicuas y contrabarras (/\), comillas ("), signos de dos puntos (:), signos de punto y coma (;), signos de interrogación (?), comas (,), asteriscos (*), barras verticales (|), signos de igual (=) y comillas simples ('). Tampoco pueden utilizarse caracteres especiales creados con tipos de letra Unicode.

Para cambiar de nombre un objeto, utilice la propiedad Name del objeto.

Cambio de nombre de capas

En este ejemplo se crea una capa denominada “NewLayer” y a continuación se cambia su nombre a “MyLayer”.

```
Sub Ch4_RenamingLayer()
    ' Create a layer
    Dim layerObj As AcadLayer
    Set layerObj = ThisDrawing.Layers.Add("NewLayer")
    ' Change the name of the layer
    layerObj.Name = "MyLayer"
```

```
End Sub
```

¿Comentarios?

Copia de objetos

Se puede copiar uno o varios objetos dentro del dibujo activo. Al desfasar objetos se crean otros nuevos a una distancia específica respecto a los objetos designados o a través de un punto específico. La función de simetría crea una imagen simétrica de los objetos en un eje de simetría definido. La disposición de los objetos copiados en forma de matriz crea conjuntos de copias en un patrón circular o rectangular.

Para obtener información acerca de la copia de objetos, véase “Copia, desfase y reflejo de objetos” en el *Manual del usuario*.

Nota Los métodos de copia no se pueden utilizar mientras se realiza una iteración en una colección. Una iteración abre el espacio de trabajo para una operación de sólo lectura a la vez que estos métodos intentan realizar una operación de lectura/escritura. Termine cualquier iteración de una colección antes de llamar a estos métodos.

- **Copia de objetos en la misma ubicación**
- **Copia de varios objetos**

Copia de objetos en la misma ubicación

Para copiar un objeto, utilice el método Copy del mismo objeto. Este método crea un objeto nuevo que será un duplicado del original. El objeto nuevo se sitúa en la misma posición que el original y es el resultado del método.

Copia de varios objetos

Para copiar varios objetos, utilice el método CopyObjects o bien cree una matriz de objetos para utilizarla con el método Copy. (para copiar los objetos de un conjunto de selección, itere en el conjunto de selección y guarde los objetos en una matriz). Itere en la matriz, copiando cada objeto por separado, y reúna en una segunda matriz los objetos recién creados.

Para copiar varios objetos en un dibujo distinto, utilice el método CopyObjects y ajuste el parámetro Owner en el espacio modelo del dibujo.

Copia de dos objetos circulares

En este ejemplo, se crean dos objetos de círculo y se utiliza el método CopyObjects para crear una copia de los mismos.

```
Sub Ch4_CopyCircleObjects()
    Dim DOC1 As AcadDocument
    Dim circleObj1 As AcadCircle
    Dim circleObj2 As AcadCircle
    Dim circleObj1Copy As AcadCircle
    Dim circleObj2Copy As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius1 As Double
    Dim radius2 As Double
    Dim radius1Copy As Double
    Dim radius2Copy As Double
    Dim objCollection(0 To 1) As Object
    Dim retObjects As Variant
    ' Define the Circle object
    centerPoint(0) = 0: centerPoint(1) = 0: centerPoint(2) = 0
    radius1 = 5#: radius2 = 7#
    radius1Copy = 1#: radius2Copy = 2#
       ' Create a new drawing
    Set DOC1 = ThisDrawing.Application.Documents.Add
    ' Add two circles to the drawing
```

```
    Set circleObj1 = DOC1.ModelSpace.AddCircle _
                     (centerPoint, radius1)
    Set circleObj2 = DOC1.ModelSpace.AddCircle _
                     (centerPoint, radius2)
    ZoomAll
    ' Put the objects to be copied into a form
    ' compatible with CopyObjects
    Set objCollection(0) = circleObj1
    Set objCollection(1) = circleObj2
    ' Copy object and get back a collection of
    ' the new objects (copies)
    retObjects = DOC1.CopyObjects(objCollection)
    ' Get newly created object and apply
    ' new properties to the copies
    Set circleObj1Copy = retObjects(0)
    Set circleObj2Copy = retObjects(1)
    circleObj1Copy.radius = radius1Copy
    circleObj1Copy.Color = acRed
    circleObj2Copy.radius = radius2Copy
    circleObj2Copy.Color = acRed
    ZoomAll
End Sub
```

Copia de objetos en un dibujo distinto

Este ejemplo crea objetos Circle y después utiliza el método CopyObjects para copiar los círculos en un dibujo nuevo.

```
Sub Ch4_Copy_to_New_Drawing()
    Dim DOC0 As AcadDocument
    Dim circleObj1 As AcadCircle, circleObj2 As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius1 As Double, radius2 As Double
    Dim radius1Copy As Double, radius2Copy As Double
    Dim objCollection(0 To 1) As Object
    Dim retObjects As Variant
    ' Define the Circle object
    centerPoint(0) = 0: centerPoint(1) = 0: centerPoint(2) = 0
    radius1 = 5#: radius2 = 7#
    radius1Copy = 1#: radius2Copy = 2#
    ' Add two circles to the current drawing
    Set circleObj1 = ThisDrawing.ModelSpace.AddCircle _
                     (centerPoint, radius1)
    Set circleObj2 = ThisDrawing.ModelSpace.AddCircle _
                     (centerPoint, radius2)
    ThisDrawing.Application.ZoomAll
    ' Save pointer to the current drawing
    Set DOC0 = ThisDrawing.Application.ActiveDocument
```

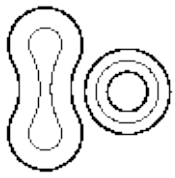
```
    ' Copy objects
    '
    ' First put the objects to be copied into a form compatible
    ' with CopyObjects
    Set objCollection(0) = circleObj1
    Set objCollection(1) = circleObj2
    ' Create a new drawing and point to its model space
    Dim Doc1MSpace As AcadModelSpace
    Dim DOC1 As AcadDocument
    Set DOC1 = Documents.Add
    Set Doc1MSpace = DOC1.ModelSpace
    ' Copy the objects into the model space of the new drawing. A
    ' collection of the new (copied) objects is returned.
    retObjects = DOC0.CopyObjects(objCollection, Doc1MSpace)
    Dim circleObj1Copy As AcadCircle, circleObj2Copy As AcadCircle
    ' Get the newly created object collection and apply new
    ' properties to the copies.
    Set circleObj1Copy = retObjects(0)
    Set circleObj2Copy = retObjects(1)
    circleObj1Copy.radius = radius1Copy
    circleObj1Copy.Color = acRed
    circleObj2Copy.radius = radius2Copy
    circleObj2Copy.Color = acRed
    ThisDrawing.Application.ZoomAll
    MsgBox "Circles copied."
End Sub
```

¿Comentarios?

Desfase de objetos

El desfase de un objeto crea uno nuevo a una distancia de desfase especificada con respecto al original. Se pueden desfasar arcos, círculos, elipses, líneas, polilíneas, polilíneas optimizadas, splines y líneas auxiliares.

Para desfasar un objeto, utilice el método Offset del objeto. La única entrada que requiere este método es la distancia a la que se desea desplazar el objeto. Si el valor de la distancia es negativo, AutoCAD supone que se pretende crear una curva “más pequeña”; un arco, por ejemplo, se desplazaría hasta un radio menor (que sería la distancia proporcionada) que el radio de la curva inicial. Si “más pequeña” no tiene sentido, AutoCAD aplica un desfase en la dirección de las coordenadas SCU *X, Y, Z* más pequeñas. Si la distancia de desfase no es válida, devuelve un error.



distancia entre objetos

En muchos objetos, esta operación tendrá como resultado una nueva curva sencilla (que puede no ser del mismo tipo que la curva original). Por ejemplo, al desfasar una elipse se obtiene una spline ya que el resultado se ajusta a la ecuación de una elipse. A veces puede ser necesario que el resultado del desfase sean varias curvas. Por esta razón, el método devuelve el nuevo objeto, o matriz de objetos, como variante.

Para obtener información acerca del desfase de objetos, véase “Copia, desfase y reflejo de objetos” en el *Manual del usuario*.

Desfase de una polilínea

Este ejemplo crea una polilínea optimizada y después la desfasa.

```
Sub Ch4_OffsetPolyline()
    ' Create the polyline
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 11) As Double
    points(0) = 1: points(1) = 1
    points(2) = 1: points(3) = 2
    points(4) = 2: points(5) = 2
    points(6) = 3: points(7) = 2
    points(8) = 4: points(9) = 4
    points(10) = 4: points(11) = 1
    Set plineObj = ThisDrawing.ModelSpace. _
                   AddLightWeightPolyline(points)
    plineObj.Closed = True
    ZoomAll
    ' Offset the polyline
    Dim offsetObj As Variant
    offsetObj = plineObj.Offset(0.25)
ZoomAll
End Sub
```

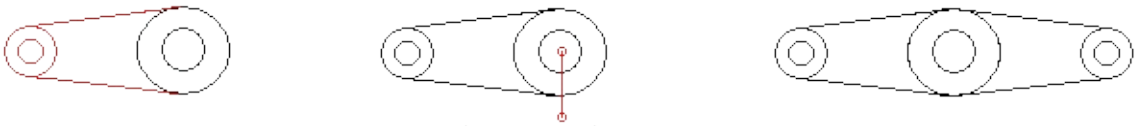
¿Comentarios?

Reflexión en simetría de objetos

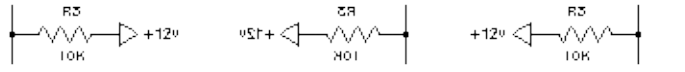
El reflejo de objetos crea una copia que es la imagen reflejada de un objeto con respecto a un eje o línea de simetría. Se pueden reflejar todos los objetos de dibujo.

Para reflejar un objeto, utilice el método Mirror. Este método requiere la entrada de dos coordenadas. Las dos coordenadas especificadas se convierten en puntos finales de la línea de simetría alrededor de la cual se refleja el objeto de base. En 3D, esta línea orienta un plano de simetría perpendicular al plano *XY* del SCP que contiene un eje de simetría especificado.

A diferencia del comando de simetría de AutoCAD, este método sitúa en el dibujo la imagen reflejada y mantiene el objeto original. Si desea eliminar el objeto original, utilice el método Erase.



Para controlar las propiedades de simetría de objetos de texto, utilice la variable de sistema MIRRTEXT. El valor por defecto de MIRRTEXT es activada (1), con el que la simetría de los objetos de texto se obtiene como la de los demás objetos. Cuando MIRRTEXT está desactivada (0), no se generan imágenes simétricas de texto. Utilice los métodos GetVariable y SetVariable para consultar y establecer el parámetro MIRRTEXT.



Puede obtener una imagen simétrica de un objeto de ventana gráfica en espacio

papel, aunque ello no afecta a la vista de los objetos en el espacio modelo ni a los objetos de dicho espacio.

Para obtener información acerca del reflejo de objetos, véase “Copia, desfase y reflejo de objetos” en el *Manual del usuario*.

Reflexión de una polilínea con respecto a un eje

Este ejemplo crea una polilínea optimizada y la refleja con respecto a un eje de simetría. La nueva polilínea es de color azul.

```
Sub Ch4_MirrorPolyline()
    ' Create the polyline
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 11) As Double
    points(0) = 1: points(1) = 1
    points(2) = 1: points(3) = 2
    points(4) = 2: points(5) = 2
    points(6) = 3: points(7) = 2
    points(8) = 4: points(9) = 4
    points(10) = 4: points(11) = 1
    Set plineObj = ThisDrawing.ModelSpace. _
               AddLightWeightPolyline(points)
    plineObj.Closed = True
    ZoomAll
    ' Define the mirror axis
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    point1(0) = 0: point1(1) = 4.25: point1(2) = 0
    point2(0) = 4: point2(1) = 4.25: point2(2) = 0
    ' Mirror the polyline
    Dim mirrorObj As AcadLWPolyline
    Set mirrorObj = plineObj.Mirror(point1, point2)
    Dim col As New AcadAcCmColor
    Call col.SetRGB(125, 175, 235)
    mirrorObj.TrueColor = col
    ZoomAll
End Sub
```

Disposición de objetos en matrices

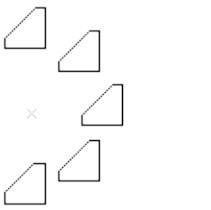
Con AutoCAD es posible copiar un objeto dispuesto en matrices polares o rectangulares. En el caso de las matrices polares, puede controlar el número de copias del objeto y el ángulo de relleno de la matriz. En cuanto a las matrices rectangulares, puede determinar el número de filas y columnas y la distancia que debe existir entre ellas.

Para obtener más información acerca de las matrices, “Creación de una matriz de objetos” en el *Manual del usuario*.

- **Creación de matrices polares**
- **Creación de matrices rectangulares**

Creación de matrices polares

Se pueden disponer en una matriz todos los objetos de dibujo. Para crear una matriz polar, utilice el método ArrayPolar del objeto. Este método requiere la entrada del número de objetos que se desean crear, los grados del ángulo y el punto central de la matriz. El número de objetos debe ser un entero positivo mayor que 1, y el ángulo a rellenar debe completarse en radianes. Un valor positivo precisa una rotación en sentido contrario a las agujas del reloj. Un valor negativo precisa una rotación en el sentido de las agujas del reloj. Un ángulo igual a 0 devuelve un error. El centro es una matriz de variantes que contiene tres dobles. Estos dobles representan las coordenadas 3D del SCU que indican el centro de la matriz polar



ángulo de matriz polar
para rellenar=180;
objetos no girados

AutoCAD determina la distancia entre el centro de la matriz y un punto de referencia del objeto original. El punto de referencia utilizado depende del tipo de objeto. AutoCAD utiliza el centro de un círculo o arco, el punto de inserción de un bloque o forma, el punto inicial del texto y un punto final de una línea o trazo.

Este método no admite la opción Girar objetos a medida que se copian del comando MATRIZ de AutoCAD.

Creación de una matriz polar

Este ejemplo crea un círculo y después crea una matriz polar del círculo. De esta forma se crean cuatro círculos que rellenan 180 grados alrededor de un punto base de (4, 4, 0).

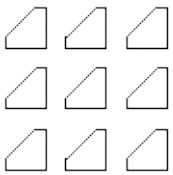
```
Sub Ch4_ArrayingACircle()
    ' Create the circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2#: center(1) = 2#: center(2) = 0#
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace. _
                    AddCircle(center, radius)
    ZoomAll
    ' Define the polar array
    Dim noOfObjects As Integer
    Dim angleToFill As Double
    Dim basePnt(0 To 2) As Double
    noOfObjects = 4
    angleToFill = 3.14 ' 180 degrees
    basePnt(0) = 4#: basePnt(1) = 4#: basePnt(2) = 0#
    ' The following example will create 4 copies
    ' of an object by rotating and copying it about
    ' the point (3,3,0).
    Dim retObj As Variant
    retObj = circleObj.ArrayPolar _
             (noOfObjects, angleToFill, basePnt)
    ZoomAll
End Sub
```

Creación de matrices rectangulares

Para crear una matriz rectangular 2D o 3D, utilice el método ArrayRectangular del objeto. Este método requiere que se especifique el número de filas y columnas, la distancia entre filas y la distancia entre columnas. Al crear una matriz 3D también hay que precisar el número de niveles y la distancia entre ellos.

Las matrices rectangulares se construyen reproduciendo el objeto del conjunto de selección el número de veces necesario. La definición de una fila implica la especificación de más de una columna, y viceversa.

Se supone que el objeto original se encuentra en la esquina inferior izquierda, por lo que la matriz se genera arriba y a la derecha. Cuando la distancia entre filas es un número negativo, las filas se añaden hacia abajo. Si la distancia entre columnas es un número negativo, las columnas se añaden hacia la izquierda.



AutoCAD construye la matriz rectangular a lo largo de la línea base definida por el ángulo de rotación de resolución actual. Por defecto, dicho ángulo viene definido con el valor 0, lo que supone que las filas y columnas de una matriz rectangular sean ortogonales con respecto a los ejes *X* e *Y* del dibujo. Puede cambiar este ángulo y crear una matriz rotada si define el ángulo de rotación de Forzcursor con un valor distinto de cero. Para ello, utilice la propiedad SnapRotationAngle.

Creación de una matriz rectangular

En este ejemplo se crea un círculo y, a continuación, una matriz rectangular del círculo con cinco filas y cinco columnas de círculos.

```
Sub Ch4_ArrayRectangularExample()
    ' Create the circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2#: center(1) = 2#: center(2) = 0#
    radius = 0.5
    Set circleObj = ThisDrawing.ModelSpace. _
                            AddCircle(center, radius)
    ZoomAll
    ' Define the rectangular array
    Dim numberOfRows As Long
    Dim numberOfColumns As Long
    Dim numberOfLevels As Long
    Dim distanceBwtnRows As Double
    Dim distanceBwtnColumns As Double
    Dim distanceBwtnLevels As Double
    numberOfRows = 5
    numberOfColumns = 5
    numberOfLevels = 2
    distanceBwtnRows = 1
    distanceBwtnColumns = 1
    distanceBwtnLevels = 1
    ' Create the array of objects
    Dim retObj As Variant
    retObj = circleObj.ArrayRectangular _
        (numberOfRows, numberOfColumns, numberOfLevels, _
        distanceBwtnRows, distanceBwtnColumns, distanceBwtnLevels)
    ZoomAll
End Sub
```

Desplazamiento de objetos

Se puede desplazar objetos a lo largo de un vector sin modificar su orientación ni su tamaño. También se puede girar objetos en torno a un punto base.

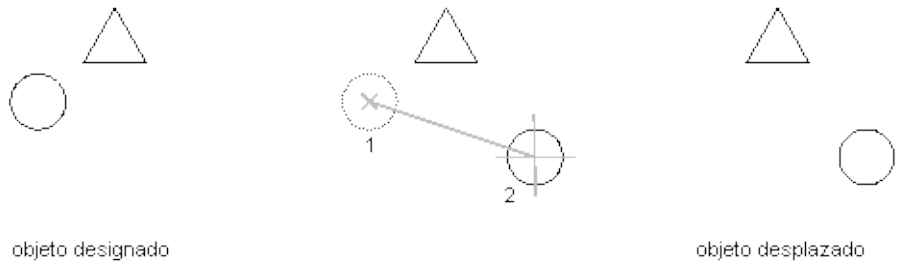
Para obtener más información acerca del desplazamiento de objetos, véase “Desplazamiento de objetos” en el *Manual del usuario*.

- **Desplazamiento de objetos a lo largo de un vector**

Desplazamiento de objetos a lo largo de un vector

Todos los objetos de dibujo y los objetos de referencia de atributos pueden desplazarse por un vector definido.

Para desplazar un objeto, utilice el método Move del objeto. Este método requiere la entrada de dos coordenadas. Estas coordenadas definen un vector de desplazamiento que indica la distancia a la que debe trasladarse el objeto y en qué dirección.



Desplazamiento de un círculo a lo largo de un vector

En este ejemplo se crea un círculo y, a continuación, se desplaza dos unidades a lo largo del eje *X*.

```
Sub Ch4_MoveCircle()
    ' Create the circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2#: center(1) = 2#: center(2) = 0#
    radius = 0.5
    Set circleObj = ThisDrawing.ModelSpace. _
                    AddCircle(center, radius)
    ZoomAll
    ' Define the points that make up the move vector.
    ' The move vector will move the circle 2 units
```

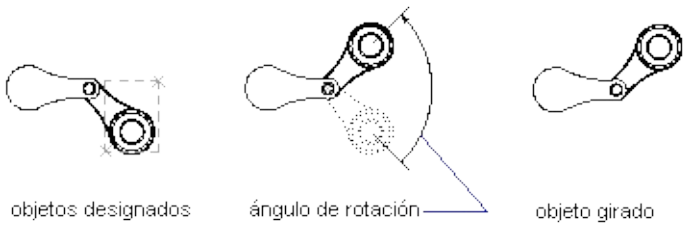
```
    ' along the x axis.
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    point1(0) = 0: point1(1) = 4.25: point1(2) = 0
    point2(0) = 2: point2(1) = 4.25: point2(2) = 0
    ' Move the circle
    circleObj.Move point1, point2
    circleObj.Update
End Sub
```

¿Comentarios?

Rotación de objetos

Puede rotar todos los objetos de dibujo y todos los objetos de referencia de atributos.

Para rotar un objeto, utilice el método Rotate del objeto. Este método requiere la entrada de un punto base y de un ángulo de rotación. El punto base es una matriz de variantes con tres dobles. Estos dobles representan las coordenadas 3D del SCU que indican el punto sobre el que está definido el eje de rotación. El ángulo de rotación se designa en radianes y determina cuánto rota un objeto alrededor del punto base respecto de su posición actual.



Para obtener más información acerca de la rotación de objetos, véase “Rotación de objetos” en el *Manual del usuario*.

Rotación de una polilínea con respecto a un punto base

Este ejemplo crea una polilínea optimizada cerrada y después la gira 45 grados con respecto al punto base (4, 4.25, 0).

```
Sub Ch4_RotatePolyline()
    ' Create the polyline
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 11) As Double
    points(0) = 1: points(1) = 2
    points(2) = 1: points(3) = 3
    points(4) = 2: points(5) = 3
```

```
    points(6) = 3: points(7) = 3
    points(8) = 4: points(9) = 4
    points(10) = 4: points(11) = 2
    Set plineObj = ThisDrawing.ModelSpace. _
                   AddLightWeightPolyline(points)
    plineObj.Closed = True
    ZoomAll
    ' Define the rotation of 45 degrees about a
    ' base point of (4, 4.25, 0)
    Dim basePoint(0 To 2) As Double
    Dim rotationAngle As Double
    basePoint(0) = 4: basePoint(1) = 4.25: basePoint(2) = 0
    rotationAngle = 0.7853981 ' 45 degrees
    ' Rotate the polyline
    plineObj.Rotate basePoint, rotationAngle
    plineObj.Update
End Sub
```

¿Comentarios?

Supresión de objetos

Los objetos individuales se suprimen con el método Delete.

Nota Los objetos Collection de ActiveX Automation tienen un método Delete debido al modo en que estos objetos están definidos en la biblioteca de tipos. No obstante, los objetos Collection como colecciones ModelSpace, Layers y Dictionaries no deben suprimirse nunca. Si se intenta suprimir una colección, se produce un error.

Creación y supresión de una polilínea

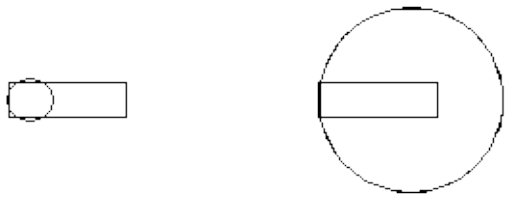
Este ejemplo crea una polilínea optimizada y después la borra.

```
Sub Ch4_DeletePolyline()
    ' Create the polyline
    Dim lwpolyObj As AcadLWPolyline
    Dim vertices(0 To 5) As Double
    vertices(0) = 2: vertices(1) = 4
    vertices(2) = 4: vertices(3) = 2
    vertices(4) = 6: vertices(5) = 4
    Set lwpolyObj = ThisDrawing.ModelSpace. _
                AddLightWeightPolyline(vertices)
    ZoomAll
' Erase the polyline
    lwpolyObj.Delete
    ThisDrawing.Regen acActiveViewport
End Sub
```

Aplicar una escala a los objetos

Se puede atribuir una escala a un objeto si se indican un punto base y una longitud, que se utilizará como factor de escala en función de las unidades de dibujo actuales. Puede ajustar la escala de todos los objetos de dibujo, así como la de todos los objetos de referencia de atributos.

Para ajustar el factor de escala de un objeto, utilice el método ScaleEntity del objeto. Este método ajusta la misma escala para el objeto en las direcciones *X*, *Y* y *Z*. Acepta como entrada el punto base de la escala y un factor de escala. El punto base es una matriz de variantes con tres dobles. Estos dobles representan las coordenadas 3D del SCU que indican el punto donde comienza la escala. El factor de escala es el valor sobre el que se ajusta la escala del objeto. Las cotas del objeto se multiplican por el factor de escala. Un factor de escala superior al valor 1 amplía el objeto. Un factor de escala entre 0 y 1 reduce el objeto.



Para obtener más información acerca de la aplicación de escala, véase “Ajuste del tamaño o la forma de los objetos” en el *Manual del usuario*.

Cambio de la escala de una polilínea

Este ejemplo crea una polilínea optimizada cerrada y después ajusta su escala con un factor 0.5.

```
Sub Ch4_ScalePolyline()
```

```
    ' Create the polyline
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 11) As Double
    points(0) = 1: points(1) = 2
    points(2) = 1: points(3) = 3
    points(4) = 2: points(5) = 3
    points(6) = 3: points(7) = 3
    points(8) = 4: points(9) = 4
    points(10) = 4: points(11) = 2
    Set plineObj = ThisDrawing.ModelSpace. _
                      AddLightWeightPolyline(points)
    plineObj.Closed = True
    ZoomAll
    ' Define the scale
    Dim basePoint(0 To 2) As Double
    Dim scalefactor As Double
    basePoint(0) = 4: basePoint(1) = 4.25: basePoint(2) = 0
    scalefactor = 0.5
    ' Scale the polyline
    plineObj.ScaleEntity basePoint, scalefactor
    plineObj.Update
End Sub
```

¿Comentarios?

<$nopage>matriz:

Transformación de objetos

Un objeto se puede desplazar, cambiar de escala o rotar con una matriz de transformación de 4 por 4 utilizando el método TransformBy.

En la tabla siguiente se muestra la configuración de la matriz de transformación, donde R = rotación y T = transformación.

Configuración de la matriz de transformación			
R00	R01	R02	T0
R10	R11	R12	T1
R20	R21	R22	T2
0	0	0	1

Para transformar un objeto, es necesario inicializar antes la matriz de transformación. En el siguiente ejemplo se muestra una matriz de transformación, asignada a la variable `tMatrix`, que rota una entidad 90 grados alrededor del punto (0, 0, 0):

```
tMatrix(0,0) = 0.0
tMatrix(0,1) = -1.0
tMatrix(0,2) = 0.0
tMatrix(0,3) = 0.0
tMatrix(1,0) = 1.0
tMatrix(1,1) = 0.0
tMatrix(1,2) = 0.0
```

```
tMatrix(1,3) = 0.0
tMatrix(2,0) = 0.0
tMatrix(2,1) = 0.0
tMatrix(2,2) = 1.0
tMatrix(2,3) = 0.0
tMatrix(3,0) = 0.0
tMatrix(3,1) = 0.0
tMatrix(3,2) = 0.0
tMatrix(3,3) = 1.0
```

Una vez terminada la matriz de transformación, debe aplicarse al objeto con el método TransformBy. La siguiente línea de código es una demostración de cómo se aplica una matriz (`tMatrix`) a un objeto (`anObj`):

```
anObj.TransformBy tMatrix
```

Rotación de una línea mediante una matriz de transformación

Este ejemplo crea una línea y la gira 90 grados aplicando una matriz de transformación.

```
Sub Ch4_TransformBy()
    ' Create a line
    Dim lineObj As AcadLine
    Dim startPt(0 To 2) As Double
    Dim endPt(0 To 2) As Double
    startPt(2) = 0
    startPt(1) = 1
    startPt(2) = 0
    endPt(0) = 5
    endPt(1) = 1
    endPt(2) = 0
    Set lineObj = ThisDrawing.ModelSpace. _
                AddLine(startPt, endPt)
    ZoomAll
    ' Initialize the transMat variable with a
    ' transformation matrix that will rotate
    ' an object by 90 degrees about the point(0,0,0)
    Dim transMat(0 To 3, 0 To 3) As Double
    transMat(0, 0) = 0#: transMat(0, 1) = -1#
    transMat(0, 2) = 0#: transMat(0, 3) = 0#
    transMat(1, 0) = 1#: transMat(1, 1) = 0#
    transMat(1, 2) = 0#: transMat(1, 3) = 0#
    transMat(2, 0) = 0#: transMat(2, 1) = 0#
    transMat(2, 2) = 1#: transMat(2, 3) = 0#
    transMat(3, 0) = 0#: transMat(3, 1) = 0#
```

```
    transMat(3, 2) = 0#: transMat(3, 3) = 1#
    ' Transform the line using the defined transformation matrix
    lineObj.TransformBy transMat
    lineObj.Update
End Sub
```

A continuación se presentan otros ejemplos de matrices de transformación:

Matriz de rotación: 90 grados alrededor del punto (0, 0, 0)

0.0	-1.0	0.0	0.0
1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1,0

Matriz de rotación: 45 grados alrededor del punto (5, 5, 0)

0.707107	-0.707107	0.0	5.0
0.707107	0.707107	0.0	-2.071068
0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0

Matriz de traslación: mueve una entidad en (10, 10, 0)

1.0	0.0	0.0	10.0
0.0	1.0	0.0	10.0

0.0	0.0	1.0	0.0
0.0	0.0	0.0	1.0

Matriz de ajuste de escala: ajuste de escala de 10, 10 en el punto (0, 0, 0)

10.0	0.0	0.0	0.0
0.0	10.0	0.0	0.0
0.0	0.0	10.0	0.0
0.0	0.0	0.0	1.0

Matriz de ajuste de escala: ajuste de escala de 10, 10 en el punto (2, 2, 0)

10.0	0.0	0.0	-18.0
0.0	10.0	0.0	-18.0
0.0	0.0	10.0	0.0
0.0	0.0	0.0	1.0

Alargamiento y recorte de objetos

Se puede cambiar el ángulo de los arcos y la longitud de las líneas abiertas, arcos, polilíneas abiertas, arcos elípticos y splines abiertas. Se obtiene un resultado muy parecido al del alargamiento y recorte de objetos.

Los objetos se pueden alargar y recortar si se modifican sus propiedades. Por ejemplo, para alargar una línea, cambie las coordenadas de las propiedades StartPoint o EndPoint. Para cambiar el ángulo de un arco, modifique las propiedades StartAngle o EndAngle del arco. Después de modificar propiedades de un objeto, debe utilizarse el método Update para ver los cambios en el dibujo.

Para obtener más información acerca del alargamiento y recorte de objetos, véase “Ajuste del tamaño o la forma de los objetos” en el *Manual del usuario*.

Alargar una línea

En este ejemplo se crea una línea y se cambia su punto final, con lo que aumenta su longitud.

```
Sub Ch4_LengthenLine()
    ' Define and create the line
    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double
    startPoint(0) = 0
    startPoint(1) = 0
    startPoint(2) = 0
    endPoint(0) = 1
    endPoint(1) = 1
    endPoint(2) = 1
    Set lineObj = ThisDrawing.ModelSpace. _
                  AddLine(startPoint, endPoint)
    lineObj.Update
    ' Lengthen the line by changing the
```

```
    ' endpoint to 4, 4, 4
    endPoint(0) = 4
    endPoint(1) = 4
    endPoint(2) = 4
    lineObj.endPoint = endPoint
    lineObj.Update
End Sub
```

¿Comentarios?

Descomposición de objetos

La descomposición de objetos fragmenta los objetos individuales en sus partes constitutivas, pero sus efectos no son visibles en la pantalla. Por ejemplo, la descomposición de formas de lugar a líneas y arcos a partir de polígonos 3D, polilíneas, mallas poligonales y regiones. Sustituye una referencia a bloque con copias de los objetos simples que componen el bloque.

Para obtener información acerca de la descomposición de objetos, véase “Disociación de objetos compuestos (Descomponer)” en el *Manual del usuario*.

Descomposición de una polilínea

Este ejemplo crea un objeto de polilínea optimizada. Después la descompone en varios objetos. El ejemplo realiza un bucle en los objetos resultantes y muestra un cuadro de mensaje que contiene el nombre de todos los objetos y su índice en la lista de objetos descompuestos.

```
Sub Ch4_ExplodePolyline()
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 11) As Double
    ' Define the 2D polyline points
    points(0) = 1: points(1) = 1
    points(2) = 1: points(3) = 2
    points(4) = 2: points(5) = 2
    points(6) = 3: points(7) = 2
    points(8) = 4: points(9) = 4
    points(10) = 4: points(11) = 1
    ' Create a light weight Polyline object
    Set plineObj = ThisDrawing.ModelSpace. _
                   AddLightWeightPolyline(points)
    ' Set the bulge on one segment to vary the
    ' type of objects in the polyline
    plineObj.SetBulge 3, -0.5
    plineObj.Update
```

```
    ' Explode the polyline
    Dim explodedObjects As Variant
    explodedObjects = plineObj.Explode
    ' Loop through the exploded objects
    ' and display a message box with
    ' the type of each object
    Dim I As Integer
    For I = 0 To UBound(explodedObjects)
    explodedObjects(I).Update
        MsgBox "Exploded Object " & I & ": " & _
                    explodedObjects(I).ObjectName
        explodedObjects(I).Update
    Next
End Sub
```

¿Comentarios?

Edición de polilíneas

Las polilíneas 2D y 3D, los rectángulos, los polígonos y las mallas poligonales 3D son variantes de polilíneas y se editan de la misma manera que ellas.

AutoCAD reconoce tanto las polilíneas ajustadas como las polilíneas ajustadas en forma de splines. Una polilínea ajustada en forma de spline utiliza un ajuste de curva, similar a una B-spline. Existen dos tipos de polilíneas ajustadas en forma de spline: cuadráticas y cúbicas. Las dos polilíneas están controladas por la variable de sistema SPLINETYPE. Una polilínea ajustada utiliza curvas estándar para el ajuste de curvas y cualquier dirección tangente definida en un vértice determinado.

Para modificar una polilínea, utilice las propiedades y los métodos de los objetos LightweightPolyline o Polyline. Para abrir o cerrar una polilínea, cambiar las coordenadas de un vértice de polilínea o agregar un vértice, utilice los siguientes métodos y propiedades:

Closed (propiedad)

Abre o cierra la polilínea.

Coordinates (propiedad)

Especifica las coordenadas de cada vértice de la polilínea.

AddVertex (método)

Añade un vértice a una polilínea optimizada.

Utilice los siguientes métodos para actualizar la curvatura o la anchura de una polilínea:

SetBulge

Define la curvatura de una polilínea, dado el índice de segmentos.

SetWidth

Define las anchuras inicial y final de una polilínea, dado el índice de segmentos.

Para obtener más información acerca de la modificación de polilíneas, véase “Modificación o unión de polilíneas” en el *Manual del usuario*.

Modificación de una polilínea

Este ejemplo crea una polilínea optimizada. Después añade una curvatura al tercer segmento de la polilínea, añade un vértice, cambia la anchura del último segmento y, por último, la cierra.

```
Sub Ch4_EditPolyline()
    Dim plineObj As AcadLWPolyline
    Dim points(0 To 9) As Double
    ' Define the 2D polyline points
    points(0) = 1: points(1) = 1
    points(2) = 1: points(3) = 2
    points(4) = 2: points(5) = 2
    points(6) = 3: points(7) = 2
    points(8) = 4: points(9) = 4
    ' Create a light weight Polyline object
    Set plineObj = ThisDrawing.ModelSpace. _
                   AddLightWeightPolyline(points)
    ' Add a bulge to segment 3
    plineObj.SetBulge 3, -0.5
    ' Define the new vertex
    Dim newVertex(0 To 1) As Double
    newVertex(0) = 4: newVertex(1) = 1
    ' Add the vertex to the polyline
    plineObj.AddVertex 5, newVertex
    ' Set the width of the new segment
    plineObj.SetWidth 4, 0.1, 0.5
    ' Close the polyline
    plineObj.Closed = True
    plineObj.Update
End Sub
```

Modificación de splines

Utilice las siguientes propiedades modificables para cambiar curvas spline:

ControlPoints

Especifica los puntos de apoyo de la spline.

EndTangent

Establece la tangente final de la spline como vector de dirección.

FitPoints

Especifica todos los puntos de ajuste de la spline.

FitTolerance

Vuelve a ajustar la curva Spline a los puntos existentes con los valores de tolerancia nuevos.

Knots

Especifica el vector nodal de la spline.

StartTangent

Especifica la tangente inicial de la spline.

También puede utilizar estos métodos para editar splines:

AddFitPoint

Agrega un punto de ajuste a la spline en el índice indicado.

DeleteFitPoint

Suprime el punto de ajuste de una spline en el índice indicado.

ElevateOrder

Eleva el orden de la spline hasta el orden indicado.

GetFitPoint

Define el punto de ajuste en el índice indicado (sólo un punto de ajuste. (Sólo un punto de ajuste. Para consultar todos los puntos de ajuste de la spline, utilice la propiedad FitPoints).

Invertir

Invierte la dirección de la spline.

SetControlPoint

Define el punto de apoyo de la spline en el índice indicado.

SetFitPoint

Define el punto de ajuste en el índice indicado. (Sólo un punto de ajuste. Para consultar todos los puntos de ajuste de la spline, utilice la propiedad FitPoints).

SetWeight

Define el grosor del punto de apoyo en un índice dado.

Utilice las siguientes propiedades de sólo lectura para consultar splines:

Area

Obtiene el área cerrada de una spline.

Closed

Indica si la spline está abierta o cerrada.

Degree

Obtiene el grado de la representación polinómica de la spline.

IsPeriodic

Especifica si la spline dada es periódica.

IsPlanar

Especifica si la spline dada es plana.

IsRational

Especifica si la spline dada es racional.

NumberOfControlPoints

Obtiene el número de puntos de apoyo de la spline.

NumberOfFitPoints

Obtiene el número de puntos de ajuste de la spline.

Para obtener más información acerca de la modificación de curvas spline, véase “Modificación de splines” en el *Manual del usuario*.

Modificación de un punto de apoyo en una curva spline

Este ejemplo crea una curva spline y cambia su primer punto de apoyo.

```
Sub Ch4_ChangeSplineControlPoint()
    ' Create the spline
    Dim splineObj As AcadSpline
    Dim startTan(0 To 2) As Double
    Dim endTan(0 To 2) As Double
    Dim fitPoints(0 To 8) As Double
    startTan(0) = 0.5: startTan(1) = 0.5: startTan(2) = 0
    endTan(0) = 0.5: endTan(1) = 0.5: endTan(2) = 0
    fitPoints(0) = 1: fitPoints(1) = 1: fitPoints(2) = 0
    fitPoints(3) = 5: fitPoints(4) = 5: fitPoints(5) = 0
    fitPoints(6) = 10: fitPoints(7) = 0: fitPoints(8) = 0
    Set splineObj = ThisDrawing.ModelSpace. _
                    AddSpline(fitPoints, startTan, endTan)
    splineObj.Update
    ' Change the coordinate of the first fit point
    Dim controlPoint(0 To 2) As Double
    controlPoint(0) = 0
    controlPoint(1) = 3
    controlPoint(2) = 0
    splineObj.SetControlPoint 0, controlPoint
    splineObj.Update
End Sub
```

Edición de sombreados

Se pueden editar tanto los contornos como los patrones de sombreado. Si se modifica el contorno de un sombreado asociativo, el patrón se actualizará siempre y cuando el proceso de edición dé como resultado un contorno válido. Los sombreados asociativos se actualizan incluso si figuran en capas inutilizadas. AutoCAD permite modificar los patrones de sombreado o elegir patrones nuevos para los sombreados existentes, pero la asociatividad sólo puede definirse una vez que el objeto esté creado. Puede utilizar la propiedad AssociativeHatch para verificar si un objeto Hatch está asociado. (para obtener más información acerca de la creación de un sombreado, véase el método) AddHatch.

Para ver los cambios de sombreado efectuados, se debe volver a evaluar el sombreado con el método Evaluate.

Para obtener más información sobre la modificación de sombreados, véase “Modificación de sombreados y áreas de relleno sólido” en el *Manual del usuario*.

- **Modificación de contornos de sombreado**
- **Edición de patrones de sombreado**

Modificación de contornos de sombreado

Se pueden agregar o insertar bucles en los contornos del sombreado. Los sombreados asociativos se actualizan para igualarse a los cambios efectuados en sus contornos. Los sombreados no asociativos no se actualizan.

Para editar un contorno de sombreado, utilice uno de los métodos siguientes:

AppendInnerLoop

Añade un bucle interior al sombreado.

AppendOuterLoop

Añade un bucle exterior al sombreado.

InsertLoopAt

Inserta un bucle en un índice dado de un sombreado.

Adición de un bucle interior a un sombreado.

Este ejemplo crea un sombreado asociativo. Después crea un círculo y lo añade como bucle interno del sombreado.

```
Sub Ch4_AppendInnerLoopToHatch()
    Dim hatchObj As AcadHatch
    Dim patternName As String
    Dim PatternType As Long
    Dim bAssociativity As Boolean
    ' Define and create the hatch
    patternName = "ANSI31"
    PatternType = 0
    bAssociativity = True
    Set hatchObj = ThisDrawing.ModelSpace. _
        AddHatch(PatternType, patternName, bAssociativity)
    ' Create the outer loop for the hatch.
```

```
    Dim outerLoop(0 To 1) As AcadEntity
    Dim center(0 To 2) As Double
    Dim radius As Double
    Dim startAngle As Double
    Dim endAngle As Double
    center(0) = 5: center(1) = 3: center(2) = 0
    radius = 3
    startAngle = 0
    endAngle = 3.141592
    Set outerLoop(0) = ThisDrawing.ModelSpace. _
       AddArc(center, radius, startAngle, endAngle)
    Set outerLoop(1) = ThisDrawing.ModelSpace. _
       AddLine(outerLoop(0).startPoint, outerLoop(0).endPoint)
    ' Append the outer loop to the hatch object
    hatchObj.AppendOuterLoop (outerLoop)
    ' Create a circle as the inner loop for the hatch.
    Dim innerLoop(0) As AcadEntity
    center(0) = 5: center(1) = 4,5: center(2) = 0
    radius = 1
    Set innerLoop(0) = ThisDrawing.ModelSpace. _
                                  AddCircle(center, radius)
    ' Append the circle as an inner loop to the hatch
    hatchObj.AppendInnerLoop (innerLoop)
    ' Evaluate and display the hatch
    hatchObj.Evaluate
    ThisDrawing.Regen True
End Sub
```

¿Comentarios?

Edición de patrones de sombreado

Puede modificar el ángulo y el espaciado de un patrón de sombreado existente o sustituirlo por un relleno sólido o por uno de los patrones predefinidos provistos por AutoCAD. La opción Patrón del cuadro de diálogo Sombreado permite ver una lista de los patrones. Para reducir el tamaño del archivo, el sombreado se define en el dibujo como un solo objeto gráfico.

Utilice los siguientes métodos y propiedades para modificar los patrones de sombreado:

PatternAngle

Determina el ángulo del patrón de sombreado.

PatternDouble

Indica si el sombreado definido por el usuario es un sombreado doble.

PatternName

Especifica el nombre del patrón de sombreado (no cambia el tipo del patrón).

PatternScale

Determina la escala de los patrones de sombreado.

PatternSpace

Especifica el espaciado del patrón de sombreado definido por el usuario.

SetPattern

Define el nombre y el tipo del patrón de sombreado.

Modificación del intervalo del patrón de un sombreado

Este ejemplo crea un sombreado. Después añade dos al espaciado del patrón actual del sombreado.

```
Sub Ch4_ChangeHatchPatternSpace()
    Dim hatchObj As AcadHatch
    Dim patternName As String
    Dim PatternType As Long
    Dim bAssociativity As Boolean
    ' Define the hatch
    patternName = "ANSI31"
    PatternType = 0
    bAssociativity = True
    ' Create the associative Hatch object
    Set hatchObj = ThisDrawing.ModelSpace. _
        AddHatch(PatternType, patternName, bAssociativity)
    ' Create the outer loop for the hatch.
    Dim outerLoop(0 To 0) As AcadEntity
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 5
    center(1) = 3
    center(2) = 0
    radius = 3
    Set outerLoop(0) = ThisDrawing.ModelSpace. _
                                AddCircle(center, radius)
    hatchObj.AppendOuterLoop (outerLoop)
    hatchObj.Evaluate
    ' Change the spacing of the hatch pattern by
    ' adding 2 to the current spacing
    hatchObj.patternSpace = hatchObj.patternSpace + 2
    hatchObj.Evaluate
    ThisDrawing.Regen True
End Sub
```

Uso de capas, colores y tipos de línea

Las capas son superposiciones transparentes en las cuales se organizan y se agrupan distintos tipos de datos del dibujo. Los objetos que se crean tienen propiedades comunes como capas, colores y tipos de línea. El color contribuye a establecer las diferencias oportunas entre elementos similares que componen el dibujo, y los tipos de línea sirven para distinguir fácilmente los distintos elementos del dibujo, como las líneas de centro o las líneas ocultas. La organización en capas de las capas y los objetos facilita el manejo de la información de los dibujos.

Para obtener más información acerca de este tema, véase “Control de las propiedades de los objetos” en el *Manual del usuario*.

- **Trabajar con capas**
- **Trabajo con colores**
- **Trabajo con tipos de línea**
- **Asignación de capas, colores y tipos de línea a objetos**

Trabajar con capas

Siempre se dibuja en una capa. Puede ser la capa por defecto o una capa que haya creado el usuario y a la que haya asignado un nombre. Cada capa tiene un color y un tipo de línea asociados. Por ejemplo, se puede crear una capa para dibujar solo líneas de centro, y asignarle a continuación el color azul y el tipo de línea CENTER. A partir de entonces, siempre que vaya a dibujar una línea de centro puede pasar a la capa de líneas de centro y comenzar a dibujar.

Todos los tipos de línea y las capas permanecen en los objetos de sus colecciones superiores. Las capas se mantienen en la colección Layers, mientras que los tipos de línea están en la colección Linetypes.

Para obtener más información acerca de las capas, véase “Utilización de capas“ en el *Manual del usuario*.

- **Ordenación de capas y tipos de línea**
- **Creación y denominación de capas**
- **Conversión de una capa en activa**
- **Activación y desactivación de capas**
- **Inutilización y reutilización de capas**
- **Bloqueo y desbloqueo de capas**
- **Asignación de color a una capa**
- **Asignación de un tipo de línea a una capa**
- **Supresión de capas**

Ordenación de capas y tipos de línea

Se puede iterar en las colecciones Layers y Linetypes para encontrar todas las capas y tipos de línea del dibujo.

Iteración en la colección de capas

El siguiente código efectúa iteraciones en la colección Layers para obtener los nombres de todas las capas del dibujo. Los nombres se muestran en un cuadro de mensajes.

```
Sub Ch4_IteratingLayers()
    Dim layerNames As String
    Dim entry As AcadLayer
    layerNames = ""
    For Each entry In ThisDrawing.Layers
        layerNames = layerNames + entry.Name + vbCrLf
    Next
    MsgBox "The layers in this drawing are: " + _
           vbCrLf + layerNames
End Sub
```

Creación y denominación de capas

Se puede crear capas nuevas y asignarles propiedades de color y tipo de línea. Todas las capas individuales forman parte de la colección Layers. Para crear un capa nueva y añadirla a la colección de capas, utilice el método Add.

El nombre de las capas se les asigna al crearlas. Si después de crear una capa desea cambiar su nombre, utilice la propiedad Name. Los nombres de capa pueden constar de un total de 31 caracteres y contener letras, números y los signos especiales de dólar ($), guión (–) y subrayado (_), pero *no pueden* contener espacios en blanco.

Para obtener más información acerca de la creación de capas, véase “Creación y denominación de capas” en el *Manual del usuario*.

Para crear una capa nueva, asignarle el color rojo y agregarle un objeto

El código del ejemplo siguiente crea un círculo y una capa nueva. Se asigna el color rojo a la nueva capa. El círculo se asigna a la capa y su color cambia en consecuencia.

```
Sub Ch4_NewLayer()
    ' Create a circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2: center(1) = 2: center(2) = 0
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace. _
                        AddCircle(center, radius)
    ' Create a color object
    Dim col As New AcadAcCmColor
    col.ColorMethod = AutoCAD.acColorMethodForeground
    ' Set the layer to the color
    Dim layColor As AcadAcCmColor
```

```
    Set layColor =
      AcadApplication.GetInterfaceObject("AutoCAD.AcCmColor.17")
    Call layColor.SetRGB(122, 199, 25)
    ThisDrawing.ActiveLayer.TrueColor = layColor
    col.ColorMethod = AutoCAD.acColorMethodByLayer
    ' Assign the circle the color "ByLayer" so
    ' that the circle will automatically pick
    ' up the color of the layer on which it resides
    circleObj.Color = acByLayer
    circleObj.Update
End Sub
```

¿Comentarios?

Conversión de una capa en activa

El dibujo siempre se realiza en la capa activa. Cuando una capa se activa, se pueden crear objetos nuevos en ella. Si se activa una capa distinta, los nuevos objetos se crearán en la nueva capa activa y utilizarán su color y tipo de línea. Las capas inutilizadas no pueden activarse.

Para convertir una capa en activa, utilice la propiedad ActiveLayer. Esta propiedad se define en el dibujo activo. Por ejemplo:

```
Dim newlayer As AcadLayer
Set newlayer = ThisDrawing.Layers.Add("LAYER1")
ThisDrawing.ActiveLayer = newlayer
```

Activación y desactivación de capas

Las capas desactivadas se regeneran con el dibujo, pero no se visualizan ni trazan. Al desactivarlas, se evita tener que regenerar el dibujo cada vez que se reutiliza una capa. Al activar una capa que se ha desactivado, AutoCAD vuelve a dibujar los objetos de esa capa.

Para activar y desactivar capas, utilice la propiedad LayerOn. Si le asigna el valor TRUE, la capa se activa. Si le asigna el valor FALSE, la capa se desactiva.

Desactivación de una capa

Este ejemplo crea una capa nueva, le añade un círculo y la desactiva para que el círculo no esté visible.

```
Sub Ch4_LayerInvisible()
    ' Create a circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2: center(1) = 2: center(2) = 0
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace. _
                        AddCircle(center, radius)
    ' Create a new layer called "ABC"
    Dim layerObj As AcadLayer
    Set layerObj = ThisDrawing.Layers.Add("ABC")
    ' Assign the circle to the "ABC" layer
    circleObj.Layer = "ABC"
    circleObj.Update
    ' Turn off layer "ABC"
    layerObj.LayerOn = False
    ThisDrawing.Regen acActiveViewport
End Sub
```

¿Comentarios?

Inutilización y reutilización de capas

Puede inutilizar capas para aumentar la velocidad de los cambios en pantalla, mejorar el rendimiento de la designación de objetos y reducir el tiempo de regeneración de dibujos complejos. AutoCAD no muestra, regenera ni traza objetos en las capas inutilizadas. Inutilice las capas que quiera que sean invisibles durante largos periodos de tiempo. Al “reutilizar” una capa inutilizada, AutoCAD regenera y muestra los objetos de dicha capa.

Para inutilizar o reutilizar una capa, utilice la propiedad Freeze. Si introduce un valor de TRUE en esta propiedad, la capa se inutiliza. Si le asigna el valor FALSE, la capa se reutiliza.

Inutilización de una capa

En este ejemplo se crea una nueva capa llamada “ABC” y, a continuación, se inutiliza.

```
Sub Ch4_LayerFreeze()
    ' Create a new layer called "ABC"
    Dim layerObj As AcadLayer
    Set layerObj = ThisDrawing.Layers.Add("ABC")
    ' Freeze layer "ABC"
    layerObj.Freeze = True
End Sub
```

Bloqueo y desbloqueo de capas

Los objetos dispuestos en una capa bloqueada no pueden editarse, pero permanecen visibles si la capa está activada y no está inutilizada. También es posible convertir en actual una capa bloqueada y añadirle objetos. Se puede inutilizar y desactivar capas bloqueadas así como cambiar sus propiedades asociadas de color y tipos de línea.

Para bloquear o desbloquear una capa, utilice la propiedad Lock. Si introduce un valor de TRUE en esta propiedad, la capa se bloquea. Si le asigna el valor FALSE, la capa se desbloquea.

Bloqueo de una capa

En este ejemplo se crea una nueva capa llamada “ABC” y, a continuación, se bloquea.

```
Sub Ch4_LayerLock()
    ' Create a new layer called "ABC"
    Dim layerObj As AcadLayer
    Set layerObj = ThisDrawing.Layers.Add("ABC")
    ' Lock layer "ABC"
    layerObj.Lock = True
End Sub
```

Asignación de color a una capa

Se puede asignar color a una capa. Los colores se identifican mediante el objeto AcCmColor. Este objeto puede contener un valor RGB, un número ACI (un entero comprendido entre 1 y 255) o un color guardado.

Para asignar color a una capa, utilice la propiedad TrueColor.

Se han proporcionado las constantes para los siete colores estándar y las designaciones BYBLOCK y BYLAYER.

Si utiliza `acByBlock`, AutoCAD dibuja los objetos nuevos con el color por defecto (blanco o negro, según la configuración) hasta que quedan agrupados en un bloque. Al insertar este bloque en el dibujo, los objetos adoptan el parámetro actual.

Si selecciona `acByLayer`, los objetos nuevos adoptan el color de la capa sobre la que se han dibujado.

Asignación de un tipo de línea a una capa

Cuando se definen capas, los tipos de línea constituyen otra manera de ofrecer información visual. Un tipo de línea es un patrón repetido de trazos, puntos y espacios en blanco que sirve para diferenciar la finalidad de cada línea.

El nombre y la definición del tipo de línea describen la secuencia particular trazo-punto y las longitudes relativas de los trazos, espacios en blanco y las características de cualquier texto o forma incluido.

Para asignar un tipo de línea a una capa, utilice la propiedad Linetype. Esta propiedad utiliza como entrada el nombre del tipo de línea.

Supresión de capas

Para suprimir una capa, utilice el método Delete.

Puede suprimir una capa en cualquier momento de la sesión de dibujo. No se pueden suprimir la capa actual, la capa 0 ni una capa que dependa de una referencia externa o que contenga objetos.

Nota Tampoco se pueden suprimir las capas a las que se refieren definiciones de bloque ni la capa especial llamada DEFPOINTS, aunque no contengan objetos visibles.

Trabajo con colores

Se pueden asignar colores verdaderos a objetos individuales de un dibujo mediante el AcCmColor. Si se utiliza un valor RGB en el objeto AcCmColor, podrá elegir entre millones de colores para definir el color de las líneas, de los círculos y de otros objetos individuales. El objeto AcCmColor también contiene métodos y propiedades para especificar nombres de colores, libros de colores, índices de colores, valores de colores y métodos de colores.

También se pueden asignar colores a capas. Cada color puede estar identificado por un nombre o un número del Índice de colores de AutoCAD (ACI), un entero entre 1 y 255. No hay límite para el número de objetos y capas que pueden utilizar el mismo número de color. Se puede utilizar el mismo número de color en todos los objetos y capas que sea necesario.

Cuando se especifica un color, se puede introducir su nombre o su número ACI. El índice ACI proporciona 255 números de colores. Sólo los siete primeros colores tienen un nombre estándar.

Colores del 1 al 7

Número de color	Nombre de color
1	Rojo
2	Amarillo
3	Verde
4	Cián

5	Azul
6	Magenta
7	Negro/Blanco

Los colores del 8 al 255 deben asignarse por números o seleccionando el color en el cuadro de diálogo. El color por defecto (7) puede ser blanco o negro, según el color de fondo.

Para obtener más información acerca del uso de los colores, véase “Utilización de colores” en el *Manual del usuario*.

Trabajo con tipos de línea

Un tipo de línea es un patrón de trazos, puntos y espacios en blanco que se repite. Un tipo de línea complejo es una patrón repetido de símbolos. Para utilizar un tipo de línea es indispensable cargarlo primero en el dibujo. Esto quiere decir que la definición de dicho tipo de línea debe estar incluida en el archivo de biblioteca .LIN para poder cargarlo en el dibujo. Para cargar un tipo de línea en el dibujo, utilice el método Load

Para obtener más información acerca del uso de los tipos de líneas, véase “Introducción a los tipos de líneas” en el *Manual del usuario*.

Nota Los tipos de línea utilizados internamente por AutoCAD no deben confundirse con los tipos de línea de impresora proporcionados por algunos trazadores. Los dos tipos de línea a trazos producen resultados similares. Sin embargo, *no* deben utilizarse los dos tipos al mismo tiempo, ya que los resultados pueden ser imprevisibles.

Carga de tipos de línea en AutoCAD

En este ejemplo se intenta cargar el tipo de línea “CENTER” del archivo *acad.lin*. Si este tipo de línea ya existe, o si el archivo no existe, se muestra un mensaje.

```
Sub Ch4_LoadLinetype()
    On Error GoTo ERRORHANDLER
    Dim linetypeName As String
    linetypeName = "CENTER"
    ' Carga el tipo de línea "CENTER" desde el archivo acad.lin
    ThisDrawing.Linetypes.Load linetypeName, "acad.lin"
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

- **Conversión de un tipo de línea en activo**
- **Cambio de nombre de tipos de línea**
- **Supresión de tipos de línea**
- **Cambio de descripciones de tipos de línea**
- **Designación de la escala del tipo de línea**

¿Comentarios?

Conversión de un tipo de línea en activo

Si desea utilizar un tipo de línea para dibujar en la capa actual, debe activarlo. Todos los objetos recién creados se trazan con el tipo de línea activo.

Nota Los tipos de línea que dependen de referencias externas no pueden convertirse en activos.

Para convertir un tipo de línea en activo, utilice la propiedad ActiveLinetype Esta propiedad se define en el dibujo activo. Por ejemplo:

```
ThisDrawing.ActiveLinetype = ThisDrawing. _
Linetypes.Item("CONTINUOUS")
```

Para obtener más información acerca de la activación de un tipo de línea, véase “Definición del tipo de línea actual” en el *Manual del usuario*.

Cambio de nombre de tipos de línea

Para cambiar de nombre un tipo de línea, utilice la propiedad Name. Al cambiar de nombre un tipo de línea, sólo se cambia el nombre de la definición del tipo de línea del dibujo. El nombre contenido en el archivo de biblioteca LIN no se actualiza para reflejar el nuevo nombre.

Supresión de tipos de línea

Para suprimir un tipo de línea, utilice el método Delete. El usuario puede suprimir un tipo de línea cuando lo desee durante una sesión de dibujo; no obstante, entre los tipos de línea que no pueden suprimirse se incluyen BALAYER, BAYBLOCK, CONTINUOUS, el tipo de línea actual y los tipos de línea que dependen de referencias externas. Asimismo, los tipos de línea referenciados por definiciones de bloque no se pueden suprimir aunque no los esté utilizando ningún objeto.

Para obtener más información acerca de la eliminación de un tipo de línea, véase “Definición del tipo de línea actual” en el *Manual del usuario*.

Cambio de descripciones de tipos de línea

Los tipos de línea pueden tener una descripción asociada. La descripción proporciona una representación ASCII del tipo de línea. Se puede asignar o cambiar la descripción de los tipos de línea con la propiedad Description.

La descripción de un tipo de línea puede tener una longitud máxima de 47 caracteres. Puede ser un comentario o una serie de caracteres de subrayado, puntos, trazos y espacios que sirva como una mera representación del patrón del tipo de línea. Por ejemplo:

```
ThisDrawing.ActiveLinetype.Description = "Exterior Wall"
```

Designación de la escala del tipo de línea

Se puede especificar la escala del tipo de línea para los objetos que se creen. Cuanto menor es la escala, más se repetirá el patrón del tipo de línea por unidad de dibujo. Por defecto, AutoCAD aplica una escala global de 1.0, que equivale a una unidad de dibujo. Se puede cambiar la escala del tipo de línea de todos los objetos de dibujo, las referencias de atributos y los grupos.

Para cambiar la escala del tipo de línea, utilice la propiedad LinetypeScale.

La variable de sistema CELTSCALE establece la escala del tipo de línea para los objetos de nueva creación. Si desea modificar globalmente la escala del tipo de línea de objetos existentes, utilice LTSCALE. Para cambiar los valores de variables de sistema con ActiveX Automation de AutoCAD, utilice el método SetVariable.

Para obtener más información acerca de las escalas del tipo de línea, véase “Control de la escala del tipo de línea“ en el *Manual del usuario*.

Modificación de la escala del tipo de línea de un círculo

```
Sub Ch4_ChangeLinetypeScale()
  ' Save the current linetype
  Set currLineType = ThisDrawing.ActiveLinetype
  ' Change the active linetype to Border, so the scale change will
  ' be visible.
  ' First see if the Border linetype is already loaded
  On Error Resume Next 'Turn on error trapping
  ThisDrawing.ActiveLinetype = ThisDrawing.Linetypes.Item("BORDER")
  If Err.Number = -2145386476 Then
    ' Error indicates linetype is not currently loaded, so load it.
    ThisDrawing.Linetypes.Load "BORDER", "acad.lin"
    ThisDrawing.ActiveLinetype = _
          ThisDrawing.Linetypes.Item("BORDER")
```

```
  End If
  On Error GoTo 0 'Turn off error trapping
  ' Create a circle object in model space
  Dim center(0 To 2) As Double
  Dim radius As Double
  Dim circleObj As AcadCircle
  center(0) = 2
  center(1) = 2
  center(2) = 0
  radius = 4
  Set circleObj = ThisDrawing.ModelSpace.AddCircle(center, radius)
  circleObj.Update
  MsgBox ("Here is the circle with the original linetype")
  ' Set the linetype scale of a circle to 3
  circleObj.LinetypeScale = 3#
  circleObj.Update
  MsgBox ("Here is the circle with the new linetype")
  ' Restore original active linetype
  ThisDrawing.ActiveLinetype = currLineType
End Sub
```

¿Comentarios?

Asignación de capas, colores y tipos de línea a objetos

Después de definir capas, colores y tipos de línea, pueden asignarse a objetos del dibujo. La asignación de objetos a capas permite agrupar los componentes asociados de un dibujo. Se controla la visibilidad, el color y el tipo de línea de la capa y se especifica si los objetos de una capa pueden editarse. Se pueden desplazar objetos de una capa a otra y cambiar el nombre de la capa.

El número de capas de un dibujo y el número de objetos por cada capa son prácticamente ilimitados. Se puede asignar un nombre a cada capa y seleccionar la visualización de cualquier combinación de capas.

Los bloques se definen a partir de objetos dibujados originalmente en diferentes capas con distintos colores y tipos de línea. Se puede conservar la información sobre la capa, el color y el tipo de línea de un bloque. Esto permite que cada vez que se inserte el bloque, los objetos del mismo se dibujen en su capa original con su color y tipo de línea originales.

- **Cambio de la capa de un objeto**
- **Cambio del color de un objeto**
- **Cambio del tipo de línea de un objeto**

Cambio de la capa de un objeto

Al crear un objeto y asignarle propiedades de capa, color y tipo de línea, puede cambiar la capa del objeto. Este cambio resulta útil si se ha creado por error un objeto en una capa equivocada o si más tarde se decide cambiar la organización de capas.

Para cambiar la capa de un objeto, utilice la propiedad Layer del objeto. Esta propiedad utiliza como entrada el nombre de la capa.

Traslado de un objeto a otra capa

En este ejemplo se crea un círculo en la capa activa y, a continuación, se crea una capa nueva llamada “ABC”. A continuación, desplaza el círculo a la nueva capa.

```
Sub Ch4_MoveObjectNewLayer()
    ' Create a circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2: center(1) = 2: center(2) = 0
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace. _
                        AddCircle(center, radius)
    ' Create a new layer called "ABC"
    Dim layerObj As AcadLayer
    Set layerObj = ThisDrawing.Layers.Add("ABC")
    ' Assign the circle to the "ABC" layer
    circleObj.Layer = "ABC"
    circleObj.Update
End Sub
```

¿Comentarios?

Cambio del color de un objeto

Para cambiar el color de un objeto, utilice la propiedad TrueColor del objeto. Se pueden asignar colores a objetos concretos de un dibujo. Cada color está identificado por un objeto AcCmColor. Este objeto puede contener un valor RGB, un número ACI (un entero comprendido entre 1 y 255) o un color guardado. Mediante los valores RGB se puede elegir entre millones de colores.

Al asignar un color al objeto se ignora el color actual de la capa en la que reside el objeto. Si se desea conservar un objeto en una determinada capa pero sin que comparta el color de la misma, se puede cambiar el color concreto del objeto.

Cambio del color de un círculo

Este ejemplo crea un círculo y después le asigna el color azul.

```
Sub Ch4_ColorCircle()
    Dim color As AcadAcCmColor
    Set color = _
    AcadApplication.GetInterfaceObject("AutoCAD.AcCmColor.17")
    Call color.SetRGB(80, 100, 244)
    Dim circleObj As AcadCircle
    Dim centerPoint(0 To 2) As Double
    Dim radius As Double
    centerPoint(0) = 0#: centerPoint(1) = 0#: centerPoint(2) = 0#
    radius = 5#
    Set circleObj = _
    ThisDrawing.ModelSpace.AddCircle(centerPoint, radius)
    circleObj.TrueColor = color
    ZoomAll
End Sub
```

Cambio del tipo de línea de un objeto

Por defecto, los objetos utilizan el tipo de línea de la capa en la que se crean. Para cambiar el tipo de línea de un objeto, utilice la propiedad Linetype del objeto. Esta propiedad utiliza como entrada el nombre del tipo de línea que se desea asignar al objeto.

Nota Para poder asignar un tipo de línea a un objeto, el tipo de línea debe encontrarse cargado en el dibujo actual. Para cargar un tipo de línea en el dibujo, utilice el método Load.

Para obtener más información acerca de los tipos de línea, véase “Introducción a los tipos de líneas” en el *Manual del usuario*.

Cambio del tipo de línea de un círculo

Este ejemplo crea un círculo. A continuación, se intenta cargar el tipo de línea “CENTER” del archivo *acad.lin*. Si este tipo de línea ya existe, o si el archivo no existe, se muestra un mensaje. Por último, se establece el tipo de línea del círculo en “CENTER.”.

```
Sub Ch4_ChangeCircleLinetype()
    On Error Resume Next
    ' Create a circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2: center(1) = 2: center(2) = 0
    radius = 1
    Set circleObj = ThisDrawing.ModelSpace. _
                        AddCircle(center, radius)
    Dim linetypeName As String
    linetypeName = "CENTER"
    ' Carga el tipo de línea "CENTER" desde el archivo acad.lin
```

```
    ThisDrawing.Linetypes.Load linetypeName, "acad.lin"
    If Err.Description <> "" Then MsgBox Err.Description
    ' Assign the circle the linetype "CENTER"
    circleObj.Linetype = "CENTER"
    circleObj.Update
End Sub
```

¿Comentarios?

<$nopage>capa (propiedades), guardar. <$endrange>capa (parámetros):almacenar, capa (parámetros):guardar:código de ejemplo, capa (parámetros):cambiar nombre a parámetros guardados:código de ejemplo, capa (parámetros):suprimir parámetros guardados:código de ejemplo, capa (parámetros):restablecer parámetros guardados, capa (parámetros):restablecer parámetros guardados:código de ejemplo, capa (parámetros):exportar parámetros guardados, capa (parámetros):importar parámetros guardados, Export (método):para parámetros de capa guardados, Import (método):para parámetros de capa guardados, capa (parámetros):exportar parámetros guardados:código de ejemplo, capa (parámetros):importar parámetros guardados:código de ejemplo,">

Almacenamiento y restablecimiento de parámetros de capas

Se pueden guardar los parámetros de las capas, y restablecerlos cuando sea necesario. De esta forma, se pueden restituir los parámetros generales definidos para las capas en las diversas etapas de realización del dibujo o de su impresión en el trazador.

Los parámetros de capa indican si la capa está o no activada, inutilizada, bloqueada, impresa en el trazador o inutilizada automáticamente en las ventanas nuevas; también determinan el color, el tipo de línea, el grosor de línea y el estilo de trazado de la capa. Se pueden especificar los parámetros que se desean almacenar de un dibujo, e incluso guardarlos en grupos.

Un objeto especial denominado LayerStateManager proporciona funciones para trabajar con los parámetros de capa en ActiveX.

Para obtener más información acerca del almacenamiento de parámetros de capas, véase “Guardado y restablecimiento de parámetros de capas“ en el *Manual del usuario*.

- **Conceptos básicos del almacenamiento de parámetros de capas en**

AutoCAD

- **Uso de LayerStateManager para gestionar parámetros de capa**

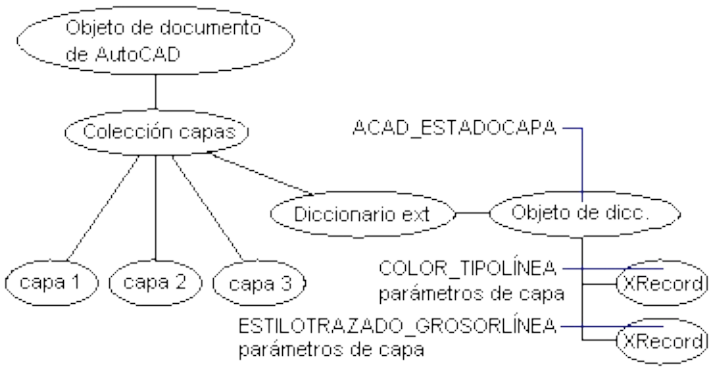
¿Comentarios?

Conceptos básicos del almacenamiento de parámetros de capas en AutoCAD

AutoCAD guarda los parámetros de capas en un diccionario de extensiones de la colección Layers de un dibujo. Cuando se guardan parámetros de capa de un dibujo por primera vez, AutoCAD procede como se indica a continuación:

- Crea un diccionario de extensiones en la colección de capas.
- Crea un objeto Dictionary denominado ACAD_LAYERSTATE en el directorio de extensiones.
- Almacena las propiedades de todas las capas del dibujo en un objeto XRecord del diccionario ACAD_LAYERSTATE. AutoCAD almacena todos los parámetros de capas incluidos en XRecord, pero identifica aquellos parámetros que el usuario eligió guardar. Cuando se restablecen los parámetros de las capas, AutoCAD restituye sólo los que el usuario eligió guardar.

Cada vez que se guarda otro parámetro de capa en el dibujo, AutoCAD crea otro objeto XRecord con la descripción de los parámetros guardados y almacena XRecord en el diccionario ACAD_LAYERSTATE. El siguiente diagrama ilustra este proceso.



No es necesario (y tampoco debe intentarse) interpretar los objetos Xrecord cuando se trabaja con parámetros de capa en ActiveX. Utilice las funciones del objeto LayerStateManager para acceder a los parámetros de capa guardados.

Listado de los parámetros de capa guardados en un dibujo

Si se guardaron los parámetros de capa en el dibujo actual, el siguiente código extrae una lista con los nombres de todos los parámetros de capa guardados:

```
Sub Ch4_ListStates()
     On Error Resume Next
    Dim oLSMDict As AcadDictionary
    Dim XRec As Object
    Dim layerstateNames As String
    layerstateNames = ""
    ' Get the ACAD_LAYERSTATES dictionary, which is in the
    ' extension dictionary in the Layers object.
    Set oLSMDict = ThisDrawing.Layers. _
           GetExtensionDictionary.Item("ACAD_LAYERSTATES")
    ' List the name of each saved layer setting. Settings are
    ' stored as XRecords in the dictionary.
    For Each XRec In oLSMDict
       layerstateNames = layerstateNames + XRec.Name + vbCrLf
    Next XRec
    MsgBox "The saved layer settings in this drawing are: " + _
            vbCrLf + layerstateNames
End Sub
```

<$endrange>capa (parámetros):almacenar, capa (parámetros):guardar:código de ejemplo, capa (parámetros):cambiar nombre a parámetros guardados:código de ejemplo, capa (parámetros):suprimir parámetros guardados:código de ejemplo, capa (parámetros):restablecer parámetros guardados, capa (parámetros):restablecer parámetros guardados:código de ejemplo, capa (parámetros):exportar parámetros guardados, capa (parámetros):importar parámetros guardados, Export (método):para parámetros de capa guardados, Import (método):para parámetros de capa guardados, capa (parámetros):exportar parámetros guardados:código de ejemplo, capa (parámetros):importar parámetros guardados:código de ejemplo,">

Uso de LayerStateManager para gestionar parámetros de capa

El objeto LayerStateManager, al igual que el objeto Utility de AutoCAD, proporciona un conjunto de funciones de manipulación de datos. Estas funciones son métodos para trabajar con parámetros de capa guardados. Use los siguientes métodos LayerStateManager para trabajar con parámetros de capa guardados :

Delete

Suprime un parámetro de capa guardado.

Export

Exporta a un archivo el parámetro de capa guardado que se haya indicado.

Import

Importa un parámetro de capa guardado desde el archivo indicado.

Rename

Cambia el nombre de un parámetro de capa guardado.

Restore

Restablece en el dibujo actual el parámetro de capa indicado.

Save

Guarda los estados y propiedades de la capa indicada.

SetDataBase

Asocia una base de datos de AutoCAD al objeto LayerStateManager.

Para acceder al objeto LayerStateManager, utilice el método GetInterfaceObject.

```
Dim oLSM As AcadLayerStateManager
Set oLSM = ThisDrawing.Application. _
   GetInterfaceObject("AutoCAD.AcadLayerStateManager.17")
```

Una vez recuperado el objeto LayerStateManager, se ha de asociar a una base de datos para tener acceso a sus métodos. Utilice el método SetDatabase para asociar una base de datos al objeto LayerStateManager.

```
oLSM.SetDatabase ThisDrawing.Database
```

- **Guardado de parámetros de capas**
- **Restitución de parámetros de capa**
- **Exportación e importación de parámetros de capa guardados**

<$endrange>capa (parámetros):almacenar, capa (parámetros):guardar:código de ejemplo, capa (parámetros):cambiar nombre a parámetros guardados:código de ejemplo, capa (parámetros):suprimir parámetros guardados:código de ejemplo,">

Guardado de parámetros de capas

Utilice el método Save para guardar un conjunto de parámetros de capa de un dibujo. El método Save acepta dos parámetros. El primer parámetro consiste en una cadena con el nombre de los parámetros de capa que se deben guardar. El segundo parámetro identifica las propiedades de capa que desean utilizarse. Utilice las constantes de la siguiente tabla para identificar propiedades de capa.

Constantes de propiedades de capa

Nombre de constante	Layer (propiedad)
acLsAll	Todos los parámetros de capa
acLsColor	Color
acLsFrozen	Inutilizada o reutilizada
acLsLineType	Linetype
acLsLineWeight	Grosor de línea

acLsLocked	Bloqueada o desbloqueada
acLsNewViewport	Capas inutilizadas o reutilizadas en ventanas nuevas
acLsNone	Ninguna
acLsOn	Activada o desactivada
acLsPlot	Trazado activado o desactivado
acLsPlotStyle	Estilo de trazado

Las constantes pueden añadirse juntas para precisar varias propiedades.

Si intenta guardar parámetros de capa con un nombre que ya existe, aparece un error. Para poder reutilizar un nombre, es necesario suprimir o cambiar de nombre los parámetros de capa guardados ya existentes.

Guardado de los parámetros de color y tipo de línea de una capa

El siguiente código guarda los parámetros de color y tipo de línea de la capa actual bajo el nombre `ColorLinetype`.

```
Sub Ch4_SaveLayerColorAndLinetype()
    Dim oLSM As AcadLayerStateManager
    ' Access the LayerStateManager object
    Set oLSM = ThisDrawing.Application. _
       GetInterfaceObject("AutoCAD.AcadLayerStateManager.17")
    ' Associate the current drawing database with LayerStateManager
    oLSM.SetDatabase ThisDrawing.Database
    oLSM.Save "ColorLinetype", acLsColor + acLsLineType
End Sub
```

Cambio de nombre de un parámetro de capa guardado

El siguiente código cambia el nombre de los parámetros de capa `ColorLinetype` por el nombre `OldColorLinetype`.

```
Sub Ch4_RenameLayerSettings()
    Dim oLSM As AcadLayerStateManager
    Set oLSM = ThisDrawing.Application. _
       GetInterfaceObject("AutoCAD.AcadLayerStateManager.17")
    oLSM.SetDatabase ThisDrawing.Database
    oLSM.Rename "ColorLinetype", "OldColorLinetype"
End Sub
```

Supresión de un parámetro de capa guardado

El siguiente código suprime los parámetros de capa que se guardaron bajo el nombre `ColorLinetype.`

```
Sub Ch4_DeleteColorAndLinetype()
    Dim oLSM As AcadLayerStateManager
    Set oLSM = ThisDrawing.Application. _
       GetInterfaceObject("AutoCAD.AcadLayerStateManager.17")
    oLSM.SetDatabase ThisDrawing.Database
    oLSM.Delete "ColorLinetype"
End Sub
```

Restitución de parámetros de capa

El método Restore restablece los valores que se guardaron con anterioridad en todos los parámetros de capa del dibujo actual. Por ejemplo, si guarda los parámetros de color y de tipo de línea con el nombre “ColorLinetype” y posteriormente cambia los parámetros, al restablecer “ColorLinetype” se restituyen los colores y los tipos de línea que tenían las capas al guardar “ColorLinetype”. Las capas nuevas que se agreguen al dibujo después de guardar “ColorLinetype” no se ven afectadas por la restitución de “ColorLinetype. ”.

Restitución de los parámetros de color y tipo de línea de las capas de un dibujo

El siguiente código de ejemplo, que presupone que los parámetros de color y tipo de línea de las capas del dibujo actual se guardaron con el nombre “ColorLinetype,”, restablece los parámetros de color y tipo de línea de todas las capas del dibujo con el valor que tenían cuando se guardó “ColorLinetype,”.

```
Sub Ch4_RestoreLayerSettings()
    Dim oLSM As AcadLayerStateManager
    Set oLSM = ThisDrawing.Application. _
       GetInterfaceObject("AutoCAD.AcadLayerStateManager.17")
    oLSM.SetDatabase ThisDrawing.Database
    oLSM.Restore "ColorLinetype"
End Sub
```

Exportación e importación de parámetros de capa guardados

Se pueden exportar e importar parámetros de capa guardados para utilizarlos en otros dibujos. Con el método Export del objeto LayerStateManager, los parámetros de capa se guardan en un archivo; con el método Import, se importan a un dibujo.

Nota La importación de parámetros de capa no implica su restitución; es necesario utilizar el método Restore para que las capas del dibujo adquieran los parámetros importados.

El método Export admite dos parámetros. El primer parámetro consiste en una cadena identificativa de los parámetros de capa que se deseen exportar. El segundo parámetro es el nombre del archivo al que se deseen exportar. Si no se indica una ruta, el archivo se guarda en el directorio de instalación de AutoCAD. Si ya existe un archivo con el nombre especificado, el nuevo nombre sustituye al anterior. Al guardar los archivos, utilice la extensión *.las*; esta extensión permite a AutoCAD reconocer los archivos de parámetros de capa exportados.

El método Import acepta un parámetro: una cadena que señale el nombre del archivo que contiene los parámetros de capa que se desea importar.

Durante la importación de parámetros de capa, puede producirse una condición de error si las propiedades a las que refieren los parámetros guardados no están disponibles en el dibujo destino de la importación. La importación termina, no obstante, y se utilizan las propiedades por defecto. Por ejemplo, cuando en una capa exportada se establece un tipo de línea que no está cargado en el dibujo destino de la importación, se produce una condición de error y el tipo de línea por defecto del dibujo queda reemplazado. El código que se escriba debe tener en cuenta esta condición de error y continuar con el proceso si se produce.

Si el archivo importado define parámetros de capas que no existen en el dibujo actual, se crean dichas capas. Cuando se utiliza el método Restore, las propiedades que se especificaron al guardar los parámetros se asignan a las capas nuevas; todas las demás propiedades de las capas nuevas adquieren los parámetros establecidos por defecto.

Exportación de parámetros de capa guardados

El siguiente código exporta los parámetros de capa guardados a un archivo denominado *Colortype.las*.

```
Sub Ch4_ExportLayerSettings()
    Dim oLSM As AcadLayerStateManager
    Set oLSM = ThisDrawing.Application. _
       GetInterfaceObject("AutoCAD.AcadLayerStateManager.17")
    oLSM.SetDatabase ThisDrawing.Database
    oLSM.Export "ColorLinetype", "c:\my documents\ColorLType.las"
End Sub
```

Importación de parámetros de capa guardados

El siguiente código importa los parámetros de capa guardados desde un archivo denominado *Colortype.las*.

```
Sub Ch4_ImportLayerSettings()
    Dim oLSM As AcadLayerStateManager
    Set oLSM = ThisDrawing.Application. _
       GetInterfaceObject("AutoCAD.AcadLayerStateManager.17")
    oLSM.SetDatabase ThisDrawing.Database
    ' If the drawing you're importing to does not contain
    ' all the linetypes referenced in the saved settings,
    ' an error is returned. The import is completed, though,
    ' and the default linetype is used.
    On Error Resume Next
    oLSM.Import "c:\my documents\ColorLType.las"
    If Err.Number = -2145386359 Then
       ' Error indicates a linetype is not defined
       MsgBox ("One or more linetypes specified in the imported " +
              "settings is not defined in your drawing")
    End If
    On Error GoTo 0
End Sub
```

¿Comentarios?

<$nopage>texto de líneas múltiples.

Adición de texto a dibujos

El texto de los dibujos proporciona al usuario información de relevancia. Así, puede emplear texto para los cuadros de títulos para asignar etiquetas al dibujo, indicar características o incluir anotaciones.

AutoCAD pone a su disposición distintas técnicas para crear texto. Para entradas breves y sencillas, utilice texto de una línea. Si desea escribir texto de proporciones mayores con formato, utilice texto de líneas múltiples (textoM). Aunque todo texto que se introduzca usa el estilo de texto actual, que establece el tipo de letra y los parámetros de formato por defecto, es posible utilizar varios métodos para personalizar su aspecto.

Para obtener más información acerca del uso de texto, véase “Creación de texto” en el *Manual del usuario*.

- **Utilización de los estilos de texto**
- **Uso del texto en una línea (Text)**
- **Uso del texto de líneas múltiples (Mtext)**
- **Uso de caracteres Unicode, códigos de control y caracteres especiales**
- **Sustitución de tipos de letra**
- **Corrección ortográfica**

Utilización de los estilos de texto

El texto de los dibujos de AutoCAD tiene un estilo asociado. Al escribir texto, AutoCAD emplea el estilo de texto actual, que determina el tipo de letra, el tamaño, el ángulo, la orientación y otras características del texto. Puede utilizar o modificar el estilo por defecto o crear y cargar un nuevo estilo. Una vez que haya creado un estilo puede modificar sus atributos o borrarlo cuando ya no lo necesite.

- **Creación y modificación de estilos de texto**
- **Asignación de tipos de letra**
- **Tipos de letra TrueType**
- **Tipos de letra Unicode y Grandes**
- **Ajuste de altura del texto**
- **Ajuste del ángulo de oblicuidad**
- **Establecimiento de indicadores de generación de texto**

Creación y modificación de estilos de texto

Cuando se crea un texto, AutoCAD asume las propiedades del estilo de texto actual, entre las que se incluye la altura, la relación anchura/altura, el ángulo de oblicuidad y las propiedades de generación de texto. Si desea crear un estilo de texto, utilice el método Add para crear un objeto TextStyle nuevo y añadirlo a la colección TextStyles. El método Add utiliza como entrada un nombre de TextStyle. Una vez creado, el nombre de los estilos de texto no se puede cambiar con ActiveX Automation de AutoCAD.

Los nombres de estilo pueden contener letras, números y los caracteres especiales del signo de dólar ($), subrayado (_) y guión (–). AutoCAD pasa los caracteres a mayúsculas. Si no se especifica un nombre de estilo, AutoCAD le asigna automáticamente el nombre Estilo*n*, donde *n* es un número que empieza en 1. Cada nuevo estilo presenta un incremento de 1.

Los estilos existentes se pueden modificar si se cambian las propiedades del objeto TextStyle. También puede actualizar texto existente de ese tipo de estilo para que los cambios queden reflejados. Para modificar un objeto TextStyle, utilice las siguientes propiedades:

FontFile

Especifica el archivo asociado a un tipo de letra (estilo de carácter).

BigFontFile

Especifica el archivo de definición de formas especiales empleado en conjuntos de caracteres no ASCII.

Height

Especifica la altura de los caracteres.

Width

Especifica la expansión o compresión de los caracteres.

ObliqueAngle

Especifica la inclinación de los caracteres.

TextGenerationFlag

Especifica si el texto debe generarse hacia la izquierda, boca abajo o en ambos sentidos.

Al modificar el tipo de letra o la orientación de un estilo existente, todo el texto que emplee dicho estilo cambia para reflejar el tipo de letra o la orientación nuevos. Al cambiar la altura del texto, la relación anchura/altura o el ángulo de oblicuidad no se modifica el texto existente, pero sí afecta a los objetos de texto que se creen posteriormente.

Nota Llame a uno de los métodos Regen o Update para ver los cambios efectuados en las propiedades anteriores.

Asignación de tipos de letra

Los tipos de letra definen las formas de los caracteres de texto que constituyen los juegos de caracteres. Un tipo de letra puede ser utilizado en más de un estilo. Para asignar un tipo de letra a un estilo de texto, utilice la propiedad FontFile del objeto TextStyle. Puede asignar un tipo de letra compilado (SHX) propio de AutoCAD al estilo de texto, especificando el archivo de tipo de letra que lo contenga.

Establecimiento de los tipos de letra del texto

En este ejemplo se obtienen los valores de tipo de letra del estilo de texto activo y se cambia a la familia de tipos “PlayBill.” Después establece el nuevo tipo de letra mediante el método SetFont. Para ver cómo afecta el cambio de tipo de letra al texto, añada un objeto Mtext o Text al dibujo actual antes de ejecutar el ejemplo. Recuerde que, si el sistema no dispone del tipo de letra PlayBill, debe sustituirlo por algún tipo que tenga instalado su sistema; de lo contrario, el código de ejemplo no funcionará.

```
Sub Ch4_UpdateTextFont()
    MsgBox ("Look at the text now...")
    Dim typeFace As String
    Dim SavetypeFace As String
    Dim Bold As Boolean
    Dim Italic As Boolean
    Dim charSet As Long
    Dim PitchandFamily As Long
    ' Get the current settings to fill in the
    ' default values for the SetFont method
    ThisDrawing.ActiveTextStyle.GetFont typeFace, _
                 Bold, Italic, charSet, PitchandFamily
    ' Change the typeface for the font
    SavetypeFace = typeFace
    typeFace = "PlayBill"
```

```
    ThisDrawing.ActiveTextStyle.SetFont typeFace, _
                Bold, Italic, charSet, PitchandFamily
    ThisDrawing.Regen acActiveViewport
    MsgBox ("Now see how it looks after changing the font...")
    'Restore the original typeface
    ThisDrawing.ActiveTextStyle.SetFont SavetypeFace, _
                Bold, Italic, charSet, PitchandFamily
    ThisDrawing.Regen acActiveViewport
End Sub
```

Tipos de letra TrueType

Los tipos de letra TrueType aparecen siempre rellenos en los dibujos; sin embargo, cuando se traza, la variable de sistema TEXTFILL controla si los tipos de letra se rellenan o no. Por defecto, el valor de TEXTFILL es 1, para trazar tipos de letra rellenos. Si exporta el dibujo al formato PostScript® con el método Export y lo imprime en un dispositivo PostScript, el tipo de letra se trazará tal como se haya diseñado.

Tipos de letra Unicode y Grandes

AutoCAD admite el estándar de codificación de caracteres Unicode. Un tipo de letra Unicode puede contener 65.535 caracteres, con formas para muchos idiomas. Todos los tipos de letra de AutoCAD con la extensión .SHX se consideran tipos de letra Unicode.

Los archivos de texto de algunos alfabetos contienen miles de caracteres que no son ASCII. Para permitir dicho texto, AutoCAD admite un tipo especial de definición de forma conocido como archivo Big Font. Puede definir un estilo para usar tanto archivos normales como Big Font. Especifique fuentes normales utilizando la propiedad FontFile. Especifique Big Fonts utilizando la propiedad BigFontFile.

Nota El nombre de los archivos de tipos de letra no debe incluir comas.

AutoCAD pone a su disposición los métodos necesarios para llevar a cabo la sustitución de un tipo de letra por otro o la definición de un tipo de letra por defecto. Para obtener más información, véase Sustitución de tipos de letra.

Cambio de los archivos de tipos de letra

Este ejemplo cambia las propiedades FontFile y BigFontFile. Debe reemplazar la información de ruta de acceso de este ejemplo con los nombres de archivo y ruta apropiados en su sistema.

```
Sub Ch4_ChangeFontFiles()
    ThisDrawing.ActiveTextStyle.BigFontFile = _
                      "C:/AutoCAD/Fonts/bigfont.shx"
    ThisDrawing.ActiveTextStyle.fontFile = _
                      "C:/AutoCAD/Fonts/italic.shx"
End Sub
```

¿Comentarios?

Ajuste de altura del texto

La altura del texto determina el tamaño de los caracteres del tipo de letra utilizado en unidades de dibujo. El valor suele representar el tamaño de las letras mayúsculas, exceptuando los tipos de letra TrueType.

En el caso de los tipos de letra TrueType, es posible que el valor de la altura del texto no represente la altura de las letras mayúsculas. La altura especificada representa la altura de las letras mayúsculas más un área de acentos reservada para tildes y otras marcas utilizadas en idiomas que no son inglés. La parte de área proporcional asignada a las letras mayúsculas y a los caracteres acentuados está determinada por el diseñador del tipo de letra en el momento de crearlo y varía de un tipo a otro.

Además de la altura de las letras mayúsculas y del área ascendente que incluye la altura especificada por el usuario, los tipos de letra TrueType tienen un área descendente para aquellas partes de los caracteres que sobrepasen por debajo la línea de inserción de texto. Algunos ejemplos de estos caracteres son y, j, p, g, q.

La altura del texto se precisa mediante la propiedad Height. Esta propiedad sólo admite números positivos.

Modificación de la altura de un objeto Text

Este ejemplo crea una línea de texto y después cambia la altura del texto.

```
Sub Ch4_ChangeTextHeight()
    Dim textObj As AcadText
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim height As Double
    ' Define the text object
    textString = "Hello, World."
    insertionPoint(0) = 3
```

```
    insertionPoint(1) = 3
    insertionPoint(2) = 0
    height = 0,5
    ' Create the text object in model space
    Set textObj = ThisDrawing.ModelSpace. _
                  AddText(textString, insertionPoint, height)
    ' Change the value of the Height to 1
    textObj.height = 1
    textObj.Update
End Sub
```

¿Comentarios?

Ajuste del ángulo de oblicuidad

El ángulo de oblicuidad determina el grado de inclinación del texto. El ángulo representa el desfase desde el eje vertical (90 grados). Para establecer el ángulo de oblicuidad, utilice la propiedad ObliqueAngle. El ángulo se especifica en radianes. Un ángulo positivo denota una inclinación hacia la derecha y a los valores negativos se les suma `2*PI` para convertirlos en su equivalente positivo.

Creación de texto oblicuo

Este ejemplo crea un objeto Text y le aplica un ángulo de oblicuidad de 45 grados.

```
Sub Ch4_ObliqueText()
    Dim textObj As AcadText
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim height As Double
    ' Define the text object
    textString = "Hello, World."
    insertionPoint(0) = 3
    insertionPoint(1) = 3
    insertionPoint(2) = 0
    height = 0,5
    ' Create the text object in model space
    Set textObj = ThisDrawing.ModelSpace. _
              AddText(textString, insertionPoint, height)
    ' Change the value of the ObliqueAngle
    ' to 45 degrees (.707 radians)
    textObj.ObliqueAngle = 0.707
    textObj.Update
End Sub
```

Establecimiento de indicadores de generación de texto

El indicador de generación de texto determina si el texto se muestra reflejado hacia la izquierda o boca abajo. Para establecer el indicador de generación de texto, utilice la propiedad TextGenerationFlag. Introduzca `acTextFlagBackward` en la propiedad si desea que el texto se genere hacia la izquierda, o `acTextFlagUpsideDown` para que se refleje hacia abajo. Para que el texto se muestre reflejado hacia la izquierda y boca abajo, añada las dos constantes a la vez, especificando `acTextFlagBackward+acTextFlagUpsidedown` como valor de esta propiedad.

Presentación del texto reflejado hacia la izquierda

En este ejemplo se crea una línea de texto y, mediante la propiedad TextGenerationFlag, se establece que se muestre hacia la izquierda.

```
Sub Ch4_ChangingTextGenerationFlag()
    Dim textObj As AcadText
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim height As Double
    ' Create the text object
    textString = "Hello, World."
    insertionPoint(0) = 3
    insertionPoint(1) = 3
    insertionPoint(2) = 0
    height = 0,5
    Set textObj = ThisDrawing.ModelSpace. _
                AddText(textString, insertionPoint, height)
    ' Change the value of the TextGenerationFlag
    textObj.TextGenerationFlag = acTextFlagBackward
    textObj.Update
End Sub
```

¿Comentarios?

Uso del texto en una línea (Text)

El texto que se añade a los dibujos incluye una gran variedad de información. Puede tratarse de una especificación detallada, un cuadro de rotulación, una etiqueta o incluso una parte de un dibujo. Para las entradas breves que no requieran varios tipos de letra o varias líneas, cree el texto de la línea mediante el objeto Text. El texto en una línea es más adecuado para las etiquetas.

- **Creación de líneas de texto**
- **Formato del texto en línea**
- **Alineación del texto de una línea**
- **Modificación del texto de una línea**

Creación de líneas de texto

Cuando se utiliza esta función de texto, cada línea de texto es un objeto independiente. Los objetos de texto en una línea se crean con el método AddText. Este método requiere tres valores de entrada: la cadena de texto, el punto de inserción, y la altura del texto.

La cadena de texto es el texto real que se desea mostrar. Se puede utilizar Unicode, código de control y caracteres especiales. El punto de inserción es una matriz de variantes con tres dobles que indican las coordenadas 3D del SCU del dibujo donde se incluirá el texto. La altura del texto es un número positivo que representa la altura de los caracteres en mayúscula. La altura se indica en la unidad de medida activa.

Creación de líneas de texto

El código siguiente crea una línea de texto en espacio modelo, en las coordenadas (2, 2, 0).

```
Sub Ch4_CreateText()
    Dim textObj As AcadText
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim height As Double
    ' Create the text object
    textString = "Hello, World."
    insertionPoint(0) = 2
    insertionPoint(1) = 2
    insertionPoint(2) = 0
    height = 0,5
    Set textObj = ThisDrawing.ModelSpace. _
                AddText(textString, insertionPoint, height)
    textObj.Update
End Sub
```

¿Comentarios?

Formato del texto en línea

Los objetos Text que se crean utilizan el estilo de texto activo. Puede cambiar el formato del objeto Text modificando el estilo de texto que tenga asociado o bien editando las propiedades del objeto Text. *No se pueden* aplicar formatos a palabras o caracteres individuales.

Para cambiar el estilo de texto asociado a un objeto Text concreto, defina la propiedad StyleName con un estilo de texto nuevo. Una vez cambiado el estilo, utilice el método Update del objeto Text para ver los cambios en el dibujo.

Entre las propiedades que pueden cambiarse en un objeto Text se incluyen, además de las propiedades estándar modificables de las entidades (color, capa, tipo de línea, etc.):

Alignment

Determina la alineación horizontal y vertical del texto.

InsertionPoint

Determina el punto de inserción del texto.

ObliqueAngle

Determina el ángulo de oblicuidad del objeto Text individual.

Rotation

Determina el ángulo de rotación del texto, en radianes.

ScaleFactor

Determina el factor de escala del texto.

TextAlignmentPoint

Determina el punto de alineación del texto.

TextGenerationFlag

Especifica si el texto se muestra reflejado hacia la izquierda, boca abajo o las dos cosas a la vez.

TextString

Especifica la cadena de texto que mostrará la pantalla.

Siempre que se modifica una propiedad, debe utilizarse el método Update para ver los cambios en el dibujo.

Nota Para obtener una lista completa de los métodos y propiedades, véase la documentación del objeto Text en *AutoCAD ActiveX and VBA Reference*.

Alineación del texto de una línea

Puede justificar texto de línea horizontal y verticalmente. La alineación por defecto es la izquierda. Para establecer las opciones de alineación horizontal y vertical, utilice la propiedad Alignment.

Modificación de la alineación del texto

Este ejemplo crea un objeto Text y un objeto Point. El objeto Point se establece como el punto de alineación del texto y se cambia por un cursor en cruz rojo para que esté visible. Se cambia la alineación del texto y se muestra un cuadro de mensaje para indicar que la ejecución de la macro está detenida. Esto permite ver qué ocurre al cambiar la alineación del texto.

```
Sub Ch4_TextAlignment()
    Dim textObj As AcadText
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim height As Double
    ' Define the new Text object
    textString = "Hello, World."
    insertionPoint(0) = 3
    insertionPoint(1) = 3
    insertionPoint(2) = 0
    height = 0,5
    ' Create the Text object in model space
    Set textObj = ThisDrawing.ModelSpace. _
                    AddText(textString, insertionPoint, height)
    ' Create a point over the text alignment point,
    ' so we can better visualize the alignment process
    Dim pointObj As AcadPoint
    Dim alignmentPoint(0 To 2) As Double
    alignmentPoint(0) = 3
    alignmentPoint(1) = 3
    alignmentPoint(2) = 0
    Set pointObj = ThisDrawing.ModelSpace. _
```

```
                    AddPoint(alignmentPoint)
    pointObj.Color = acRed
    ' Set the point style to crosshair
    ThisDrawing.SetVariable "PDMODE", 2
   ' Align the text to the Left
    textObj.Alignment = acAlignmentLeft
    ThisDrawing.Regen acActiveViewport
    MsgBox "The Text object is now aligned left"
    ' Align the text to the Center
    textObj.Alignment = acAlignmentCenter
    ' Align the text to the point (necessary for
    ' all but left aligned text.)
    textObj.TextAlignmentPoint = alignmentPoint
    ThisDrawing.Regen acActiveViewport
    MsgBox "The Text object is now centered"
    ' Align the text to the Right
    textObj.Alignment = acAlignmentRight
    ThisDrawing.Regen acActiveViewport
    MsgBox "The Text object is now aligned right"
End Sub
```

Modificación del texto de una línea

Como cualquier otro objeto, los objetos Text se pueden desplazar, girar, borrar y copiar. También se puede crear simetría de texto. Si no desea que el texto figure de forma inversa al reflejarlo en simetría, asigne el valor 0 a la variable MIRRTEXT.

En la lista siguiente figuran algunos de los métodos de los que disponen los objetos Text para la edición. Para obtener una lista completa, véase la documentación sobre objetos Text en la *ActiveX and VBA Reference* de AutoCAD.

ArrayPolar

Crea una matriz polar.

ArrayRectangular

Crea una matriz rectangular.

Copy

Copia el objeto Text.

Erase

Borra el objeto Text.

Mirror

Refleja el objeto Text en simetría.

Move

Desplaza el objeto Text.

Rotate

Gira el objeto Text.

¿Comentarios?

<$nopage>texto de líneas múltiples.

Uso del texto de líneas múltiples (Mtext)

Para entradas largas y complejas, cree un texto de líneas múltiples (textoM). El texto de líneas múltiples se ajusta a una anchura determinada, si bien verticalmente su longitud puede extenderse indefinidamente. Dentro de un objeto textoM se puede asignar formato por palabras o por caracteres.

- **Para crear texto de líneas múltiples**
- **Formato del texto de líneas múltiples**

Para crear texto de líneas múltiples

Puede crear un objeto de texto de varias líneas (objeto textoM) con el método AddMText. Este método requiere tres valores de entrada: la cadena de texto, el punto de inserción del dibujo en el que colocar el texto, y la anchura del prisma de contorno.

La cadena de texto es el texto real que se desea mostrar. Se puede utilizar Unicode, código de control y caracteres especiales. El punto de inserción es una matriz de variantes con tres dobles que indican las coordenadas 3D del SCU del dibujo donde se incluirá el texto. La anchura del texto es un número positivo que representa la anchura de la caja que contiene el texto. La anchura se mide en la unidad de medida actual.

Una vez creado el objeto Text de líneas múltiples, puede aplicarle la altura del texto, la justificación, el ángulo de rotación y el estilo o bien aplicar un formato a caracteres individuales.

Para obtener una lista de los métodos y las propiedades relacionados con el objeto MText, véase la entrada sobre MText de la *ActiveX and VBA Reference*.

Para crear texto de líneas múltiples

El código siguiente crea un objeto Mtext en las coordenadas (2, 2, 0) del espacio modelo.

```
Sub Ch4_CreateMText()
    Dim mtextObj As AcadMText
    Dim insertPoint(0 To 2) As Double
    Dim width As Double
    Dim textString As String
    insertPoint(0) = 2
    insertPoint(1) = 2
```

```
    insertPoint(2) = 0
    width = 4
    textString = "This is a text string for the mtext object."
    ' Create a text Object in model space
    Set mtextObj = ThisDrawing.ModelSpace. _
                AddMText(insertPoint, width, textString)
    ZoomAll
End Sub
```

¿Comentarios?

Formato del texto de líneas múltiples

El texto nuevo adquiere de forma automática el estilo de texto activo. El estilo de texto por defecto es ESTANDAR. Puede ignorar el estilo de texto por defecto aplicando elementos de formato a caracteres individuales y aplicando propiedades al objeto Text. También se pueden indicar el formato o caracteres especiales con los métodos descritos en esta sección.

Las opciones de orientación como el estilo, la justificación, la anchura y la rotación afectan a todo el texto incluido en el contorno de un texto de líneas múltiples, y no sólo a unas palabras o caracteres determinados. Utilice la propiedad AttachmentPoint para cambiar la justificación del objeto MText y la propiedad Rotation para determinar el ángulo de rotación del contorno del texto.

La propiedad StyleName establece los tipos de letra y los aspectos de formato por defecto del nuevo texto. Conforme se crea texto, se puede seleccionar el estilo que desea utilizarse en la lista de estilos existentes. Cuando se cambia el estilo de un objeto Text de líneas múltiples con atributos de formato de caracteres asignados a cualquier parte del texto, el estilo se aplica a todo el objeto y es posible que no se conserve todo el formato de los caracteres. Por ejemplo, cambiar un estilo TrueType por un estilo que utiliza un tipo de letra SHX u otro tipo TrueType conlleva la sustitución del tipo de letra en todo el objeto de texto y la pérdida del formato particular que puedan tener asignado algunos caracteres.

Las opciones de formato como el subrayado, el texto apilado o el tipo de letra pueden aplicarse de forma individual a los caracteres o palabras que conforman un párrafo. También puede cambiar el color, el tipo de letra y la altura del texto. Puede modificar el espacio entre los caracteres de texto o aumentar la anchura de los caracteres.

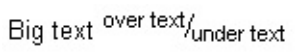
Utilice llaves ({ }) para identificar el texto al que desee aplicar el cambio de

formato. Las llaves pueden anidarse hasta alcanzar 8 niveles.

También se puede introducir el equivalente ASCII de códigos de control en las líneas o los párrafos para indicar caracteres de formato o especiales, como símbolos de tolerancia y de acotación.

Puede utilizar los caracteres de control siguientes para crear el texto de la ilustración. (Puede ver el equivalente ASCII de esta cadena en el ejemplo que sigue).

{{\H1.5x; Big text} \A2; over text\A1;/\A0; under text}



Para obtener más información acerca de dar formato a líneas múltiples, véase “Aplicación de formato a los caracteres de texto de líneas múltiples” en el *Manual del usuario*.

Utilización de caracteres de control para dar formato al texto

En este ejemplo se crea un objeto de líneas múltiples y se le asigna formato.

```
Sub Ch4_FormatMText()
    Dim mtextObj As AcadMText
    Dim insertPoint(0 To 2) As Double
    Dim width As Double
    Dim textString As String
    insertPoint(0) = 2
    insertPoint(1) = 2
    insertPoint(2) = 0
    width = 4
    ' Define the ASCII characters for the control characters
    Dim OB As Long ' Open Bracket {
    Dim CB As Long ' Close Bracket }
    Dim BS As Long ' Back Slash \
    Dim FS As Long ' Forward Slash /
    Dim SC As Long ' Semicolon ;
    OB = Asc("{")
    CB = Asc("}")
    BS = Asc("\")
    FS = Asc("/")
    SC = Asc(";")
    ' Assign the text string the following line of control
    ' characters and text characters:
    ' {{\H1.5x; Big text}\A2; over text\A1;/\A0; under text}
```

```
    textString = Chr(OB) + Chr(OB) + Chr(BS) + "H1.5x" _
    + Chr(SC) + "Big text" + Chr(CB) + Chr(BS) + "A2" _
    + Chr(SC) + "over text" + Chr(BS) + "A1" + Chr(SC) _
    + Chr(FS) + Chr(BS) + "A0" + Chr(SC) + "under text" _
    + Chr(CB)
    ' Create a text Object in model space
    Set mtextObj = ThisDrawing.ModelSpace. _
                AddMText(insertPoint, width, textString)
    ZoomAll
End Sub
```

¿Comentarios?

Uso de caracteres Unicode, códigos de control y caracteres especiales

Puede representar símbolos en las cadenas de texto mediante caracteres Unicode, códigos de control y caracteres especiales. (Todos los caracteres que no sean texto deben especificarse mediante su equivalente ASCII).

Para crear caracteres especiales, se puede introducir las siguientes cadenas de caracteres Unicode:

Descripción de los caracteres Unicode

Carácter Unicode	Descripción
\U+00B0	Símbolo de grado
\U+00B1	Símbolo de tolerancia más/menos
\U+2205	Símbolo de cota de diámetro

Los caracteres especiales se pueden especificar, además de con caracteres Unicode, incluyendo información de control en la cadena de texto. Utilice un par de signos de porcentaje (%%) delante de cada secuencia de control. Por ejemplo, el siguiente código de control utiliza un texto de AutoCAD estándar y tipos de

letra PostScript para dibujar el número de carácter *nnn*:

```
%%nnn
```

En una cadena de texto de VB o VBA, el ejemplo anterior se escribiría como:

```
Dim percent as Long
percent = ASC("%")
TextString = chr(percent) + chr(percent) + "nnn"
```

Estos códigos de control sólo funcionan con los tipos de letra de texto estándar de AutoCAD:

Descripción de los códigos de control

Código de control	Descripción
%%o	Activa y desactiva el modo de suprarrayado.
%%u	Activa y desactiva el modo de subrayado.
%%d	Dibuja el símbolo de grados.
%%p	Dibuja un símbolo de tolerancia más/menos.
%%c	Dibuja el símbolo de cota de diámetro.
%%%	Dibuja el signo de porcentaje (uno solo).

Sustitución de tipos de letra

Puede designar tipos de letra para que sean sustituidos por otros o para que sirvan como tipos de letra por defecto cuando AutoCAD no pueda encontrar un tipo de letra especificado en un dibujo.

Si AutoCAD no puede localizar un tipo de letra especificado en un dibujo, se pueden designar qué tipos de letras determinados se sustituyen por otros o por los tipos por defecto.

Los tipos de letra empleados en el texto de los dibujos están determinados por el estilo del texto y, en el caso de texto de líneas múltiples, por los formatos de tipos de letra individuales aplicados a determinadas partes del texto. Se pueden utilizar tablas de asignación de tipos de letra para asegurarse de que el dibujo sólo emplea determinados tipos de letra, o para convertir los tipos de letra empleados en un tipo de letra distinto. AutoCAD incorpora una tabla de asignación de tipos de letra por defecto. Este archivo puede editarse utilizando un editor de texto ASCII. Se pueden especificar otras tablas de asignación de tipos mediante la propiedad FontFileMap del objeto Preferences.

Para obtener más información acerca de las tablas de asignación de tipos de letra y la sustitución de tipos, véase “Tipos de letra alternativos” en el *Manual del usuario*.

- **Definición de un tipo de letra alternativo por defecto**

Definición de un tipo de letra alternativo por defecto

Si el dibujo especifica un tipo de letra que no figura en el sistema, AutoCAD procede a su sustitución por el tipo designado a tal efecto. AutoCAD utiliza por defecto el archivo *simplex.shx*. No obstante, si lo necesita, puede indicar otro tipo de letra. Utilice la propiedad AltFontFile del objeto Preferences cuando desee definir el nombre del archivo de tipos de letra alternativos.

Si utiliza una estilo de texto que emplea un tipo de letra grande, puede asignarlo a otro tipo de letra con la propiedad AltFontFile. Esta variable de sistema utiliza dos archivos de tipos de letra por defecto, *txt.shx* y *bigfont.shx*.

Si AutoCAD no puede abrir un archivo de fuente al abrir un dibujo, aplica un conjunto por defecto de reglas de sustitución.

Corrección ortográfica

Durante la comprobación ortográfica, AutoCAD establece correspondencias entre palabras del dibujo y las del diccionario principal de ese momento. Las palabras que añada se almacenarán en el diccionario personalizado que se encuentre en uso en el momento de la verificación ortográfica. Por ejemplo, puede añadir nombres propios para que AutoCAD los reconozca.

Si desea comprobar la ortografía de un texto redactado en otro idioma, utilice el diccionario principal correspondiente.

ActiveX Automation de AutoCAD no dispone de ningún método de revisión ortografía. No obstante, se puede especificar un diccionario principal distinto con la propiedad MainDictionary, u otro diccionario personalizado mediante la propiedad CustomDictionary del objeto Preferences.

Para obtener más información acerca de la comprobación ortográfica, véase “Corrección ortográfica” en el *Manual del usuario*.

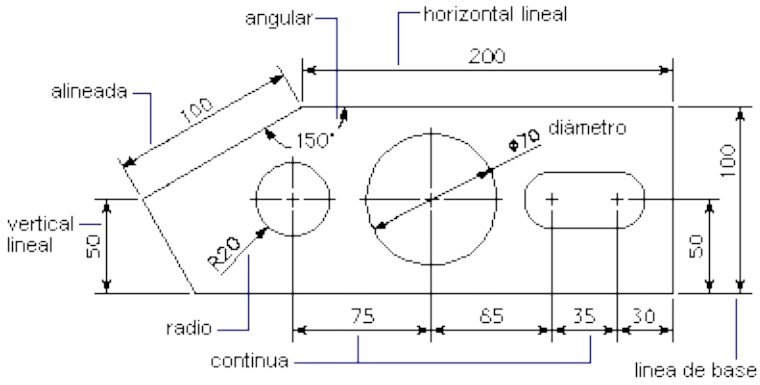
Cotas y tolerancias

Las cotas agregan medidas a los dibujos. Las tolerancias establecen la variación posible de una cota. Mediante ActiveX Automation se pueden gestionar cotas con estilos y modificaciones de cota.

- **Conceptos sobre cotas**
- **Creación de cotas**
- **Edición de cotas**
- **Trabajo con estilos de cota**
- **Acotación en espacio modelo y en espacio papel**
- **Creación de directrices y anotaciones**
- **Uso de tolerancias geométricas**

Conceptos sobre cotas

Las cotas indican las medidas geométricas de objetos, las distancias o ángulos entre objetos o las coordenadas *X* y *Y* de una característica. AutoCAD® ofrece tres tipos básicos de acotación: lineal, radial y angular. Las cotas lineales incluyen cotas alineadas, rotadas y de coordenadas.



Se pueden acotar líneas, líneas múltiples, arcos, círculos y segmentos de polilínea, o bien crear cotas independientes.

AutoCAD dibuja las cotas en la capa activa. Todas las cotas tienen un estilo asociado, que puede ser el establecido por defecto o uno definido por el usuario. El estilo controla aspectos como el color, el estilo de texto y la escala de tipos de línea. Los datos de altura del objeto no se consideran. La familia de estilos permite realizar ligeras modificaciones en los diferentes tipos de cotas a partir de un estilo base. Las sustituciones, por su parte, permiten modificar el estilo de cotas determinadas.

Para obtener más información sobre cotas, véase “Modificación de objetos existentes” en el *Manual del usuario*.

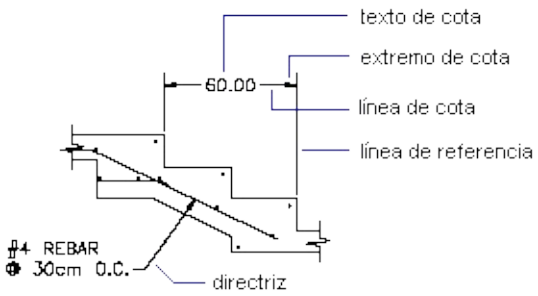
- **Partes de una cota**
- **Definición de variables de sistema para la acotación.**
- **Definición de estilos del texto de cota**

- **Conceptos básicos de las líneas directrices**
- **Conceptos básicos de las cotas asociativas**

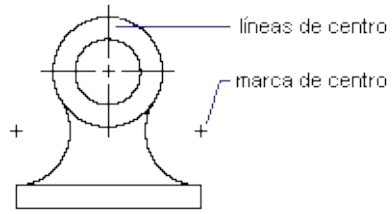
¿Comentarios?

Partes de una cota

Esta sección describe brevemente las partes de una cota



Una línea de cota consiste en una línea que indica la dirección y la extensión de una cota. En las cotas angulares, la línea de cota es un arco. Las líneas de referencia, también llamadas líneas de extensión o de proyección, se extienden desde la característica que se está acotando hasta la línea de cota. Los extremos de cota, también llamados símbolos de terminación o finalización, se añaden en cada extremo de la línea de cota. El texto de cota es una cadena de texto que normalmente indica la medida actual. Un texto también puede incluir prefijos, sufijos y tolerancias. Una directriz es una línea continua que une una anotación y la característica a la que hace referencia. Una marca de centro es una pequeña cruz que indica el centro de un círculo o de un arco. Las líneas de centro son líneas discontinuas que indican el centro de un círculo o de un arco.



Para obtener más información acerca de las partes de una cota, véase “Partes de una cota” en el *Manual del usuario*.

¿Comentarios?

Definición de variables de sistema para la acotación.

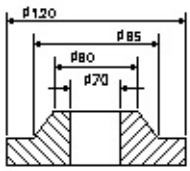
Las variables del sistema controlan la apariencia de las cotas. Las variables del sistema de cotas incluyen DIMAUNIT, DIMUPT, DIMTOFL, DIMFIT, DIMTIH, DIMTOH, DIMJUST, y DIMTAD. Puede definir estas variables utilizando el método SetVariable Por ejemplo, la siguiente línea de código establece la variable de sistema DIMAUNIT (el formato de las unidades de las cotas angulares) en radianes (3):

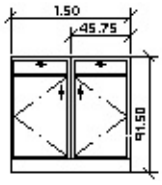
```
ThisDrawing.SetVariable "ACOUNANG", 3
```

Para obtener información acerca de las variables de sistema para la acotación, véase “Utilización de estilos de cotas” en el *Manual del usuario*.

Definición de estilos del texto de cota

El texto de cota es una cadena de caracteres asociada a una cota, que puede incluir medidas, tolerancias (laterales y geométricas), prefijos, sufijos y notas de texto en una línea o en un párrafo. El texto de cota puede incluir la medida que AutoCAD calcula por defecto o cualquier texto que proporcione el usuario; también se puede suprimir todo el texto de cota. El texto de cota puede utilizarse para agregar información, ya sean procedimientos especiales de fabricación o indicaciones de ensamblaje.



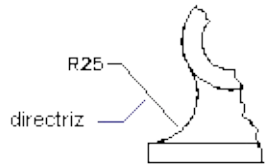


El texto de cota en una línea utiliza el estilo de texto activo que designa la propiedad ActiveTextStyle Los párrafos de texto utilizan el estilo de texto activo con todas las modificaciones que se realicen en la cadena de texto.

Para obtener más información acerca del texto de cota, véase “Control del texto de la cota” en el *Manual del usuario*.

Conceptos básicos de las líneas directrices

Una línea directriz por defecto es una línea recta con un extremo de cota (cabeza de flecha) que hace referencia a una característica de un dibujo. Por regla general, la función de una directriz es conectar la anotación con la característica anotada. En este caso, anotación significa texto de párrafo, bloques o marcos de control de características. Estas líneas directrices son diferentes de las líneas directrices sencillas que AutoCAD crea automáticamente para las cotas radiales, de diámetro y lineales, cuyo texto no se ajusta entre las líneas de referencia.



Los objetos directrices se asocian a una anotación, por lo que al editar la anotación la directriz se actualiza en consonancia. Se puede copiar una anotación de cualquier punto de un dibujo y añadirla a una directriz o crear una anotación nueva. También se pueden crear directrices sin anotaciones.

Para obtener más información acerca de las directrices, véase “Introducción a la creación de texto y directrices” en el *Manual del usuario*.

Conceptos básicos de las cotas asociativas

Las cotas asociativas ajustan automáticamente sus ubicaciones, orientaciones y medidas cuando se modifican los objetos geométricos asociados con ellas. La variable de sistema DIMASSOC controla las cotas asociativas. Establezca DIMASSOC en 2 para activar las cotas asociativas.

Para obtener más información sobre cotas asociativas, véase “Cotas asociativas” en el *Manual del usuario*.

Creación de cotas

Puede crear cotas lineales, radiales, angulares y de coordenadas.

Cuando se crean cotas, se utiliza el estilo de cota activo. Una vez creada la línea de referencia, puede modificar su origen, su ubicación y el contenido del texto de cota, así como su ángulo con respecto a la línea de cota. También se puede modificar el estilo que utiliza la cota.

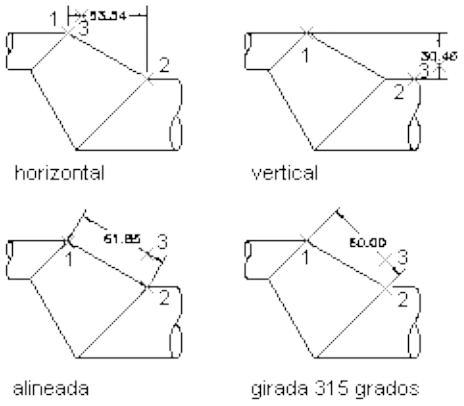
Para obtener más información acerca de la creación de cotas, véase "Modificación de objetos existentes" en el *Manual del usuario*.

- **Creación de cotas lineales**
- **Creación de cotas radiales**
- **Creación de cotas angulares**
- **Creación de cotas por coordenadas**

Creación de cotas lineales

Las cotas lineales pueden estar alineadas o giradas. Las cotas alineadas tienen la línea de cota paralela a la línea donde se encuentre el origen de la línea de referencia. En el caso de las cotas giradas, la línea de cota se coloca en ángulo con respecto al origen de la línea de referencia.

Para crear una cota lineal, utilice el método AddDimAligned o AddDimRotated. Una vez creadas las cotas lineales se puede modificar el texto, el ángulo del texto y el ángulo de la línea de cota. En las siguientes figuras, se designa de forma explícita el origen de las líneas de referencia. Se muestra la ubicación final de la línea de cota:



Para crear una cota alineada utilice el método AddDimAligned. Este método requiere tres coordenadas de entrada: el origen de ambas líneas de cota y la posición del texto.

Para crear una cota girada, utilice el método AddDimRotated. Este método requiere la entrada de tres coordenadas y del ángulo de la línea de cota. Las tres coordenadas son el origen de ambas líneas de referencia y la posición del texto. El ángulo debe proporcionarse en radianes y representa el ángulo de rotación de la línea de cota.

Para obtener más información acerca de la creación de cotas lineales, véase “Creación de cotas lineales” en el *Manual del usuario*.

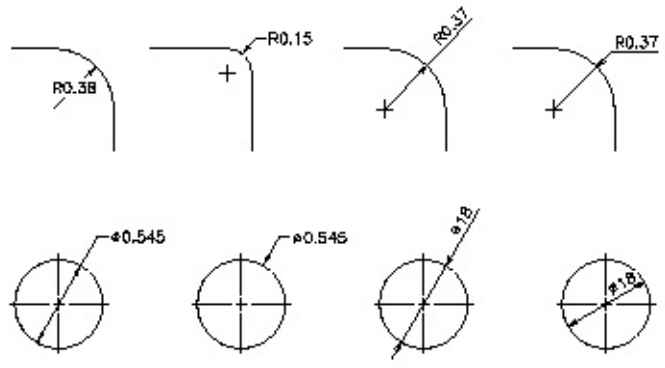
¿Comentarios?

Creación de cotas radiales

Las cotas radiales miden los radios y los diámetros de arcos y círculos. Para crear una cota radial, utilice el método AddDimRadial.

Se crean tipos de cotas distintos según el tamaño del círculo del arco, la propiedad TextPosition y los valores de las variables del sistema de cotas DIMUPT, DIMTOFL, DIMFIT, DIMTIH, DIMTOH, DIMJUST y DIMTAD. (Las variables del sistema se pueden consultar o configurar con los métodosGetVariable y SetVariable

Para el texto de cotas horizontales, si el ángulo de la línea de cota radial con respecto a la horizontal tiene más de 15 grados y está fuera del círculo o arco, AutoCAD dibuja una línea de conexión, también llamada de quiebro o aterrizaje. La línea de conexión tiene la longitud de un extremo de cota y se ubica junto al texto de la cota, como muestran las siguientes ilustraciones:



Para crear cotas radiales, utilice los métodos AddDimRadial o AddDimDiametric. Estos métodos requieren tres valores de entrada: la coordenada del círculo o del centro del arco, la coordenada para el enlace de la directriz y la longitud de la directriz.

Estos métodos utilizan el parámetro `LeaderLength` como distancia desde el

punto de cuerda hasta el punto donde la cota realiza una línea de conexión horizontal al texto de anotación (o se detiene si la línea de conexión no es necesaria).

Para obtener más información acerca de la creación de cotas lineales, véase “Creación de cotas radiales” en el *Manual del usuario*.

Creación de una cota radial

Este ejemplo crea una cota radial en espacio modelo.

```
Sub Ch5_CreateRadialDimension()
    Dim dimObj As AcadDimRadial
    Dim center(0 To 2) As Double
    Dim chordPoint(0 To 2) As Double
    Dim leaderLen As Integer
    ' Define the dimension
    center(0) = 0
    center(1) = 0
    center(2) = 0
    chordPoint(0) = 5
    chordPoint(1) = 5
    chordPoint(2) = 0
    leaderLen = 5
    ' Create the radial dimension in model space
    Set dimObj = ThisDrawing.ModelSpace. _
           AddDimRadial(center, chordPoint, leaderLen)
    ZoomAll
End Sub
```

Nota El parámetro LeaderLength sólo se utiliza en la creación de las cotas, y únicamente si está definido con el valor por defecto de posición del texto. Cuando la cota se cierra por primera vez, el cambio del valor de LeaderLength no afecta a la presentación de la cota, pero el nuevo parámetro queda almacenado y se muestra en las aplicaciones DXF, LISP y ADSRX.

Creación de cotas angulares

Las cotas angulares miden el ángulo formado por dos líneas o tres puntos. Se pueden utilizar, por ejemplo, para medir el ángulo que forman dos radios de un círculo. La línea de cota forma un arco.

Para crear una cota angular, utilice el método AddDimAngular. Este método requiere tres valores de entrada: el vértice del ángulo, los orígenes de las líneas de referencia y la ubicación del texto. El AngleVertex es el centro del círculo o arco, o el vértice común a las dos líneas que se están acotando. El origen de las líneas de referencia está en los puntos por los que pasan ambas líneas.

El AngleVertex puede coincidir con uno de los puntos del origen. Las líneas de referencia se añaden de forma automática cuando se necesitan.

Para obtener más información acerca de la creación de cotas angulares, véase “Creación de cotas angulares” en el *Manual del usuario*.

Creación de una cota angular

Este ejemplo crea una cota angular en espacio modelo.

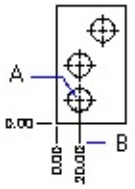
```
Sub Ch5_CreateAngularDimension()
    Dim dimObj As AcadDimAngular
    Dim angVert(0 To 2) As Double
    Dim FirstPoint(0 To 2) As Double
    Dim SecondPoint(0 To 2) As Double
    Dim TextPoint(0 To 2) As Double
    ' Define the dimension
    angVert(0) = 0
    angVert(1) = 5
    angVert(2) = 0
    FirstPoint(0) = 1
    FirstPoint(1) = 7
    FirstPoint(2) = 0
```

```
    SecondPoint(0) = 1
    SecondPoint(1) = 3
    SecondPoint(2) = 0
    TextPoint(0) = 3
    TextPoint(1) = 5
    TextPoint(2) = 0
    ' Create the angular dimension in model space
    Set dimObj = ThisDrawing.ModelSpace. _
     AddDimAngular(angVert, FirstPoint, SecondPoint, TextPoint)
    ZoomAll
End Sub
```

¿Comentarios?

Creación de cotas por coordenadas

Las cotas de coordenadas, o de referencia, miden la distancia perpendicular desde un punto de origen llamado punto de referencia hasta una característica acotada como, por ejemplo, el agujero de una pieza. La acotación por coordenadas evita los errores de escala, al mantener desfases exactos de los objetos con respecto al punto de referencia.



Las cotas de coordenadas están formadas por las coordenadas *X* o *Y* y una línea directriz. Las cotas de coordenadas de referencia *X* indican la distancia de un elemento desde la cota de referencia a lo largo del eje *X*. Las cotas de coordenadas de referencia *Y* miden la misma distancia a lo largo del eje *Y*. AutoCAD utiliza el origen del sistema de coordenadas personales (SCP) actual para determinar las coordenadas medidas. Se utiliza el valor absoluto de las coordenadas.

El texto se alinea con la línea directriz de la coordenada sin considerar la orientación de texto definida en el estilo de cota activo. Se puede aceptar el texto por defecto o escribir uno nuevo.

Para crear una cota de coordenadas, utilice el método AddDimOrdinate. Este método requiere tres valores de entrada: una coordenada que determine el punto que se desea acotar (A), una coordenada que determine el final de la línea directriz (B) y una señal booleana que determine si es una cota de coordenada de referencia *X* o *Y*. Si introduce TRUE como valor de la señal booleana, el método creará una cota de coordenada de referencia *X*. Si introduce FALSE, creará una cota de coordenada de referencia *Y*.

Para obtener más información acerca de la creación de cotas por coordenadas, véase “Creación de cotas por coordenadas” en el *Manual del usuario*.

Creación de una cota de coordenadas

Este ejemplo crea una cota de coordenadas en espacio modelo.

```
Sub Ch5_CreatingOrdinateDimension()
    Dim dimObj As AcadDimOrdinate
    Dim definingPoint(0 To 2) As Double
    Dim leaderEndPoint(0 To 2) As Double
    Dim useXAxis As Long
    ' Define the dimension
    definingPoint(0) = 5
    definingPoint(1) = 5
    definingPoint(2) = 0
    leaderEndPoint(0) = 10
    leaderEndPoint(1) = 5
    leaderEndPoint(2) = 0
    useXAxis = 5
    ' Create an ordinate dimension in model space
    Set dimObj = ThisDrawing.ModelSpace. _
                 AddDimOrdinate(definingPoint, _
                 leaderEndPoint, useXAxis)
    ZoomAll
End Sub
```

Edición de cotas

Al igual que ocurre con otros objetos gráficos de AutoCAD, las cotas pueden modificarse por medio de los métodos y propiedades estándar propios del objeto.

La mayoría de los objetos de acotación disponen de las siguientes propiedades:

Rotation

Determina el ángulo de rotación de la línea de cota, en radianes.

StyleName

Designa el nombre del estilo de acotación.

TextOverride

Determina la cadena de texto de la cota.

TextPosition

Determina la posición del texto de cota.

TextRotation

Determina el ángulo de rotación del texto de cota.

Measurement

Designa la medida real de la cota.

Algunos objetos de acotación también cuentan con propiedades para editar el origen de las líneas de referencias y la longitud de las directrices.

En cuanto a la edición de los objetos de acotación, se incluyen los siguientes métodos:

ArrayPolar

Crea una matriz polar.

ArrayRectangular

Crea una matriz rectangular.

Copy

Copia el objeto de cota.

Erase

Borra el objeto de cota.

Mirror

Refleja el objeto de cota.

Move

Mueve el objeto de cota.

Rotate

Gira el objeto de cota.

ScaleEntity

Cambia la escala del objeto de cota.

Para obtener más información acerca de la modificación de cotas, véase "Modificación de cotas existentes" en el *Manual del usuario*.

- **Sustitución del texto de cota**

Sustitución del texto de cota

El valor de cota mostrado puede reemplazarse mediante la propiedad TextOverride. Esta propiedad permite reemplazar completamente el valor mostrado de la cota o añadirle texto.

Modificación del texto de cota

Este ejemplo añade texto al valor para que se muestren tanto la cadena como el valor de la cota.

```
Sub Ch5_OverrideDimensionText()
    Dim dimObj As AcadDimAligned
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    Dim location(0 To 2) As Double
    ' Define the dimension
    point1(0) = 5#: point1(1) = 3#: point1(2) = 0#
    point2(0) = 10#: point2(1) = 3#: point2(2) = 0#
    location(0) = 7.5: location(1) = 5#: location(2) = 0#
    ' Create an aligned dimension object in model space
    Set dimObj = ThisDrawing.ModelSpace. _
                    AddDimAligned(point1, point2, location)
    ' Change the text string for the dimension
    dimObj.TextOverride = "The value is <>"
    dimObj.Update
End Sub
```

Trabajo con estilos de cota

Un estilo de acotación es un conjunto de parámetros que determinan el aspecto de una cota. Mediante los estilos de cota memorizados, el usuario puede establecer e imponer estándares para los bocetos.

Todas las cotas que se crean utilizan el estilo de texto activo. Si antes de crear una cota no se define ni se aplica ningún estilo, AutoCAD aplica el estilo por defecto STANDARD. Para establecer el estilo de acotación activo, utilice la propiedad ActiveDimStyle

Para establecer un estilo de cota superior, primero se debe guardar y asignar nombre a un estilo. En el momento de crearse, el nuevo estilo es una copia del actual, pero posteriormente se incluyen todas las modificaciones realizadas en la presentación de las piezas acotadas, en la ubicación del texto de cota y en el aspecto de la anotación. En este caso, la anotación incluye unidades principales y alternativas, tolerancias y texto.

Para obtener más información acerca de los estilos de cota, véase “Utilización de estilos de cotas” en el *Manual del usuario*.

- **Creación, modificación y copia de estilos de cota**
- **Sustitución del estilo de cota**

Creación, modificación y copia de estilos de cota

Para crear un estilo de cota nuevo, utilice el método Add. Este método requiere la entrada del nombre del nuevo estilo de cota.

ActiveX Automation de AutoCAD permite agregar nuevos estilos de cota y modificar el estilo de cota activo. También se puede cambiar el estilo de cota asociado a cada cota, con el método StyleName

Así mismo, se puede copiar un estilo o un conjunto de valores ya existentes. Utilice el método CopyFrom para copiar un estilo de cota de un objeto de origen en un nuevo estilo de cota. El objeto de origen puede ser otro objeto DimStyle, una cota, un objeto Tolerance o Leader, o incluso un objeto Document Cuando se copian los parámetros de estilo de otra cota, el estilo se duplica con exactitud. Si los parámetros de estilo se copian de una cota o de un objeto Tolerance o Leader, en el nuevo estilo se copian los parámetros actuales, incluidas las modificaciones del objeto. Si se copia el estilo de un objeto Document, en el nuevo estilo se copian el estilo de cota activo y todas las modificaciones del dibujo.

Copia de estilos de cota y modificaciones

En este ejemplo se crean tres estilos de cota nuevos y se copian respectivamente en cada uno los parámetros actuales del documento, un estilo de cota determinado y una cota concreta. Si sigue el proceso de configuración adecuado antes de ejecutar este código de ejemplo, comprobará que se crean varios estilos de cota diferentes.

1. Cree un nuevo dibujo y selecciónelo para que sea el dibujo activo.
2. Cree una cota lineal en el nuevo dibujo. La cota debe ser el único objeto del dibujo.

3. Cambie el color de la línea de cota a amarillo.
4. Cambie la variable de sistema DIMCLRD a 5 (azul).
5. Ejecute el siguiente ejemplo:

```
Sub Ch5_CopyDimStyles()
    Dim newStyle1 As AcadDimStyle
    Dim newStyle2 As AcadDimStyle
    Dim newStyle3 As AcadDimStyle
    Set newStyle1 = ThisDrawing.DimStyles.Add _
                    ("Style 1 copied from a dim")
    Call newStyle1.CopyFrom(ThisDrawing.ModelSpace(0))
    Set newStyle2 = ThisDrawing.DimStyles.Add _
                    ("Style 2 copied from Style 1")
    Call newStyle2.CopyFrom(ThisDrawing.DimStyles.Item _
                    ("Style 1 copied from a dim"))
    Set newStyle2 = ThisDrawing.DimStyles.Add _
                    ("Style 3 copied from the running drawing values
    Call newStyle2.CopyFrom(ThisDrawing)
End Sub
```

Abra el cuadro de diálogo DIMSTYLE. Ahora debería tener tres estilos de cota en la lista. El estilo 1 debe tener una línea de cota amarilla. El estilo 2 deber ser igual al estilo 1, y el estilo 3 debe tener la línea de cota azul.

Sustitución del estilo de cota

En todas las cotas es posible ignorar los parámetros de su estilo de acotación. La mayoría de los objetos de acotación disponen de las siguientes propiedades:

AltRoundDistance

Especifica el redondeo de las unidades alternativas.

AngleFormat

Establece el formato de las unidades de las cotas angulares.

Arrowhead1Block, Arrowhead2Block

Designa el bloque que se utiliza como extremo de cota personalizado de las líneas de cota.

Arrowhead1Type, Arrowhead2Type

Designa el tipo de extremo de cota de las líneas de cota.

ArrowheadSize

Especifica el tamaño de los extremos de cota de las líneas de cota, los extremos de cota de las líneas directrices y las líneas de conexión

CenterMarkSize

Especifica el tamaño de la marca central de las cotas radiales y de diámetro.

CenterType

Especifica el tipo de la marca central de las cotas radiales y de diámetro.

DecimalSeparator

Especifica el carácter utilizado como separador decimal en los valores de cota y de tolerancia decimales.

DimensionLineColor

Determina el color de la línea de cota de un objeto de cota, directriz o tolerancia.

DimensionLineWeight

Determina el grosor de las líneas de cota.

DimLine1Suppress, DimLine2Suppress

Designan la supresión de las líneas de cota.

DimLineInside

Especifica la presentación de líneas de cota solamente dentro de las líneas de referencia.

ExtensionLineColor

Designa el color de las líneas de referencia de cota.

ExtensionLineExtend

Especifica la distancia de la línea de referencia que sobrepasa a la línea de cota.

ExtensionLineOffset

Determina la distancia que se desvían las líneas de referencia con respecto a sus puntos de origen.

ExtensionLineWeight

Determina el grosor de las líneas de referencia.

ExtLine1EndPoint, ExtLine2EndPoint

Determina el punto final de las líneas de referencia.

ExtLine1StartPoint, ExtLine2StartPoint

Especifica el punto inicial de las líneas de referencia.

ExtLine1Suppress, ExtLine2Suppress

Determina la supresión de las líneas de referencia.

Ajustar

Especifica la ubicación del texto y los extremos de cota fuera o dentro de las líneas de referencia.

ForceLineInside

Determina si una línea de cota se dibuja entre las líneas de referencia, incluso cuando el texto está situado fuera de dichas líneas.

FractionFormat

Especifica el formato de los valores fraccionarios en las cotas y las tolerancias.

HorizontalTextPosition

Designa la justificación horizontal del texto de la cota.

LinearScaleFactor

Especifica un factor de escala global para la medición de cotas lineales.

PrimaryUnitsPrecision

Designa el número de posiciones decimales mostradas para las unidades principales de una cota o una tolerancia.

SuppressLeadingZeros, SuppressTrailingZeros

Especifica la supresión de los ceros a la derecha y a la izquierda en los valores de las cotas.

SuppressZeroFeet, SuppressZeroInches

Especifica la supresión de una medida de cero pies y cero pulgadas en los valores de las cotas.

TextColor

Designa el color del texto de los objetos de cota y de tolerancia.

TextGap

Especifica la distancia entre el texto de la cota y la línea de cota cuando se divide la línea para incluir el texto.

TextHeight

Determina la altura del texto de la cota o la tolerancia.

TextInside

Especifica si el texto de la cota debe dibujarse dentro de las líneas de referencia.

TextInsideAlign

Especifica la posición del texto de la cota dentro de las líneas de referencia para todos los tipos de acotaciones excepto las de coordenadas.

TextMovement

Indica cómo se dibuja el texto de la cota cuando el texto se desplaza.

TextOutsideAlign

Especifica la posición del texto de la cota fuera de las líneas de referencia para todos los tipos de acotaciones excepto las de coordenadas.

TextPosition

Determina la posición del texto de cota.

TextPrecision

Indica la precisión del texto de las cotas angulares.

TextPrefix

Designa el prefijo del valor de la cota.

TextRotation

Determina el ángulo de rotación del texto de cota.

TextSuffix

Designa el sufijo del valor de la cota.

ToleranceDisplay

Especifica si se muestran las tolerancias junto con el texto de la cota.

ToleranceHeightScale

Especifica un factor de escala para la altura del texto de los valores de tolerancia con respecto a la altura del texto de la cota.

ToleranceJustification

Designa la justificación vertical de los valores de tolerancia con respecto al texto de cota nominal.

ToleranceLowerLimit

Especifica el límite de tolerancia mínima del texto de las cotas.

TolerancePrecision

Indica la precisión de los valores de tolerancia en las cotas principales.

ToleranceSuppressLeadingZeros

Especifica la supresión de los ceros a la izquierda en los valores de tolerancia.

ToleranceSuppressTrailingZeros

Especifica la supresión de los ceros a la derecha en los valores de cota.

ToleranceUpperLimit

Especifica el límite de tolerancia máxima del texto de las cotas.

UnitsFormat

Designa el formato de las unidades de las todas las cotas excepto las angulares.

VerticalTextPosition

Designa la posición vertical del texto con respecto a la línea de cota.

Introducción de un sufijo definido por el usuario para utilizarlo en una cota alineada

Este ejemplo crea una cota alineada en espacio modelo y utiliza la propiedad TextSuffix para permitir que el usuario cambie el sufijo del texto de la cota.

```
Sub Ch5_AddTextSuffix()
    Dim dimObj As AcadDimAligned
    Dim point1(0 To 2) As Double
    Dim point2(0 To 2) As Double
    Dim location(0 To 2) As Double
    Dim suffix As String
    ' Define the dimension
    point1(0) = 1.3: point1(1) = 7.8: point1(2) = 0
```

```
    point2(0) = 5: point2(1) = 5: point2(2) = 0
    location(0) = 5: location(1) = 7: location(2) = 0
    ' Create an aligned dimension object in model space
    Set dimObj = ThisDrawing.ModelSpace. _
                 AddDimAligned(point1, point2, location)
    ThisDrawing.Application.ZoomAll
    ' Allow the user to change the text suffix for the dimension
    suffix = InputBox("Enter a new text suffix for the dimension" _
                      , "Set Dimension Suffix", ":SUFFIX")
    ' Apply the change to the dimension
    dimObj.TextSuffix = suffix
    ThisDrawing.Regen acAllViewports
End Sub
```

¿Comentarios?

Acotación en espacio modelo y en espacio papel

Se pueden dibujar cotas tanto en espacio papel como en espacio modelo. No obstante, si la geometría que se está acotando está en el espacio modelo, conviene dibujar las cotas en dicho espacio, ya que AutoCAD sitúa los puntos de definición de las cotas en el espacio donde está dibujada la geometría.

Cuando se dibuja en espacio papel una cota que describe la geometría de un modelo, la cota no cambia cuando se utilizan comandos de edición o se modifica el factor de ampliación de la ventana gráfica del espacio modelo. Tampoco se altera la posición de las cotas de espacio papel cuando se cambia una vista del espacio papel al espacio modelo.

Si dibuja cotas en el espacio papel y el factor de escala global para acotación lineal (valor de la variable de sistema DIMLFAC) es menor que cero, la distancia medida se multiplica por el valor absoluto de DIMLFAC. Si se dibujan cotas en el espacio modelo, se utiliza el valor 1.0 aunque DIMLFAC sea menor que cero. AutoCAD calcula un valor para DIMLFAC si modifica la variable en la solicitud Acotar y selecciona la opción Ventana. AutoCAD calcula la escala de espacio modelo a espacio papel y asigna este valor con signo negativo a la variable DIMLFAC.

Creación de directrices y anotaciones

Una directriz es una línea que conecta anotaciones con una característica de un dibujo. Las directrices y sus anotaciones tienen carácter asociativo, por lo que al modificar una anotación su directriz se actualiza consecuentemente. El objeto Leader no se debe confundir con las líneas directrices que AutoCAD genera automáticamente como parte de una línea de cota.

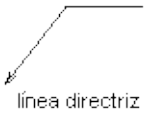
Para obtener más información acerca de las directrices, véase “Creación de texto con directrices” en el *Manual del usuario*.

- **Creación de líneas directrices**
- **Adición de una anotación a una línea directriz**
- **Asociatividad de las directrices**
- **Modificación de la asociatividad de las directrices**
- **Edición de directrices**

Creación de líneas directrices

Puede crear una línea directriz desde cualquier punto o característica de un dibujo y controlar los parámetros que determinan su aspecto durante el proceso de creación. Una línea directriz puede consistir en una serie de segmentos de línea recta o en una suave curva spline. El color de la directriz se controla mediante el color de la línea de cota actual. La escala de la directriz se controla con la escala general de cotas que tenga definida el estilo de cota activo. El tipo y tamaño de los extremos de cotas (flechas), si se designa alguno, se controla mediante el primer extremo que se haya definido en el estilo activo.

Por lo general, la directriz y la anotación se conectan mediante una pequeña línea conocida como línea de conexión. Las líneas de conexión aparecen con cuadros de control de características y MText cuando el último segmento de la directriz tiene un ángulo mayor de 15 grados desde la horizontal. La longitud de la línea de conexión es igual a la de una sola flecha. Cuando la directriz no tiene anotaciones, tampoco cuenta con línea de conexión.



Para crear una línea directriz, utilice el método AddLeader. Este método requiere tres valores de entrada: la matriz de coordenadas que especifican dónde crear la directriz, el objeto de anotación (o `NULL` si la línea directriz no tiene anotación, y el tipo de directriz que se va a crear. El tipo determina si la directriz será una línea recta o una curva spline suave. También determina si tendrá flechas o no. Use una de las constantes siguientes para especificar el tipo de directriz: `acLineNoArrow`, `acLineWithArrow`, `acSplineNoArrow`, o `acSplineWithArrow`. Estas constantes se excluyen mutuamente.

Creación de una línea directriz

Este ejemplo crea una línea directriz en espacio modelo. La directriz no tiene asociadas anotaciones.

```
Sub Ch5_CreateLeader()
    Dim leaderObj As AcadLeader
    Dim points(0 To 9) As Double
    Dim leaderType As Integer
    Dim annotationObject As AcadObject
    points(0) = 0: points(1) = 0: points(2) = 0
    points(3) = 4: points(4) = 4: points(5) = 0
    points(6) = 4: points(7) = 5: points(8) = 0
    leaderType = acLineWithArrow
    Set annotationObject = Nothing
    ' Create the leader object in model space
    Set leaderObj = ThisDrawing.ModelSpace. _
          AddLeader(points, annotationObject, leaderType)
    ZoomAll
End Sub
```

Adición de una anotación a una línea directriz

Una anotación de directriz puede ser un objeto Tolerance, MText o BlockRef. Se puede crear una anotación nueva o utilizar la copia de una ya existente, pero sólo se añade a la directriz una vez creada.

Para añadir una anotación mientras se crea una directriz, introduzca la nota en el método AddLeader.

Asociatividad de las directrices

Debido a que las directrices se asocian con su anotación, cuando la anotación se mueve el punto final de la directriz se mueve con ella. Conforme se desplaza el rectángulo de tolerancia y el texto de la anotación, el segmento final de la línea directriz alterna la conexión entre las caras izquierda y derecha de la anotación, según la relación de la anotación con el penúltimo punto (el segundo por el final) de la directriz. Si el punto medio de la anotación se encuentra a la derecha del penúltimo punto de la directriz, ésta se enlaza a la derecha; de lo contrario, el enlace se realiza a la izquierda.

Cuando se elimina un objeto del dibujo mediante los métodos Erase, Add (para agregar un bloque) o WBlock, se rompe la asociatividad. Cuando la directriz y su anotación se copian juntas en una misma operación, la nueva copia es asociativa. No es asociativa si se copian por separado. Si la asociatividad se rompe, por ejemplo por copiar sólo el objeto Leader o borrar la anotación, la línea de conexión de la directriz desaparece.

Asociación de una línea directriz a una anotación

En este ejemplo se crea un objeto MText. Después se crea una línea directriz utilizando el objeto MText como anotación.

```
Sub Ch5_AddAnnotation()
    Dim leaderObj As AcadLeader
    Dim mtextObj As AcadMText
    Dim points(0 To 9) As Double
    Dim insertionPoint(0 To 2) As Double
    Dim width As Double
    Dim leaderType As Integer
    Dim annotationObject As Object
    Dim textString As String, msg As String
    ' Create the MText object in model space
    textString = "Hello, World."
```

```
    insertionPoint(0) = 5
    insertionPoint(1) = 5
    insertionPoint(2) = 0
    width = 2
    Set mtextObj = ThisDrawing.ModelSpace. _
               AddMText(insertionPoint, width, textString)
    ' Data for Leader
    points(0) = 0: points(1) = 0: points(2) = 0
    points(3) = 4: points(4) = 4: points(5) = 0
    points(6) = 4: points(7) = 5: points(8) = 0
    leaderType = acLineWithArrow
    ' Create the Leader object in model space and associate
    ' the MText object with the leader
    Set annotationObject = mtextObj
    Set leaderObj = ThisDrawing.ModelSpace. _
            AddLeader(points, annotationObject, leaderType)
    ZoomAll
End Sub
```

¿Comentarios?

Modificación de la asociatividad de las directrices

Salvo en lo relativo a la relación de asociatividad, la directriz y su anotación son objetos totalmente independientes del dibujo. Igual que la modificación de una directriz no afecta a la anotación, la edición de ésta tampoco afecta a la primera.

Aunque la anotación de texto se crea utilizando las variables de sistema DIMCLRT, DIMTXT y DIMTXSTY para definir su color, altura y estilo, estas variables no pueden emplearse para editar la anotación, ya que no se trata de un verdadero objeto de cota. La anotación de texto debe editarse como cualquier otro objeto Mtext.

Utilice el método Evaluate para evaluar la relación de la directriz con la anotación que tiene asociada. Si es preciso, este método actualiza la geometría directriz.

Edición de directrices

Cualquier modificación realizada en una anotación de directriz que cambie su posición afecta a la posición del extremo de la directriz asociada. De igual forma, al rotar la anotación también gira la línea directriz (si existe).

Si desea ajustar el tamaño de una directriz, puede atribuirle un factor de escala. Cuando se atribuye una escala sólo se actualiza el objeto designado. Por ejemplo, si se ajusta la escala de la directriz, la anotación permanece en la misma posición respecto del punto final de la directriz, pero no cambia de escala.

Las directrices, además de ampliarse o reducirse, también se pueden desplazar, reflejar y rotar. Utilice los métodos ScaleEntity, Move, Mirror y Rotate para modificar directrices. El estilo de texto que tiene asociado una anotación también se puede cambiar mediante la propiedad StyleName.

Uso de tolerancias geométricas

Las tolerancias geométricas indican las desviaciones de forma, perfil, orientación, ubicación y oscilación de una característica. Las tolerancias geométricas se añaden a los rectángulos de tolerancia. Estos rectángulos contienen toda la información sobre la tolerancia de una sola cota.

Para obtener más información acerca del uso de rectángulos de tolerancia y el trabajo con tolerancias geométricas, véase la sección “Adición de tolerancias geométricas” en el *Manual del usuario*.

- **Creación de tolerancias geométricas**
- **Modificación de tolerancias**

Creación de tolerancias geométricas

Para crear una tolerancia geométrica, utilice el método AddTolerance. Este método requiere la introducción de tres valores: la cadena de texto que contiene el símbolo de tolerancia, la ubicación del dibujo donde se situará la tolerancia y un vector de dirección para determinar la dirección de la tolerancia. Las tolerancias también se pueden copiar, desplazar, borrar, ampliar o reducir y rotar.

Creación de una tolerancia geométrica

Este ejemplo crea una tolerancia geométrica sencilla en espacio modelo.

```
Sub Ch5_CreateTolerance()
    Dim toleranceObj As AcadTolerance
    Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim direction(0 To 2) As Double
    ' Define the tolerance object
    textString = "Here is the Feature Control Frame"
    insertionPoint(0) = 5
    insertionPoint(1) = 5
    insertionPoint(2) = 0
    direction(0) = 1
    direction(1) = 1
    direction(2) = 0
    ' Create the tolerance object in model space
    Set toleranceObj = ThisDrawing.ModelSpace. _
        AddTolerance(textString, insertionPoint, direction)
    ZoomAll
End Sub
```

Modificación de tolerancias

Las tolerancias están influidas por varias variables de sistema: DIMCLRD controla el color del rectángulo de tolerancia, DIMCLRT controla el color del texto de la tolerancia, DIMGAP controla el espacio entre el rectángulo de tolerancia y el texto, DIMTXT controla el tamaño del texto de la tolerancia y DIMTXTSTY controla el estilo de texto de la tolerancia. Utilice el método SetVariable para ajustar los valores de las variables del sistema.

<$nopage>MenuGroups (colección):<$nopage>menú (objetos): <$startrange>InsertInMenuBar (método):código de ejemplo, separadores:añadir a menús, PopupMenuItem (objeto):AddSeparator (método), AddSeparator (método), utilizar la propiedad Type, Label (propiedad):tecla rápida, PopupMenuItem (objeto): teclas rápidas, asignar, submenús:añadir, submenús:colocar, PopupMenu (objeto):crear submenús, menús:crear submenús, <$nopage>cascada (menús). <$endrange>InsertInMenuBar (método):código de ejemplo, menús:suprimir, PopupMenu (objeto):suprimir elementos de menú, PopupMenuItem (objeto):suprimir elementos de menú, Delete (método):código de ejemplo, PopupMenuItem (objeto):Tag (propiedad), PopupMenuItem (objeto):Label (propiedad), PopupMenuItem (objeto):Caption (propiedad), PopupMenuItem (objeto):Macro (propiedad), PopupMenuItem (objeto):HelpString (propiedad), PopupMenuItem (objeto):Enable (propiedad), PopupMenuItem (objeto):Check (propiedad), menús:activar, menús:desactivar, menús:verificar, menús:colocar, PopupMenuItem (objeto):Index (propiedad), PopupMenuItem (objeto):Type (propiedad), PopupMenuItem (objeto):Submenu (propiedad), PopupMenuItem (objeto):Parent (propiedad), PopupMenuItem (objeto):Parent (propiedad), menús:tipo de elemento de menú, menús:devolver submenús, menús:asignar elementos de menú, menús:macros de menús, caracteres especiales, InsertInMenuBar (método):código de ejemplo, Toolbars (colección):Add (método), Toolbars (colección):Name (propiedad), Name (propiedad):barras de herramientas, Toolbar (objeto): asignar nombre, Add (método):barras de herramientas, código de ejemplo, Toolbar (objeto):código de ejemplo, AddToolbarButton (método), Toolbar (objeto):AddToolbarButton (método), ToolbarItem (objeto), ToolbarItem (objeto):colocar botones de barras de herramientas, ToolbarItem (objeto):Name (propiedad), Name (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):HelpString (propiedad), HelpString (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):Macro (propiedad), Macro (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):crear botón desplegable, botón desplegable, Toolbar (objeto):código de ejemplo, AddToolbarButton (método):código de ejemplo, ToolbarItem (objeto):código de ejemplo, Toolbar (objeto):AddSeparator (método), Toolbar (objeto):usar la propiedad Type, separadores:añadir a barras de herramientas, SetBitmaps (método), ToolbarItem (objeto):SetBitmaps (método), GetBitmaps (método), ToolbarItem (objeto):GetBitmaps, SetBitmaps (método):SmallIconName (parámetro), SetBitmaps (método):LargeIconName (parámetro), Type (propiedad):código de ejemplo, GetBitmaps (método):código de ejemplo, ToolbarItem (objeto):código de ejemplo, iconos desplegables, barras de herramientas:AddToolbarButton (método), AddToolbarButton (método):crear

barras de herramientas desplegables, Toolbar (objeto):barras de herramientas desplegables (crear), AddToolbarButton (método):código de ejemplo, Toolbar (objeto):código de ejemplo, ToolbarItem (objeto):código de ejemplo, iconos desplegables, barras de herramientas:código de ejemplo, AttachToolbarToFlyout (método), código de ejemplo, Visible (propiedad):código de ejemplo, Toolbar (objeto):Float (método), Float (método):barras de herramientas flotantes, Toolbar (objeto):Dock (método), Dock (método):anclar barras de herramientas, Toolbar (objeto):Docked (propiedad), Dock (método):código de ejemplo, Toolbar (objeto):código de ejemplo, ToolbarItem (objeto):suprimir, ToolbarItem (objeto):Tag (propiedad), Tag (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):Name (propiedad), Name (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):Macro (propiedad), Macro (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):HelpString (propiedad), HelpString (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):Index (propiedad), Index (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):Type (propiedad), Type (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):Flyout (propiedad), Flyout (propiedad):ToolbarItem (objeto), ToolbarItem (objeto):Parent (propiedad), Parent (propiedad):ToolbarItem (objeto), PopupMenuItem (objeto):escribir macros, ToolbarItem (objeto):escribir macros, macros:en menús, macros:en barras de herramientas, macros:directrices de escritura, PICKAUTO (variable de sistema), PICKADD (variable de sistema), macros:caracteres especiales (tabla), macros de menú:caracteres especiales (tabla), macros de barras de herramientas:caracteres especiales (tabla), macros:terminar, terminar macros:ejemplos de código, macros:datos del usuario, entrada de usuario, interrumpir macros, CAPA (comando):macros, macros:CAPA (comando), DESIGNA (comando):macros, macros:DESIGNA (comando), contrabarra (\\):macros, macros:contrabarra, carácter, macros:retardos (lista), interrumpir macros:retardos (lista), macros:cancelar comandos, macros:manipulación de comandos, macros:uso de la repetición, macros:designación de objetos únicos, designación de objetos únicos en macros, BORRA (comando):macros, macros:BORRA (comando), ToolbarItem (objeto):mensaje de ayuda en la línea de estado, PopupMenuItem (objeto):mensaje de ayuda en la línea de estado, ayuda en la línea de estado, para menús y barras de herramientas, Ayuda, línea de estado:para menús y barras de herramientas, InsertInMenuBar (método):código de ejemplo, cursor (menús):ShortcutMenu (propiedad), cursor (menús):añadir elementos nuevos, <$nopage>menú contextual.

Personalización de barras de herramientas y menús

ActiveX Automation de AutoCAD permite un alto grado de control sobre la personalización de menús y barras de herramientas en la sesión actual de AutoCAD.

Con AutoCAD ActiveX/VBA, puede modificar o incrementar la estructura de menús existente, o bien reemplazarla por completo. También puede realizar cambios en las barras de herramientas y en los menús contextuales.

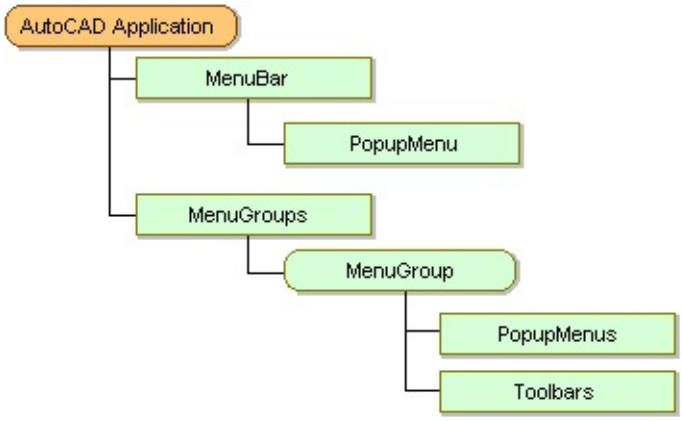
La personalización de menús puede aumentar el rendimiento, ya que permite exponer tareas específicas de una aplicación o concentrar tareas de muchos pasos en una sencilla opción de menú.

Para obtener información adicional sobre personalización de menús y barras de herramientas, consulte el *Manual de personalización*.

- **Conceptos básicos sobre las colecciones MenuBar y MenuGroups**
- **Carga de grupos de menús**
- **Cambio de la barra de menús**
- **Creación y modificación de menús desplegables y contextuales**
- **Creación y modificación de barras de herramientas**
- **Creación de macros**
- **Creación de mensajes de ayuda en la línea de estado para elementos de menús y de barras de herramientas**
- **Adición de entradas al menú contextual**

Conceptos básicos sobre las colecciones MenuBar y MenuGroups

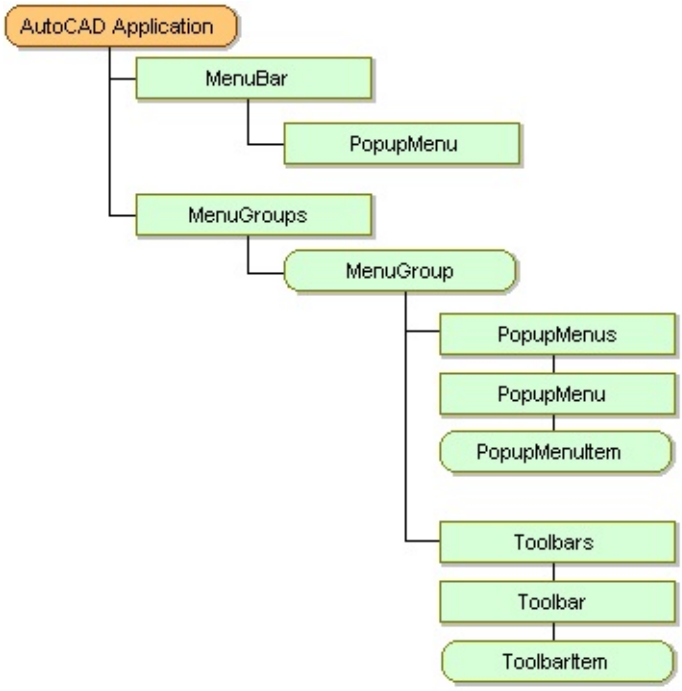
AutoCAD® ActiveX® ofrece varios objetos relacionados con los menús. Los dos más importantes son la colección MenuBar y la colección MenuGroups. La colección MenuBar contiene todos los menús que se muestran en la barra de menús de AutoCAD.



La colección MenuGroups incluye los grupos de menús que se cargan en la sesión activa de AutoCAD. Estos grupos contienen todos los menús que están disponibles en la sesión de AutoCAD en la barra de menús se pueden mostrar todos los menús o parte de ellos. Además de los menús, los grupos de menús incluyen también todas las barras de herramientas que están disponibles en la sesión activa de AutoCAD. Los grupos de menús también representan menús en mosaico, menús de pantalla o menús de tablero.

Todos los grupos de menús contienen una colección PopupMenus y otra Toolbars. La colección PopupMenus incluye todos los menús del grupo de menús. De igual forma, la colección Toolbars contiene todas las barras de herramientas del grupo de menús.

Cada PopupMenu es en realidad una colección que contiene un objeto individual por cada elemento de menú que aparece en el mismo. De igual forma, cada Toolbar es una colección que consta de un objeto individual por cada elemento de la barra de herramientas.



¿Comentarios?

<$nopage>MenuGroups (colección):

Carga de grupos de menús

Los grupos de menús se cargan en AutoCAD mediante el método Load. Por ejemplo, el siguiente código carga el archivo de personalización *acad.cui:*

```
ThisDrawing.Application.MenuGroups.Load "acad.cui"
```

Cuando utilice dicho método, asigne al parámetro `BaseMenu` el valor `TRUE` para cargar un grupo de menús nuevo a la barra de menús. El grupo de menús se cargará como menú base de la misma manera que el comando MENU en AutoCAD.

Para cargar como menú parcial un grupo de menús nuevo, omita el parámetro `BaseMenu`. El grupo de menús se cargará de la misma manera que el comando MENULOAD en AutoCAD. Una vez cargados en la colección MenuGroups, los menús parciales pueden añadirse a la barra de menús mediante los métodos InsertMenuInMenuBar o InsertInMenuBar.

Después de cargar un grupo de menús, todos los menús y barras de herramientas definidos por él estarán disponibles. Es posible:

- Añadir nuevos menús a la barra de menús
- Eliminar menús de la barra de menús
- Reorganizar los menús en la barra de menús
- Añadir nuevos elementos a un menú o una barra de herramientas existente
- Eliminar elementos de un menú o una barra de herramientas existente
- Crear nuevos menús y barras de herramientas

- Dejar las barras de herramientas flotantes o fijas
- Activar o desactivar elementos de menú o barra de herramientas
- Seleccionar o quitar la selección de un elemento de menú
- Cambiar el identificador, la etiqueta o la cadena de búsqueda de un elemento de menú o barra de herramientas
- Volver a asignar las macros asociadas a elementos de menús o barras de herramientas

Nota Sólo se pueden modificar menús desplegables y barras de herramientas mediante ActiveX Automation. No obstante, se puede utilizar ActiveX Automation para cagar y descargar otros tipos de menús, como elementos de menús de símbolos, menús de pantalla o de tablero.

Cambio de la barra de menús

Como ha podido comprobar, la barra de menús puede reemplazarse por un grupo de menús nuevo si éste se encuentra cargado como menú base. También se pueden añadir, eliminar y redistribuir menús individuales en la barra de menús.

- **Inserción de menús en la barra de menús**
- **Eliminación de menús de la barra de menús**
- **Reorganización de opciones de menú en la barra de menús**

Inserción de menús en la barra de menús

Para incluir un menú existente en la barra de menús, utilice los métodos InsertMenuInMenuBar o InsertInMenuBar. Ambos métodos cumplen el mismo objetivo: insertar un menú existente en la barra de menús.

La diferencia entre estos dos métodos radica en el objeto desde el que se llaman. El método InsertMenuInMenuBar se llama desde la colección PopupMenus. Con este método se pueden insertar los menús de una colección en una ubicación específica de la barra de menús. Requiere como entrada el nombre del menú que se desea insertar y la posición de la barra de menús donde se desea colocar.

El método InsertInMenuBar se llama directamente desde el objeto PopupMenu que se desea insertar. La única entrada que requiere este método es la posición en la barra de menús. No es necesario introducir el nombre del menú ya que se está llamando al método directamente desde el objeto.

Utilice el método que sea más idóneo para la aplicación que esté utilizando.

Inserción de menús en la barra de menús

Este ejemplo crea un menú llamado TestMenu e inserta en él un elemento de menú. Se asigna al elemento el comando ABRIR. Después se muestra el menú en la barra de menús.

```
Sub Ch6_InsertMenu()
    ' Define a variable for the current menu group
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application. _
                                    MenuGroups.Item(0)
    ' Create a new menu
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")
    ' Declare the variables for the menu item
```

```
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String
    ' Assign the macro string the VB equivalent of
    ' "ESC ESC _open " and create the menu item
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newMenuItem = newMenu.AddMenuItem(newMenu.Count + 1, _
        "Open", openMacro)
    ' Display the menu on the menu bar
    currMenuGroup.Menus.InsertMenuInMenuBar "TestMenu", ""
End Sub
```

¿Comentarios?

Eliminación de menús de la barra de menús

Para eliminar un menú de la barra de menús, utilice los métodos RemoveMenuFromMenuBar o RemoveFromMenuBar. Ambos métodos cumplen el mismo objetivo: eliminar un menú de la barra de menús.

La diferencia entre estos dos métodos radica en el objeto desde el que se llaman. El método RemoveMenuFromMenuBar se llama desde la colección PopupMenus. Requiere como entrada el nombre del menú que se desea eliminar o la posición donde se encuentra dentro de la barra de menús. Por ejemplo, las siguientes instrucciones eliminan el menú que se añadió en el apartado Inserción de menús en la barra de menús:

```
currMenuGroup.Menus.RemoveMenuFromMenuBar ("TestMenu")
```

El método RemoveFromMenuBar se llama directamente desde el objeto PopupMenu que se desea eliminar. Este método no requiere ninguna entrada. No es necesario introducir el nombre del menú ya que se está llamando al método directamente desde el mismo objeto que se desea eliminar.

Utilice el método que sea más idóneo para la aplicación que esté utilizando.

Nota Los menús que se eliminan de la barra de menús permanecen disponibles en su grupo de menús, aunque ocultos a la vista del usuario.

Reorganización de opciones de menú en la barra de menús

Para organizar los menús de la barra de menús, inserte y elimine los que necesite hasta obtener la configuración deseada.

Desplazamiento del primer menú al final de la barra de menús

Este ejemplo elimina el primer menú de la barra de menús y lo inserta como último menú de la barra.

```
Sub Ch6_MoveMenu()
    ' Define a variable to hold the menu to be moved
    Dim moveMenu As AcadPopupMenu
    Dim MyMenuBar As AcadMenuBar
    Set MyMenuBar = ThisDrawing.Application.menuBar
    ' Set moveMenu equal to the first menu displayed
    ' on the menu bar
    Set moveMenu = MyMenuBar.Item(0)
    ' Remove the first menu from the menu bar
    MyMenuBar.Item(0).RemoveFromMenuBar
    ' Add the menu back into the menu bar
    ' in the last position on the bar
    moveMenu.InsertInMenuBar (MyMenuBar.count)
End Sub
```

<$nopage>menú (objetos):<$startrange>InsertInMenuBar (método):código de ejemplo, separadores:añadir a menús, PopupMenuItem (objeto):AddSeparator (método), AddSeparator (método), utilizar la propiedad Type, Label (propiedad):tecla rápida, PopupMenuItem (objeto): teclas rápidas, asignar, submenús:añadir, submenús:colocar, PopupMenu (objeto):crear submenús, menús:crear submenús, <$nopage>cascada (menús). <$endrange>InsertInMenuBar (método):código de ejemplo, menús:suprimir, PopupMenu (objeto):suprimir elementos de menú, PopupMenuItem (objeto):suprimir elementos de menú, Delete (método):código de ejemplo, PopupMenuItem (objeto):Tag (propiedad), PopupMenuItem (objeto):Label (propiedad), PopupMenuItem (objeto):Caption (propiedad), PopupMenuItem (objeto):Macro (propiedad), PopupMenuItem (objeto):HelpString (propiedad), PopupMenuItem (objeto):Enable (propiedad), PopupMenuItem (objeto):Check (propiedad), menús:activar, menús:desactivar, menús:verificar, menús:colocar, PopupMenuItem (objeto):Index (propiedad), PopupMenuItem (objeto):Type (propiedad), PopupMenuItem (objeto):Submenu (propiedad), PopupMenuItem (objeto):Parent (propiedad), PopupMenuItem (objeto):Parent (propiedad), menús:tipo de elemento de menú, menús:devolver submenús, menús:asignar elementos de menú, menús:macros de menús, caracteres especiales, InsertInMenuBar (método):código de ejemplo,">

Creación y modificación de menús desplegables y contextuales

AutoCAD ActiveX/VBA tiene la capacidad de personalizar dos tipos de menús de AutoCAD: menús desplegables y contextuales (a veces denominados menús de cursor). Ambos se muestran como menús en cascada. El menú contextual permite acceder con rapidez a elementos de menú de uso frecuente, como los modos de referencia a objetos.

Un menú desplegable puede contener hasta 999 elementos de menú. Un menú contextual puede contener hasta 499 elementos de menú. Ambos límites

incluyen a todos los menús de la jerarquía. Si el número de elementos de un menú excede dichos límites, AutoCAD ignora los elementos sobrantes. Si un menú desplegable o contextual sobrepasa el espacio disponible en la pantalla gráfica, se trunca para ajustarlo a la pantalla.

Los menús desplegables siempre se despliegan desde la barra de menús, mientras que los contextuales siempre se muestran en la posición de los punteros en cruz de la pantalla gráfica, o cerca de ellos. El manejo de ambos tipos de menús es el mismo, excepto que el título del menú contextual no está incluido en la barra de menús. Simplemente no se muestra. El acceso al menú contextual se realiza a través de un menú único situado en el grupo de menús base. El menú contextual se puede identificar con la propiedad ShortcutMenu. Si esta propiedad devuelve TRUE, el menú consultado es el menú contextual del grupo.

- **Creación de menús nuevos**
- **Adición de nuevos elementos de menú a un menú**
- **Adición de separadores a un menú**
- **Asignación de tecla rápida a una opción de menú**
- **Creación de submenús en cascada**
- **Eliminación de opciones de menús**
- **Examen de propiedades de elementos de menú**

Creación de menús nuevos

Para crear un menú nuevo, agregue un objeto PopupMenu a la colección PopupMenus con el método Add.

Para crear un menú contextual, primero debe eliminar uno ya existente. Sólo puede haber un menú contextual en cada grupo de menús. Si no hay otro menú contextual, puede añadir uno con la etiqueta “POP0”. Así indicará a AutoCAD que desea crear un menú de este tipo.

El método Add requiere como entrada el nombre (etiqueta) del menú que se desea añadir. Este nombre será el título del menú cuando se cargue en la barra de menús. Además, el nombre es la manera más fácil de identificar el menú en la colección.

El nombre de un menú puede ser una simple cadena de texto o contener códigos especiales. Los nombres de los menús se pueden cambiar después de su creación. Para cambiar el nombre de un menú existente, utilice la propiedad Name del menú.

Creación de un menú emergente

En este ejemplo se crea un nuevo menú emergente llamado “TestMenu” en el primer grupo de menús de la colección MenuGroups.

```
Sub Ch6_CreateMenu()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Create the new menu
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")
End Sub
```

¿Comentarios?

<$nopage>menú (objetos):<$startrange>InsertInMenuBar (método):código de ejemplo,">

Adición de nuevos elementos de menú a un menú

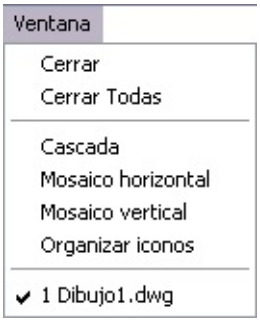
Para agregar nuevos elementos a un menú, utilice el método AddMenuItem. Este método crea un nuevo objeto PopupMenuItem y lo añade al menú indicado.

El método AddMenuItem requiere cuatro parámetros de entrada:`Index`, `Label`, `Tag` y `Macro`.

- **Especificar el parámetro Index**
- **Especificar el parámetro Label**
- **Especificar el parámetro Tag**
- **Especificar el parámetro Macro**

Especificar el parámetro Index

El parámetro `Index`, o índice, es un número entero que determina la posición del nuevo elemento en el menú. El índice comienza por la posición cero (0), que corresponde al primer puesto en el menú después del título. Para añadir el nuevo elemento al final de un menú, asigne al parámetro `Index` el valor de la propiedad Count del menú (la propiedad Count del menú indica el número total de elementos que contiene el menú). (La propiedad Count del menú representa el número total de elementos de menú en ese menú.)



La primera posición del índice es cero (0) y que los separadores aparecen como elementos de menú individuales con su propio número de índice. La propiedad Count del menú de la ilustración sería seis (6). Para añadir un elemento de menú entre Tile Horizontally y Tile Vertically, asigne al parámetro `Index` el valor dos (2), que es el número de índice del elemento de menú Tile Vertically. El nuevo elemento de menú se inserta en el número de índice dos (2) y empuja a todos los demás elementos una posición hacia abajo.

Una vez creada una opción de menú, su índice no se puede cambiar con la propiedad Index. Para cambiar el número de índice de una opción de menú existente, debe borrar la opción y volver a añadirla en una posición diferente, o añadir o suprimir las opciones circundantes hasta alcanzar la posición adecuada.

¿Comentarios?

Especificar el parámetro Label

Una etiqueta (Label) es una cadena que define el contenido y el formato de las opciones de menú. Las etiquetas de elementos de menú pueden contener expresiones de cadena DIESEL condicionales que las alteren cada vez que se visualicen.

Además de las expresiones de cadena DIESEL, la etiqueta puede contener códigos especiales. Por ejemplo, un signo & colocado justo antes de un carácter indica que dicho carácter es la tecla rápida del elemento.

El texto que el usuario ve en pantalla como opción de menú se denomina 'leyenda' y procede de la interpretación de todas las expresiones de cadena DIESEL y códigos especiales incluidos en la etiqueta. Por ejemplo, la etiqueta “&Edit” produce la leyenda “Editar.”.

Si desea cambiar la etiqueta después de crear la opción de menú, utilice la propiedad Label.

Especificar el parámetro Tag

El identificador, o identificador de nombre, es una cadena compuesta por caracteres alfanuméricos y de subrayado (_). Esta cadena identifica de forma única un elemento dentro de un determinado menú.

Si desea cambiar el identificador después de crear la opción de menú, utilice la propiedad TagString.

<$startrange>InsertInMenuBar (método):código de ejemplo,">

Especificar el parámetro Macro

Una macro es una serie de comandos que ejecutan acciones específicas cuando se selecciona un menú. Las macros de menús pueden ser simples registros de pulsaciones de teclas que efectúen una tarea o una combinación compleja de comandos, AutoLISP, DIESEL, o código de programación de ActiveX.

Si desea cambiar la macro después de crear la opción de menú, utilice la propiedad Macro.

Adición de opciones a un menú emergente

En este ejemplo se crea un nuevo menú llamado “TestMenu” y se inserta en el mismo una opción de menú. La opción de menú recibe el nombre “Open” y la macro que se le asigna es el comando OPEN.

```
Sub Ch6_AddAMenuItem()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Create the new menu
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")
    ' Add a menu item to the new menu
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String
    ' Assign the macro the VBA equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newMenuItem = newMenu.AddMenuItem _
                      (newMenu.count + 1, "Open", openMacro)
    ' Display the menu on the menu bar
    newMenu.InsertInMenuBar _
                  (ThisDrawing.Application.menuBar.count + 1)
End Sub
```

¿Comentarios?

Adición de separadores a un menú

Para añadir un separador a un menú, utilice el método AddSeparator. Este método crea un nuevo objeto PopupMenuItem y lo añade al menú indicado. A esta clase de objeto PopupMenuItem se le asigna el tipo `acSeparator`. El tipo de elemento de menú se puede conocer con la propiedad Type.

El método AddSeparator utiliza como única entrada el parámetro `Index`. El parámetro `Index`, o índice, es un número entero que especifica la posición del separador dentro del menú. El índice comienza por la posición cero (0), que corresponde al primer puesto en el menú después del título.

Para obtener un ejemplo de cómo añadir separadores a un menú, véase Activación y desactivación de opciones de menú.

Asignación de tecla rápida a una opción de menú

Para asignar la tecla rápida a un elemento de menú mediante AutoCAD ActiveX/VBA, utilice la propiedad Label de ese elemento. Para especificar una tecla rápida, inserte el equivalente ASCII de un signo & en la etiqueta justo antes del carácter que se utilizará como acelerador. Por ejemplo, la etiqueta `Chr(Asc("&")) + "Edit"` se mostrará como “Editar”, con el carácter “E” utilizado como tecla rápida.

Adición de teclas rápidas a menús

En este ejemplo se repite el ejemplo de Adición de opciones a un menú emergente, con la adición de las teclas rápidas a los menús “TestMenu” y “Open”. La “s” y la “O” se utilizan respectivamente como teclas rápidas de los menús “TestMenu” y “Open”.

```
Sub Ch6_AddAMenuItem()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Create the new menu
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add _
                              ("Te" + Chr(Asc("&")) + "stMenu")
    ' Add a menu item to the new menu
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String
    ' Assign the macro the VBA equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newMenuItem = newMenu.AddMenuItem _
                      (newMenu.count + 1, Chr(Asc("&")) _
                       + "Open", openMacro)
    ' Display the menu on the menu bar
    newMenu.InsertInMenuBar _
                (ThisDrawing.Application.menuBar.count + 1)
```

```
End Sub
```

¿Comentarios?

<$nopage>cascada (menús). <$endrange>InsertInMenuBar (método):código de ejemplo,">

Creación de submenús en cascada

Para agregar un submenú en cascada, cree el submenú con el método AddSubmenu. Este método crea un nuevo objeto PopupMenuItem y lo añade al menú indicado. A esta clase de objeto PopupMenuItem se le asigna el tipo `acSubmenu`.

El método AddSubmenu requiere tres parámetros de entrada: `Index`, `Label` y `Tag`.

El parámetro `Index`, o índice, es un número entero que determina la posición del nuevo elemento en el menú. El índice comienza por la posición cero (0), que corresponde al primer puesto en el menú después del título. Para añadir el nuevo elemento al final de un menú, asigne al parámetro `Index` el valor de la propiedad Count del menú (la propiedad Count del menú indica el número total de elementos que contiene el menú). (La propiedad Count del menú representa el número total de elementos de menú en ese menú.)

El parámetro `Label`, o etiqueta, es una cadena que define el contenido y el formato de los elementos de menú. El texto que el usuario ve en pantalla como opción de menú se denomina 'leyenda' y procede de la interpretación de todas las expresiones de cadena DIESEL y códigos especiales incluidos en la etiqueta. Por ejemplo, la etiqueta “&Edit” produce la leyenda “Editar.”.

El parámetro `Tag`, o identificador de nombre, es una cadena compuesta por caracteres alfanuméricos y de subrayado (_). Esta cadena identifica de forma única un elemento dentro de un determinado menú.

El método AddSubmenu no da como resultado el objeto PopupMenuItem que crea, sino que devuelve el nuevo menú al que apunta el submenú. En su lugar,

devuelve el nuevo menú al que apunta el submenú. El nuevo menú, devuelto como objeto PopupMenu, podrá rellenarse como cualquier otro menú normal. Para obtener información sobre cómo rellenar un menú, véase Adición de nuevos elementos de menú a un menú.

Creación y relleno de un submenú

En este ejemplo, se crea un nuevo menú denominado “TestMenu” y se agrega a un submenú denominado “OpenFile.” A continuación el submenú se rellena con una opción de menú denominada “Open,”, que, al ejecutarla, abre un dibujo. Por último, se muestra el menú en la barra de menús.

```
Sub Ch6_AddASubMenu()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Create the new menu
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")
    ' Add the submenu
    Dim FileSubMenu As AcadPopupMenu
    Set FileSubMenu = newMenu.AddSubMenu("", "OpenFile")
    ' Add a menu item to the sub menu
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String
    ' Assign the macro the VB equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newMenuItem = FileSubMenu.AddMenuItem _
                          (newMenu.count + 1, "Open", openMacro)
    ' Display the menu on the menu bar
    newMenu.InsertInMenuBar _
                    (ThisDrawing.Application.menuBar.count + 1)
End Sub
```

Eliminación de opciones de menús

Para eliminar elementos de un menú, utilice el método Delete del elemento de menú.

Advertencia Si suprime una opción de menú, no invoque otros métodos o propiedades que directa o indirectamente vuelvan a cargar el mismo archivo IUP dentro de la misma macro. Por ejemplo, después de suprimir una opción de menú, no utilice el método MenuGroup Load ni la propiedad Preferences.Profiles.ActiveProfile, ni utilice el comando “Menuload” usando el método Document.SendCommand. Estas opciones causan directa o indirectamente la carga de archivos IUP. Solo debe usar estos métodos o propiedades en una macro separada.

Eliminación de una opción de un menú

Este ejemplo añade un elemento de menú al final del último menú mostrado en la barra de menús. Después elimina el elemento.

```
Sub Ch6_DeleteMenuItem()
    Dim LastMenu As AcadPopupMenu
    Set LastMenu = ThisDrawing.Application.menuBar. _
                Item(ThisDrawing.Application.menuBar.count - 1)
    ' Add a menu item
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String
    ' Assign the macro the VB equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newMenuItem = LastMenu.AddMenuItem _
                          (LastMenu.count + 1, "Open", openMacro)
    ' Remove the menu item from the menu
    newMenuItem.Delete
End Sub
```

¿Comentarios?

Examen de propiedades de elementos de menú

Las siguientes propiedades están presentes en todas las opciones de menús:

TagString

El identificador, o identificador de nombre, es una cadena compuesta por caracteres alfanuméricos y de subrayado (_). Esta cadena identifica de forma única un elemento dentro de un determinado menú. Los identificadores denotan las teclas rápidas (secuencias de pulsación de teclas) correspondientes a cada opción de menú.

Utilice la propiedad TagString para leer o escribir el valor de los identificadores.

Label

Una etiqueta (Label) es una cadena que define el contenido y el formato de las opciones de menú.

Las etiquetas de elementos de menú pueden contener expresiones de cadena DIESEL condicionales que las alteren cada vez que se visualicen.

Utilice la propiedad Label para leer o escribir el valor de las etiquetas.

Caption

Una leyenda es el texto que el usuario ve como nombre del menú. Es una propiedad de sólo lectura que procede de la propiedad Label una vez que se eliminan las expresiones de cadena DIESEL.

Utilice la propiedad Caption para leer o escribir el valor de las leyendas.

Macro

Una macro es una serie de comandos que ejecutan acciones específicas cuando se selecciona un menú. Las macros de menús pueden ser simples

registros de pulsaciones de teclas que efectúen una tarea o una combinación compleja de comandos, AutoLISP, DIESEL, o código de programación de ActiveX.

Utilice la propiedad Macro para leer o escribir el valor de las macros de menú.

HelpString

La cadena de ayuda (HelpString) es la secuencia de texto que aparece en la línea de estado de AutoCAD cuando un usuario resalta una opción para seleccionarla.

Utilice la propiedad HelpString para escribir o leer el valor de una cadena de ayuda.

Enable

Utilice la propiedad Enable para activar o desactivar una opción de menú. También puede leer la propiedad Enable para determinar si un elemento de menú está seleccionado o no. Cuando se utiliza esta propiedad para activar o desactivar una opción de menú, quedan sustituidos los parámetros de activación que tenga la expresión DIESEL de la opción.

Para obtener un ejemplo de desactivación de elementos de menú, véase Examen de propiedades de elementos de menú.

Check

Utilice la propiedad Check para seleccionar o retirar la selección de una opción de menú. También puede leer la propiedad Check para determinar si un elemento de menú está seleccionado o no. Cuando se utiliza esta propiedad para seleccionar o retirar la selección de una opción de menú, quedan sustituidos los parámetros de selección que tenga la expresión DIESEL de la opción.

Index

El índice de una opción de menú determina su posición dentro del menú al que pertenece. La posición del índice de un menú siempre es la 0. Por ejemplo, si un elemento es el primero de un menú, devuelve una posición de índice 0. Si es el segundo elemento, devuelve la posición de índice 1 y así sucesivamente.

Type

Utilice la propiedad Type para determinar el tipo de opción de menú. Una opción de menú puede pertenecer a las siguientes tipologías: menú normal, separador o encabezamiento de un submenú. Si es un elemento de menú normal, esta propiedad devuelve `acMenuItem`. Si el elemento es un separador, esta propiedad devuelve `acMenuSeparator`. Si el elemento es el encabezamiento de un submenú, esta propiedad devuelve `acSubMenu`.

SubMenu

Utilice la propiedad SubMenu para localizar el submenú. Si el elemento de menú es del tipo `acSubMenu`, esta propiedad devuelve el menú que está asociado como submenú, o menú incrustado. El menú incrustado se devuelve como un objeto PopupMenu.

Si el elemento de menú no es del tipo `acSubMenu`, esta propiedad devuelve un error.

Parent

Utilice la propiedad Parent para localizar el menú al que pertenece una opción de menú. Esta propiedad devuelve el menú donde reside la opción. El menú superior se devuelve como objeto PopupMenu.

Activación y desactivación de opciones de menú

En este ejemplo se crea un menú nuevo denominado “TestMenu” y se insertan dos opciones de menú. Se desactiva el segundo elemento mediante la propiedad Enable y se muestra el menú en la barra de menús.

```
Sub Ch6_DisableMenuItem()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Create the new menu
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add("TestMenu")
    ' Add two menu items and a menu separator to the new menu
    Dim MenuEnable As AcadPopupMenuItem
    Dim MenuDisable As AcadPopupMenuItem
    Dim MenuSeparator As AcadPopupMenuItem
    Dim openMacro As String
    ' Assign the macro the VB equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set MenuEnable = newMenu.AddMenuItem _
                (newMenu.count + 1, "OpenEnabled", openMacro)
```

```
    Set MenuSeparator = newMenu.AddSeparator("")
    Set MenuDisable = newMenu.AddMenuItem _
                (newMenu.count + 1, "OpenDisabled", openMacro)
    ' Disable the second menu item
    MenuDisable.Enable = False
    ' Display the menu on the menu bar
    newMenu.InsertInMenuBar _
                (ThisDrawing.Application.menuBar.count + 1)
End Sub
```

¿Comentarios?

Creación y modificación de barras de herramientas

Se puede utilizar AutoCAD ActiveX/VBA para crear y modificar barras de herramientas.

- **Creación de barras de herramientas**
- **Adición de botones nuevos a una barra de herramientas**
- **Adición de separadores a una barra de herramientas**
- **Definición de los símbolos de botones de barras de herramientas**
- **Creación de barras de herramientas desplegables**
- **Barras de herramientas flotantes y fijas**
- **Eliminación de botones de una barra de herramientas**
- **Examen de propiedades de elementos de barra de herramientas**

Creación de barras de herramientas

Para crear una barra de herramientas nueva, agregue un nuevo objeto Toolbar a la colección Toolbars con el método Add.

El método Add requiere como entrada el nombre de la barra de herramientas que se desea añadir. El nombre es una cadena de caracteres alfanuméricos sin más signos de puntuación que un guión (–) o un subrayado (_). El nombre es la manera más fácil de identificar la barra de herramientas en la colección.

Los nombres de barras de herramientas se pueden cambiar después de su creación. Para cambiar el nombre de una barra de herramientas existente, utilice la propiedad Name de dicha barra de herramientas.

Creación de una barra de herramientas nueva

En este ejemplo se crea una barra de herramientas nueva denominada “TestToolbar” en el primer grupo de menús de la colección MenuGroups.

```
Sub Ch6_CreateToolbar()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Create the new toolbar
    Dim newToolbar As AcadToolbar
    Set newToolbar = currMenuGroup.Toolbars.Add("TestToolbar")
End Sub
```

Adición de botones nuevos a una barra de herramientas

Para añadir un nuevo botón de herramienta a una barra de herramientas, utilice el método AddToolbarButton. Este método crea un objeto ToolbarItem nuevo y lo añade a la barra de herramientas indicada. Sólo debe añadir botones a una barra de herramientas mientras ésta esté visible.

El método AddToolbarButton requiere cinco parámetros de entrada: `Index`, `Name`, `HelpString`, `Macro` y `FlyoutButton`.

Index

El parámetro `Index` es un número entero que determina la posición del nuevo elemento de Toolbar en la barra de herramientas. El índice comienza por la posición cero (0), que corresponde al primer puesto en la barra de herramientas después del título. Para añadir el nuevo botón al final de una barra de herramientas, asigne al parámetro `Index` el valor de la propiedad Count de la barra de herramientas. (La propiedad Count de la barra de herramientas representa el número total de botones que contiene.)

Una vez creado un botón de barra de herramientas, su índice no se puede cambiar con la propiedad Index. Para cambiar el número de índice de un botón de barra de herramientas existente, debe eliminar y volver a añadir el botón en una posición diferente, o añadir o eliminar los botones que lo rodean hasta obtener la posición correcta.

Name

Un nombre es una cadena identificativa del botón de barra de herramientas. La cadena debe estar formada por caracteres alfanuméricos, sin más signos de puntuación que un guión (–) o un subrayado (_). Esta cadena se muestra como una pista cuando el cursor se sitúa sobre el botón de herramienta.

Una vez creado un botón de barra de herramientas, se puede cambiar su nombre con el parámetro `Name`.

HelpString

La cadena de ayuda (HelpString) es la secuencia de texto que aparece en la línea de estado de AutoCAD cuando un usuario resalta una opción para seleccionarla.

Una vez creado un botón de barra de herramientas, se puede cambiar su cadena de ayuda con el parámetro `HelpString`.

Macro

Una macro es una serie de comandos que ejecutan acciones específicas cuando se selecciona un botón. Las macros de barras de herramientas pueden ser registros simples de pulsaciones de teclas que efectúen una tarea o una combinación compleja de comandos y código de programación de lenguaje DIESEL.

Una vez creado un botón de barra de herramientas, se puede cambiar su macro con el parámetro `Macro`.

FlyoutButton

El parámetro `FlyoutButton` es un identificador opcional que indica si el nuevo botón será desplegable o no. Si el nuevo botón es desplegable, el parámetro debe tomar el valor `TRUE`. En caso negativo, este parámetro debe tomar el valor `FALSE` o se puede pasar por alto.

Adición de botones a una barra de herramientas nueva

Este ejemplo crea una barra de herramientas nueva y le añade un botón. El botón se asigna a una macro que ejecuta el comando OPEN cuando se pulsa el botón.

```
Sub Ch6_AddButton()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Create the new toolbar
    Dim newToolbar As AcadToolbar
    Set newToolbar = currMenuGroup.Toolbars.Add("TestToolbar")
    ' Add a button to the new toolbar
    Dim newButton As AcadToolbarItem
    Dim openMacro As String
    ' Assign the macro the VB equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newButton = newToolbar.AddToolbarButton _
                    ("", "NewButton", "Open a file.", openMacro)
End Sub
```

¿Comentarios?

Adición de separadores a una barra de herramientas

Para añadir un separador a una barra de herramientas, utilice el método AddSeparator. Este método crea un objeto ToolbarItem nuevo y lo añade a la barra de herramientas indicada. A esta clase de objetos ToolbarItem se les asigna el tipo `acSeparator`. El tipo de botón de barra de herramientas se puede conocer con la propiedad Type.

El método AddSeparator requiere la entrada de un parámetro: `Index`. El parámetro `Index` es un número entero que determina la posición del separador dentro de la barra de herramientas. El índice comienza por la posición cero (0), que corresponde al primer puesto en la barra de herramientas después del título.

Definición de los símbolos de botones de barras de herramientas

Para definir las imágenes que se desean utilizar en cada botón de una barra de herramientas, utilice los métodos SetBitmaps y GetBitmaps.

El método SetBitmaps requiere dos parámetros: `SmallIconName` y `LargeIconName`.

SmallIconName

El nombre del icono pequeño define la cadena de identificación del recurso de imagen pequeña (mapa de bits de 16×15). Esta cadena debe estar compuesta por caracteres alfanuméricos, sin más signos de puntuación que un guión (–) o subrayado (_) y debe incluir la extensión *.bmp*. El recurso puede ser un mapa de bits del sistema o un mapa de bits definido por el usuario. Los mapas de bits definidos por el usuario deben tener el tamaño apropiado y deben estar ubicados en la ruta Support.

LargeIconName

El nombre del icono grande define la cadena de identificación del recurso de imagen grande (mapa de bits de 24×22). Esta cadena debe estar compuesta por caracteres alfanuméricos, sin más signos de puntuación que un guión (–) o subrayado (_) y debe incluir la extensión *.bmp*. El recurso puede ser un mapa de bits del sistema o un mapa de bits definido por el usuario. Los mapas de bits definidos por el usuario deben tener el tamaño apropiado y deben estar ubicados en la ruta Support.

Consulta de una barra de herramientas para obtener el nombre de los iconos de los botones

```
Sub Ch6_GetButtonImages()
```

```
    Dim Button As AcadToolbarItem
    Dim Toolbar0 As AcadToolbar
    Dim MenuGroup0 As AcadMenuGroup
    Dim SmallButtonName As String
    Dim LargeButtonName As String
    Dim msg As String
    Dim ButtonType As String
    ' Get the first toolbar in the first menu group
    Set MenuGroup0 = ThisDrawing.Application. _
                                MenuGroups.Item(0)
    Set Toolbar0 = MenuGroup0.Toolbars.Item(0)
    ' Clear the string variables
    SmallButtonName = ""
    LargeButtonName = ""
    ' Create a header for the message box and
    ' display the toolbar to be queried
    msg = "Toolbar: " + Toolbar0.Name + vbCrLf
    Toolbar0.Visible = True
    ' Iterate through the toolbar and collect data
    ' for each button in the toolbar. If the toolbar is
    ' a normal button or a flyout, collect the small
    ' and large button names for the button.
    For Each Button In Toolbar0
        ButtonType = Choose(Button.Type + 1, "Button", _
                        "Separator", "Control", "Flyout")
        msg = msg & ButtonType & ": "
        If Button.Type = acToolbarButton Or _
                       Button.Type = acToolbarFlyout Then
            Button.GetBitmaps SmallButtonName, _
                              LargeButtonName
            msg = msg + SmallButtonName + ", " _
                      + LargeButtonName
        End If
        msg = msg + vbCrLf
    Next Button
    ' Display the results
    MsgBox msg
End Sub
```

¿Comentarios?

Creación de barras de herramientas desplegables

Para añadir un botón desplegable a una barra de herramientas, utilice el método AddToolbarButton. Este método crea un objeto ToolbarItem nuevo y lo añade a la barra de herramientas indicada.

El método AddToolbarButton requiere cinco parámetros de entrada: `Index`, `Name`, `HelpString`, `Macro` y `FlyoutButton`. Al asignar el valor `TRUE` al parámetro `FlyoutButton`, el nuevo botón se crea como un icono desplegable. El valor que devuelva este método será la nueva barra de herramientas desplegable, que podrá rellenarse como cualquier barra de herramientas normal.

Para obtener más información acerca de cómo rellenar una barra de herramientas, véase Adición de botones nuevos a una barra de herramientas.

Creación de un botón desplegable para barras de herramientas

Este ejemplo crea dos barras de herramientas. La primera contiene un botón desplegable. La segunda está enlazada con el botón desplegable de la primera.

```
Sub Ch6_AddFlyoutButton()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application. _
                                MenuGroups.Item(0)
    ' Create the first toolbar
    Dim FirstToolbar As AcadToolbar
    Set FirstToolbar = currMenuGroup.Toolbars. _
                               Add("FirstToolbar")
    ' Add a flyout button to the first menu on the menu bar
    Dim FlyoutButton As AcadToolbarItem
    Set FlyoutButton = FirstToolbar.AddToolbarButton _
            ("", "Flyout", "Demonstrates a flyout button", _
             "OPEN", True)
    ' Create the second toolbar. This will be attached to
    ' the first toolbar via the flyout button.
```

```
    Dim msg As String
    Set SecondToolbar = currMenuGroup.Toolbars. _
            Add("SecondToolbar")
    ' Add a button to the next toolbar
    Dim newButton As AcadToolbarItem
    Dim openMacro As String
    ' Assign the macro the VB equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newButton = SecondToolbar.AddToolbarButton _
            ("", "NewButton", "Open a file.", openMacro)
    ' Attach the second toolbar to the flyout
    ' button on the first toolbar
    FlyoutButton.AttachToolbarToFlyout currMenuGroup.Name, _
            SecondToolbar.Name
    ' Display the first toolbar, hide the second toolbar
    FirstToolbar.Visible = True
    SecondToolbar.Visible = False
End Sub
```

¿Comentarios?

Barras de herramientas flotantes y fijas

Las barras de herramientas pueden establecerse como fijas o flotantes mediante programación.

Para que una barra de herramientas sea flotante, utilice el método Float de la barra. El método Float requiere la entrada de tres parámetros: `Top`, `Left` y `NumberFloatRows`. Los parámetros `Top` y `Left` especifican la ubicación, en píxeles, de los bordes superior e izquierdo de la barra de herramientas. El parámetro `NumberFloatRows` especifica el número de filas con que se crea una barra de herramientas horizontal. Este número debe ser igual o mayor que uno. Los botones de la barra de herramientas se distribuyen de forma equidistante a lo largo del número de filas indicado. En las barras de herramientas alineadas en sentido vertical, este valor especifica el número de columnas.

Para anclar una barra de herramientas, utilice el método Dock de la barra. El método Dock requiere tres parámetros de entrada: `Side`, `Row` y `Columna`. El parámetro `Side` determina el lado de la barra de herramientas que se situará en la maniobra de anclaje. Puede especificar el lado superior, inferior, izquierdo o derecho de la barra de herramientas. Los parámetros `Row` y `Column` determinan el número de fila y columna de las barras de herramientas ancladas donde se anclará la barra de herramientas.

Puede consultar una barra de herramientas para saber si es fija mediante la propiedad DockStatus. La propiedad DockStatus devolverá `TRUE` si la barra de herramientas está anclada y `FALSE` si es flotante.

Anclaje de una barra de herramientas

Este ejemplo crea una barra de herramientas nueva con tres botones. Después se muestra la barra de herramientas y se ancla a la izquierda de la pantalla.

```
Sub Ch6_DockToolbar()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application. _
                                        MenuGroups.Item(0)
    ' Create the new toolbar
    Dim newToolbar As AcadToolbar
    Set newToolbar = currMenuGroup.Toolbars. _
                                       Add("TestToolbar")
    ' Add three buttons to the new toolbar.
    ' All three buttons will have the same macro attached.
    Dim newButton1 As AcadToolbarItem
    Dim newButton2 As AcadToolbarItem
    Dim newButton3 As AcadToolbarItem
    Dim openMacro As String
    ' Assign the macro the VB equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newButton1 = newToolbar.AddToolbarButton _
                  ("", "NewButton1", "Open a file.", openMacro)
    Set newButton2 = newToolbar.AddToolbarButton _
                  ("", "NewButton2", "Open a file.", openMacro)
    Set newButton3 = newToolbar.AddToolbarButton _
                  ("", "NewButton3", "Open a file.", openMacro)
    ' Display the toolbar
    newToolbar.Visible = True
    ' Dock the toolbar to the left of the screen.
    newToolbar.Dock acToolbarDockLeft
End Sub
```

Eliminación de botones de una barra de herramientas

Para eliminar botones de una barra de herramientas, utilice el método Delete del botón de la barra de herramientas. Sólo debe eliminar botones de una barra de herramientas mientras ésta esté visible.

Examen de propiedades de elementos de barra de herramientas

Las siguientes propiedades están presentes en todos los elementos de barras de herramientas:

Tagstring

El identificador, o identificador de nombre, es una cadena compuesta por caracteres alfanuméricos y de subrayado (_). Esta cadena identifica de forma única un elemento de barra de herramientas dentro de una barra de herramientas determinada. Cuando se crea un elemento de barra de herramientas, se le asigna automáticamente un identificador.

Utilice la propiedad Tagstring para leer o escribir el valor de los identificadores.

Name

Un nombre es una cadena identificativa del elemento de barra de herramientas. También se utiliza como texto de la información de herramienta que aparece en AutoCAD cuando el usuario mantiene el ratón u otro dispositivo señalador sobre el elemento de barra de herramientas.

Utilice la propiedad Name para leer o escribir el valor de los nombres.

Macro

Una macro es una serie de comandos que ejecutan acciones específicas cuando se selecciona un elemento de barra de herramientas. Las macros pueden ser simples registros de pulsaciones de teclas que efectúen una tarea o una combinación compleja de comandos, AutoLISP, DIESEL, o código de programación de ActiveX.

Utilice la propiedad Macro para leer o escribir el valor de las macros.

HelpString

La cadena de ayuda (HelpString) es la secuencia de texto que aparece en la línea de estado de AutoCAD en representación de un botón de barra de herramientas.

Utilice la propiedad HelpString para escribir o leer el valor de una cadena de ayuda.

Index

El índice de un elemento de barra de herramientas determina la posición de dicho elemento en la barra de herramientas a la que pertenece. La primera posición del índice de una barra de herramientas siempre es la 0. Por ejemplo, si un elemento es el primero de una barra de herramientas, tendrá una posición de índice 0; si es el segundo elemento, tendrá la posición de índice 1, y así sucesivamente.

Utilice la propiedad Index para leer o escribir el valor de las posiciones de índice.

Type

Un elemento de una barra de herramientas puede pertenecer a uno de los siguientes tipos: un botón normal de barra de herramientas, un separador, un botón desplegable o un elemento de control especial. Si el elemento es un botón de barra de herramientas normal, esta propiedad devuelve `acButton`. Si el elemento es un separador, esta propiedad devuelve `acToolButtonSeparator`. Si el elemento es un botón desplegable, esta propiedad devuelve `acFlyout`. Si es un elemento de control especial, esta propiedad devuelve `acControl`.

Utilice la propiedad Type para determinar el tipo de elemento de barra de herramientas.

Flyout

Si el elemento de barra de herramientas es del tipo `acFlyout`, esta propiedad devuelve la barra de herramientas que tiene enlazada como barra de herramientas desplegable. La barra de herramientas se devuelve como objeto Toolbar.

Si el elemento de menú no es del tipo `acFlyout`, esta propiedad devuelve `NULL`.

Utilice la propiedad Flyout para localizar la barra de herramientas

desplegable de un elemento de barra de herramientas.

Parent

Esta propiedad devuelve la barra de herramientas donde reside el elemento. La barra de herramientas superior se devuelve como objeto Toolbar.

Utilice la propiedad Parent para localizar la barra de herramientas a la que pertenece un elemento de barra de herramientas.

Toolbar Properties

Hay otras propiedades que se aplican a todos los elementos de la barra de herramientas. Estas propiedades definen si la barra de herramientas está anclada o flotante, si está visible u oculta, o si utiliza botones grandes o pequeños.

Creación de macros

Una macro es una serie de comandos que ejecutan acciones específicas cuando se selecciona un elemento de barra de herramientas. Las macros pueden ser simples registros de pulsaciones de teclas que efectúen una tarea o una combinación compleja de comandos, AutoLISP, DIESEL, o código de programación de ActiveX.

Si pretende incluir parámetros de comando en una macro de menú, debe conocer el orden en que el comando espera sus parámetros. Todos los caracteres de una macro de menú son importantes, inclusive los espacios en blanco. Como AutoCAD ha sido revisado y mejorado, podría cambiar la secuencia de los mensajes de algunos comandos y, en ocasiones, el propio nombre del comando. Por tanto, es posible que los menús personalizados requieran algunos pequeños cambios para actualizarse a la nueva versión de AutoCAD.

Cuando la entrada de un comando proviene de un elemento de menú, se supone que los valores de las variables de sistema PICKADD y PICKAUTO son 1 y 0 respectivamente. Esto mantiene la compatibilidad con versiones anteriores de AutoCAD y facilita la personalización, ya que no es necesario verificar los valores de dichas variables.

Asignación de caracteres de macro a caracteres ASCII equivalentes

En la siguiente tabla se presenta una descripción breve de los caracteres especiales que se utilizan en macros de menús y sus números ASCII equivalentes en VB y VBA. Durante la creación de la cadena de la propiedad Macro, utilice el equivalente ASCII de estos caracteres especiales.

Caracteres especiales utilizados en las macros de menús y barras de herramientas

Carácter	Equivalente ASCII	Descripción
;	chr(59)	Ejecuta INTRO
^M	chr(13)	Ejecuta INTRO
^|	chr(94) + chr(124)	Ejecuta TAB
BARRA ESPACIADORA	chr(32)	Los espacios en blanco entre secuencias de comandos de una opción de menú equivalen a pulsar la BARRA ESPACIADORA.

\	chr(92)	Se detiene para introducción de datos de usuario
_	chr(95)	Traduce los comandos de AutoCAD y las palabras clave que los siguen
+	chr(43)	Continúa la macro de menú en la línea siguiente (si se trata del último carácter)
=*	chr(61) + chr(42)	Muestra el menú de símbolos actual, desplegable o contextual
*^C^C	chr(42) + chr(3) + chr(3)	Prefijo de un elemento de repetición
$	chr(36)	Carga una sección de menú o introduce una expresión de macro condicional de DIESEL.
^B	chr(2)	Activa o desactiva Forzcursor (CTRL+B).
^C	chr(3)	Cancela el comando (CTRL+C).

ESC	chr(3)	Cancela el comando (ESC).
^D	chr(4)	Activa o desactiva coordenadas (CTRL+D).
^E	chr(5)	Define el siguiente plano isométrico (CTRL+E)
^G	chr(7)	Activa o desactiva la rejilla (CTRL+G).
^H	chr(8)	Ejecuta retroceso
^O	chr(15)	Activa o desactiva el modo Orto CTRL+O).
^P	chr(16)	Activa o desactiva MENUECHO.
^Q	chr(17)	Transmite todos los mensajes, listas de estado y entradas a la impresora (CTRL+Q)
^T	chr(20)	Activa o desactiva el tablero (CTRL+T).
^V	chr(22)	Cambia la ventana gráfica actual (CTRL+V).
^Z	chr(26)	Carácter nulo que suprime la adición

	automática de BARRA ESPACIADORA al final de un elemento de menú.
	Terminación de macros

Terminación de la macro

Cuando se ejecuta una macro, AutoCAD coloca un espacio al final de la macro antes de procesar la secuencia de comandos. AutoCAD procesa la siguiente macro de menú como si hubiera escrito **line** BARRA ESPACIADORA.

```
línea
```

Algunas veces, esto no es deseable; por ejemplo, los comandos TEXTO o ACOTA deben terminarse con un INTRO, no con un espacio. Algunas veces es necesario utilizar varios espacios (o INTRO) para terminar un comando, pero algunos editores de texto no permiten crear una línea con espacios en blanco a la derecha. Dos convenciones especiales permiten resolver estos problemas.

- Cuando aparece un punto y coma (;) en una macro, AutoCAD lo sustituye por un INTRO.
- Si una línea termina en un carácter de control, una contrabarra (\), un signo más (+) o un punto y coma (;), AutoCAD no añade un espacio al final de la línea.

Examine la siguiente macro:

```
erase \;
```

Si este elemento terminara simplemente con una contrabarra (lo que indica que el usuario debe introducir datos), la operación ERASE no se efectuaría correctamente, ya que AutoCAD no añade un espacio en blanco después de la contrabarra. Por ello, esta macro de menú utiliza un punto y coma (;) para forzar un INTRO después de los datos introducidos por el usuario. Aquí se muestran otros ejemplos:

```
scp
ucs ;
```

```
text \.4 0 DRAFT Inc;;;Main St.;;;City, State;
```

Al seleccionar la primera macro se introduce **scp** y BARRA ESPACIADORA en la línea de comando, y aparece este mensaje:

Especifique una opción [New/Move/orthoGraphic/Prev/Restore/Save/Del/Apply/?/World] <World>:

Al seleccionar la segunda macro se introduce **scp**, BARRA ESPACIADORA y punto y coma (;) en la línea de comando, y se acepta el valor por defecto, World. En la pantalla no se observa ninguna diferencia entre el primer elemento y el segundo; pero normalmente no se incluirían ambos en el mismo menú.

Al seleccionar la tercera macro, se muestra un mensaje solicitando un punto inicial y se dibuja la dirección en tres líneas. De los tres puntos y comas (;;;), el primero termina la cadena de texto, el segundo vuelve a ejecutar el comando TEXT y el tercero solicita la posición por defecto debajo de la línea anterior.

Nota Todos los caracteres especiales deben especificarse con sus equivalentes ASCII. Para obtener una lista de equivalentes ASCII, véase Asignación de caracteres de macro a caracteres ASCII equivalentes.

Detención para datos de usuario

A veces resulta útil aceptar datos del teclado o del dispositivo señalador en medio de una macro, para lo cual debe colocar una contrabarra (\) en el punto donde deben aparecer los datos.

```
circle \1
layer off \;
```

La primera macro se detiene para pedir al usuario el punto central y, a continuación, lee un radio de valor 1 de la macro. Tenga en cuenta que no hay ningún espacio después del carácter de contrabarra (\). La siguiente macro hace una pausa para pedir al usuario que escriba un nombre de capa, la desactiva y sale del comando CAPA. Normalmente, el comando CAPA pregunta si desea realizar otra operación y sólo sale si se pulsa BARRA ESPACIADORA (espacio) o INTRO (;).

Normalmente, la macro continúa después de introducir un elemento. Por tanto, no es posible construir una macro que acepte un número variable de entradas (como en una selección de objetos) y después continúe. No obstante, el comando DESIGNA es una excepción; una contrabarra detiene la macro hasta que termine la selección de objetos. Por ejemplo, considere esta macro:

```
select \change previous ;properties color red ;
```

Esta macro utiliza el comando DESIGNA para crear un conjunto de selección de uno o varios objetos. A continuación, ejecuta el comando CAMBIA, hace referencia a este conjunto de selección con la opción Previo y cambia el color de todos los objetos seleccionados a rojo.

Dado que el carácter de contrabarra (\) hace que la macro se detenga para recibir datos del usuario, no puede utilizarse para ningún otro fin en una macro. Cuando especifique las rutas de directorio de archivos, utilice una barra oblicua (/) como

delimitador de la ruta: por ejemplo, */directorio/archivo*.

Las siguientes situaciones retrasan la reanudación de una macro:

- Si se espera la entrada de un punto, es posible que le precedan modos de referencia a objetos.
- Cuando se utilizan puntos con filtros X/Y/Z, la macro se interrumpe hasta que se acumula todo el punto.
- Únicamente en el caso del comando DESIGNA, la macro no continúa hasta que termina la designación de objetos.
- Si el usuario responde con un comando transparente, la macro interrumpida permanece en este estado hasta que se termina el comando y se reciben los datos solicitados originalmente.
- Si el usuario responde seleccionando otra macro (para proporcionar opciones o ejecutar un comando transparente), la macro original se interrumpe y el elemento de menú elegido se procesa en su totalidad antes de que la macro interrumpida reanude su ejecución.

Cancelación de comandos

Para cerciorarse de que ninguno de los comandos anteriores queda incompleto, utilice ^C^C en las macros. Esto equivale a pulsar ESC dos veces desde el teclado. Aunque la cadena única ^C cancela la mayoría de los comandos, se necesita ^C^C para regresar a la solicitud Comando desde un comando ACOTA. Por ello, en la mayoría de los casos, ^C^C garantiza el regreso de AutoCAD a la solicitud de comando.

Repetición de macros

Una vez seleccionado un comando, es probable que lo quiera utilizar varias veces antes de pasar a otro comando. Normalmente, el trabajo con las herramientas se realiza de la forma siguiente: se selecciona una herramienta, se realizan varias acciones con ella, a continuación se escoge otra y así sucesivamente. Para no tener que seleccionar la herramienta antes de cada uso, AutoCAD proporciona una función de repetición de comandos, que se activa por una respuesta nula. Sin embargo, no puede utilizar esta función para especificar opciones de comandos.

Esta función permite repetir los comandos de uso más frecuente hasta que se elija otro comando. Si una macro empieza con *^C^C inmediatamente después de la etiqueta del elemento, la macro se guarda en la memoria. La macro responde a las siguientes solicitudes de comando hasta que se pulse ESC o se seleccione otra macro.

No utilice ^C (Cancelar) dentro de una macro que comience por la cadena *^C^C; esto cancelaría la repetición de la macro.

El siguiente es un ejemplo de la función repetitiva o modal, de la manipulación de comandos.

```
*^C^CMOVE Single
*^C^CCOPY Single
*^C^CERASE Single
*^C^CSTRETCH Single Crossing
*^C^CROTATE Single
*^C^CSCALE Single
```

La repetición de macros no funciona en el caso de los menús de símbolos.

Modo de designación de objetos únicos

La designación de objetos únicos activa el modo de designación única, desactiva el diálogo habitual de la designación de objetos y hace que la designación regrese al primer objeto designado por una opción subsiguiente. Esto puede resultar bastante práctico en una macro. Por ejemplo, considere esta macro:

```
*^C^CERASE single
```

Esta macro termina el comando actual y activa el comando BORRA con la opción de designación única. Una vez seleccionado el elemento, es necesario señalar el objeto único que ha de borrarse o apuntar a un área en blanco y especificar una ventana. Los objetos designados de esta forma se borran y la macro se repite (debido al asterisco inicial) para que puedan suprimirse otros elementos. Con el modo de designación única, la interacción con AutoCAD es más dinámica;.

Creación de mensajes de ayuda en la línea de estado para elementos de menús y de barras de herramientas

Un aspecto importante del sistema original de asistencia son los mensajes de ayuda de la línea de estado. Se trata de sencillos mensajes descriptivos que aparecen en la línea de estado al resaltar un elemento de menú o de barra de herramientas. La ayuda de la línea de estado correspondiente a todos los menús y barras de herramientas se encuentra en la propiedad HelpString del propio elemento.

Cuando se crea por primera vez un elemento de menú o de barra de herramientas, la propiedad HelpString está vacía.

Adición de ayuda de línea de estado a un elemento de menú

En este ejemplo se crea un nuevo menú denominado “TestMenu” y a continuación se crea un elemento de menú denominado “Open.” Después se asigna al elemento de menú ayuda de la línea de estado, mediante la propiedad HelpString.

```
Sub Ch6_AddHelp()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Create the new menu
    Dim newMenu As AcadPopupMenu
    Set newMenu = currMenuGroup.Menus.Add _
                              ("Te" + Chr(Asc("&")) + "stMenu")
    ' Add a menu item to the new menu
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String
    ' Assign the macro the VBA equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    ' Create the menu item
    Set newMenuItem = newMenu.AddMenuItem _
```

```
                        (newMenu.count + 1, Chr(Asc("&")) _
                         + "Open", openMacro)
    ' Add the status line help to the menu item
     newMenuItem.HelpString = "Opens an AutoCAD drawing file."
    ' Display the menu on the menu bar
    newMenu.InsertInMenuBar _
                  (ThisDrawing.Application.menuBar.count + 1)
End Sub
```

¿Comentarios?

<$nopage>menú contextual.

Adición de entradas al menú contextual

El menú contextual, o de cursor, es un menú especial incluido en el grupo de menús base de AutoCAD. Este menú aparece al mantener presionada la tecla MAYÚS y hacer clic con el botón derecho del ratón.

Para encontrar el menú contextual, AutoCAD busca en el grupo de menús de base un menú cuya propiedad ShortcutMenu sea TRUE. Para añadir nuevos elementos al menú contextual, siga los pasos enumerados en Adición de nuevos elementos de menú a un menú.

Los nuevos grupos de menús pueden tener o no un menú contextual disponible. Para crear un menú contextual para un grupo de menús, siga las directrices indicadas en Creación de menús nuevos y utilice POP0 como etiqueta del nuevo menú.

Adición de una opción de menú al final del menú contextual

En este ejemplo se añade la opción de menú “OpenDWG” al final del menú contextual.

```
Sub Ch6_AddMenuItemToshortcutMenu()
    Dim currMenuGroup As AcadMenuGroup
    Set currMenuGroup = ThisDrawing.Application.MenuGroups.Item(0)
    ' Find the shortcut menu and assign it to the
    ' shortcutMenu variable
    Dim scMenu As AcadPopupMenu
    Dim entry As AcadPopupMenu
    For Each entry In currMenuGroup.Menus
        If entry.shortcutMenu = True Then
            Set scMenu = entry
        End If
    Next entry
    ' Add a menu item to the shortcut menu
```

```
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String
    ' Assign the macro the VBA equivalent of "ESC ESC _open "
    openMacro = Chr(3) + Chr(3) + "_open "
    Set newMenuItem = scMenu.AddMenuItem _
                                      ("", Chr(Asc("&")) _
                                       + "OpenDWG", openMacro)
End Sub
```

¿Comentarios?

Utilización de eventos

Los eventos son notificaciones o mensajes enviados por AutoCAD para informar al usuario del estado actual de la sesión o para advertirle de algo que ha sucedido. Por ejemplo, cuando se abre un dibujo en AutoCAD se activa el evento BeginOpen. Este evento contiene el nombre del dibujo de AutoCAD. También se activa un evento cuando se cierra un dibujo. Con esta información se puede escribir una subrutina o un controlador de eventos que utilicen estos eventos para registrar la cantidad de tiempo que dedica un usuario a trabajar en un dibujo concreto.

Conceptos básicos sobre los eventos de AutoCAD

Existen tres tipos de eventos en AutoCAD®:

- Los eventos de nivel de aplicación responden a cambios de la aplicación AutoCAD y de su entorno. Estos eventos corresponden a las operaciones de apertura, guardado, cierre e impresión de dibujos, creación de nuevos dibujos, ejecución de comandos de AutoCAD, carga y descarga de aplicaciones ARX y LISP, cambios de las variables del sistema y cambios en la ventana de la aplicación.
- Los eventos de documento responden a cambios de un documento concreto o de su contenido. Estos eventos responden a la adición, eliminación o modificación de objetos, activación de menús contextuales, cambios en el conjunto de selección Pickfirst, cambios en la ventana de dibujo y regeneración del dibujo. También existen eventos de documento para la apertura, el cierre y la impresión de dibujos, así como para la carga y descarga de aplicaciones ARX y LISP desde el dibujo.
- Los eventos de nivel de objeto responden a los cambios de un objeto concreto. En la actualidad sólo hay un evento de objetos. Se activa siempre que se modifica un objeto.

Las subrutinas que responden a eventos se denominan *controladores de eventos* y se ejecutan de forma automática cada vez que se activa su evento correspondiente. La información que contienen los eventos, como el nombre del dibujo en el evento BeginOpen, se transfiere a los controladores de eventos mediante parámetros.

Directrices para los controladores de eventos

Es importante recordar que los eventos sólo proporcionan información sobre el estado o las actividades que tienen lugar en AutoCAD. Aunque pueden escribirse controladores de eventos que respondan a esos eventos, es frecuente que AutoCAD se encuentre procesando comandos cuando se activa el controlador de eventos. Los controladores de eventos, por tanto, tienen ciertas restricciones de seguridad en cuanto a su capacidad de actuación en combinación con AutoCAD y su base de datos.

- No cree dependencias de la secuencia de eventos.

  Al escribir controladores de eventos, no confíe en que el orden de los eventos siga una secuencia exacta que considere lógica. Por ejemplo, si ejecuta un comando ABRIR, se activarán los eventos BeginCommand, BeginOpen, EndOpen y EndCommand. No obstante, es posible que no se produzcan en ese orden. La única secuencia de eventos en la que puede confiar es la de que un evento Begin se producirá antes que un evento End. En el ejemplo anterior los eventos pueden activarse en el orden siguiente: BeginCommand, BeginOpen, EndCommand, y EndOpen, o incluso BeginCommand, EndCommand, BeginOpen, y EndOpen.

- No cree dependencias de la secuencia de operaciones.

  Si suprime objeto1 y, a continuación, objeto2, no confíe en recibir primero el evento ObjectErased del objeto1 y a continuación el del objeto2. Podría recibir en primer lugar el evento ObjectErased del objeto2.

- No realice funciones interactivas desde un controlador de eventos.

  Intentar ejecutar funciones interactivas desde un controlador de eventos podría causar serios problemas ya que AutoCAD podría estar procesando un comando en el momento de activarse el evento. Es, por

tanto, necesario evitar el uso de métodos de entrada-precisión como GetPoint, GetEntity, GetKeyword, etc., de operaciones con conjuntos de selección y del método SendCommand en los controladores de eventos.

- No inicie cuadros de diálogo desde un controlador de eventos.

  Los cuadros de diálogo se consideran funciones interactivas y pueden interferir con la operación actual de AutoCAD. Los cuadros de mensaje y de advertencia no se consideran funciones interactivas y pueden iniciarse sin riesgos; sin embargo, generar un cuadro de mensaje en un controlador de eventos para los eventos BeginModal, EndModal, Activate, Deactivate y BeginRightClick puede producir una secuencia inesperada.

- Puede escribir datos en cualquier objeto de la base de datos, excepto en el objeto que generó el evento.

  Cualquier objeto que activa un evento se puede abrir para utilizarlo con AutoCAD y con la operación que está actualmente en curso. Por tanto, no escriba información en un objeto desde un controlador de eventos del mismo objeto. La lectura de información desde el objeto que activa un evento puede realizarse sin ningún riesgo. Por ejemplo, suponga que tiene un suelo de baldosas y que crea un controlador de eventos asociado al borde del suelo. Si cambia el tamaño del suelo, el controlador de eventos añade o sustrae baldosas de forma automática para rellenar el área nueva. El controlador de eventos podrá leer el nuevo área del borde, pero no podrá realizar ningún cambio en él.

- No realice ninguna acción desde un controlador de eventos que pueda activar el mismo evento.

  Si lleva a cabo la misma acción en un controlador de eventos que activa el mismo evento, creará un bucle infinito. Por ejemplo, no debe nunca intentar abrir un dibujo desde dentro del evento BeginOpen; AutoCAD continuaría abriendo más dibujos hasta que se alcanzara el número máximo de dibujos abiertos posible.

- Recuerde que no pueden activarse eventos mientras AutoCAD está mostrando un cuadro de diálogo modal.

Control de eventos de nivel de aplicación

Los eventos de nivel de aplicación no son permanentes en AutoCAD VBA. Es decir, que no se activan de forma automática al cargar un proyecto VBA Por tanto, es necesario activarlos para VBA y para todos los demás controladores de ActiveX® Automation.

Una vez activados los eventos de nivel de aplicación, tendrá a su disposición un potencial de eventos muy importante. Estos eventos incluyen:

AppActivate

Se inicia justo antes de activar la ventana de la aplicación principal.

AppDeactivate

Se inicia justo antes de desactivar la ventana de la aplicación principal.

ARXLoaded

Se activa después de cargar una aplicación ObjectARX.

ARXUnloaded

Se activa después de descargar una aplicación ObjectARX.

BeginCommand

Se activa justo después de ejecutar un comando, pero antes de su finalización.

BeginFileDrop

Se activa al colocar un archivo en la ventana de la aplicación principal.

BeginLISP

Se activa cuando AutoCAD recibe la solicitud de calcular una expresión LISP.

BeginModal

Se activa justo antes de la apertura de un cuadro de diálogo modal.

BeginOpen

Se activa justo después de que AutoCAD reciba la solicitud de abrir un dibujo existente.

BeginPlot

Se activa justo después de que AutoCAD reciba la solicitud de imprimir un dibujo.

BeginQuit

Se activa justo antes de que finalice una sesión de AutoCAD.

BeginSave

Se activa justo después de que AutoCAD reciba la solicitud de guardar el dibujo.

EndCommand

Se activa justo después de que finalice un comando.

EndLISP

Se activa cuando finaliza la evaluación de una expresión LISP.

EndModal

Se activa justo después del cierre de un cuadro de diálogo modal.

EndOpen

Se activa cuando AutoCAD termina de abrir un dibujo existente.

EndPlot

Se activa después del envío de un documento a la impresora.

EndSave

Se activa cuando AutoCAD termina de guardar el dibujo.

LISPCancelled

Se activa cuando se cancela la evaluación de una expresión LISP.

NewDrawing

Se activa justo antes de la creación de un nuevo dibujo.

SysVarChanged

Se activa cuando se cambia el valor de una variable del sistema.

WindowChanged

Se activa cuando se realiza un cambio en la ventana de la aplicación.

WindowMovedOrResized

Se activa justo después de desplazar o cambiar el tamaño de la ventana de la aplicación.

- **Activación de eventos de nivel de aplicación**

¿Comentarios?

Activación de eventos de nivel de aplicación

Para poder utilizar eventos del nivel de aplicación es necesario crear un nuevo módulo de clase y declarar un objeto del tipo AcadApplicacion con eventos. Por ejemplo, suponga que se ha creado un módulo de clase nuevo denominado ModuloEventoClase. La nueva clase contiene la declaración de la aplicación con la palabra clave de VBA WithEvents.

Para crear una clase nueva y declarar un objeto Application con eventos

1. En VBA IDE, inserte un módulo de clase. En el menú Insertar, elija Módulo de clase.
2. En la ventana Proyecto, elija el módulo de clase que acaba de insertar.
3. En la ventana Propiedades, cambie el nombre de la clase por Modulo-EventoClase.
4. Abra la ventana Código de la clase utilizando F7 o elija Código en el menú Ver.
5. En la ventana de código de la clase, añada la siguiente línea:

```
Public WithEvents App As AcadApplication
```

Después de declararlo, el nuevo objeto con eventos aparece en la lista desplegable de objetos del módulo de clase. Ahora puede escribir procedimientos de evento para el nuevo objeto del módulo de clase. Al seleccionar el nuevo objeto en la lista de objetos, los eventos válidos para dicho objeto aparecerán en la lista desplegable de procedimientos.

Antes de ejecutar los procedimientos, debe conectar el objeto declarado del módulo de clase con el objeto Application. Para ello, puede utilizar el siguiente código desde cualquier módulo.

Para conectar el objeto declarado con el objeto Application

1. En la ventana Código del módulo principal, añada la siguiente línea a la sección de declaraciones:

```
Dim X As New EventClassModule
```

2. En la misma ventana, añada la siguiente subrutina:

```
Sub InitializeEvents()
Set X.App = ThisDrawing.Application
End Sub
```

3. En el código del módulo principal, añada una llamada a la subrutina InitializeEvents:

```
Call InitializeEvents
```

Después de ejecutar el procedimiento InitializeEvents, el objeto App del módulo de clase señala al objeto Application especificado y los procedimientos de evento del módulo de clase se ejecutan siempre que se activa el evento.

Solicitud para continuar cuando se arrastra y suelta un dibujo en AutoCAD

Este ejemplo intercepta el proceso de carga cuando se arrastra un archivo y se suelta en AutoCAD. Se muestra un cuadro de mensaje que contiene el nombre del archivo y los botones Sí, No y Continuar, los cuales permiten al usuario decidir si el archivo debe seguir cargándose o visualizándose. Si el usuario elige cancelar la operación, esta decisión se devuelve mediante el parámetro `Cancel` del evento BeginFileDrop y el archivo no se carga.

```
Public WithEvents ACADApp As AcadApplication
Sub Example_AcadApplication_Events()
 ' This example intializes the public variable (ACADApp)
 ' which will be used to intercept AcadApplication Events
 '
 ' Run this procedure FIRST!
 ' We could get the application from the ThisDocument
 ' object, but that would require having a drawing open,
 ' so we grab it from the system.
 Set ACADApp = GetObject(, "AutoCAD.Application.17")
End Sub
```

```
Private Sub ACADApp_BeginFileDrop _
 (ByVal FileName As String, Cancel As Boolean)
 ' This example intercepts an Application BeginFileDrop event.
 '
 ' This event is triggered when a drawing file is dragged
 ' into AutoCAD.
 '
 ' To trigger this example event:
    ' 1) Make sure to run the example that initializes
 ' the public variable (named ACADApp) linked to this event.
 '
 ' 2) Drag an AutoCAD drawing file into the AutoCAD
 ' application from either the Windows Desktop
 ' or Windows Explorer
 ' Use the "Cancel" variable to stop the loading of the
 ' dragged file, and the "FileName" variable to notify
 ' the user which file is about to be dragged in.
 If MsgBox("AutoCAD is about to load " & FileName & vbCrLf _
 & "Do you want to continue loading this file?", _
 vbYesNoCancel + vbQuestion) <> vbYes Then
 Cancel = True
 End If
End Sub
```

Control de eventos de documento

Los eventos de nivel de aplicación no son permanentes en AutoCAD VBA. Es decir, que no se activan de forma automática al cargar un proyecto VBA No obstante, no están habilitados para ningún otro controlador, como VB. Es necesario, por tanto, activarlos para todos los demás controladores de ActiveX Automation.

Una vez activados los eventos de nivel de aplicación, tendrá a su disposición un potencial de eventos muy importante. Incluye:

Activate

Se inicia al activar una ventana de documento.

BeginDocClose

Se inicia justo después de recibir una solicitud para cerrar un dibujo.

BeginCommand

Se activa justo después de ejecutar un comando, pero antes de su finalización.

BeginDoubleClick

Se activa cuando el usuario hace doble clic en un objeto del dibujo.

BeginLISP

Se activa cuando AutoCAD recibe la solicitud de calcular una expresión LISP.

BeginPlot

Se activa justo después de que AutoCAD reciba la solicitud de imprimir un dibujo.

BeginRightClick

Se activa cuando el usuario hace clic con el botón derecho en la ventana del dibujo.

BeginSave

Se activa justo después de que AutoCAD reciba la solicitud de guardar el dibujo.

BeginShortcutMenuCommand

Se activa después de que el usuario haga clic con el botón derecho en la ventana de dibujo y antes de que aparezca el menú contextual en modo Command.

BeginShortcutMenuDefault

Se activa después de que el usuario haga clic con el botón derecho en la ventana de dibujo y antes de que aparezca el menú contextual en modo Default.

BeginShortcutMenuEdit

Se activa después de que el usuario haga clic con el botón derecho en la ventana de dibujo y antes de que aparezca el menú contextual en modo Edit.

BeginShortcutMenuGrip

Se activa después de que el usuario haga clic con el botón derecho en la ventana de dibujo y antes de que aparezca el menú contextual en modo Grip.

BeginShortcutMenuOsnap

Se activa después de que el usuario haga clic con el botón derecho en la ventana de dibujo y antes de que aparezca el menú contextual en modo Osnap.

Deactivate

Se inicia al desactivar la ventana del dibujo.

EndCommand

Se activa justo después de que finalice un comando.

EndLISP

Se activa cuando finaliza la evaluación de una expresión LISP.

EndPlot

Se activa después del envío de un documento a la impresora.

EndSave

Se activa cuando AutoCAD termina de guardar el dibujo.

EndShortcutMenu

Se activa cuando aparece el menú contextual.

LayoutSwitched

Se activa después de que el usuario cambie de presentación.

LISPCancelled

Se activa cuando se cancela la evaluación de una expresión LISP.

ObjectAdded

Se activa después de añadir un objeto al dibujo.

ObjectErased

Se activa después de borrar un objeto del dibujo.

ObjectModified

Se activa después de modificar un objeto del dibujo.

SelectionChanged

Se activa al cambiar el conjunto de selección Pickfirst.

WindowChanged

Se activa cuando se realiza un cambio en la ventana del documento.

WindowMovedOrResized

Se activa justo después de desplazar o cambiar el tamaño de la ventana del dibujo.

- **Activación de eventos de documento en otros entornos (no VBA)**
- **Programación de eventos de documento en otros entornos (no VBA)**
- **Programación de eventos de documento en VBA**

¿Comentarios?

Activación de eventos de documento en otros entornos (no VBA)

Para poder utilizar eventos de documento en VB u otro entorno que no sea VBA, es necesario crear un nuevo módulo de clase y declarar un objeto del tipo AcadDocument con eventos. Por ejemplo, suponga que se ha creado un módulo de clase nuevo denominado ModuloEventoClase. La nueva clase contiene la declaración de la aplicación con la palabra clave de VBA WithEvents.

Para crear una clase nueva y declarar un objeto Documento con eventos

1. En VBA IDE, inserte un módulo de clase. En el menú Insertar, elija Módulo de clase.
2. En la ventana Proyecto, elija el módulo de clase que acaba de insertar.
3. En la ventana Propiedades, cambie el nombre de la clase por Modulo-EventoClase.
4. Abra la ventana Código de la clase utilizando F7 o elija Código en el menú Ver.
5. En la ventana de código de la clase, añada la siguiente línea:

```
Public WithEvents Doc As AcadDocument
```

Después de declararlo, el nuevo objeto con eventos aparece en la lista desplegable de objetos del módulo de clase. Ahora puede escribir procedimientos de evento para el nuevo objeto del módulo de clase. Al seleccionar el nuevo objeto en la lista de objetos, los eventos válidos para dicho objeto aparecerán en la lista desplegable de procedimientos.

Antes de ejecutar los procedimientos, debe conectar el objeto declarado del

módulo de clase con el objeto Document. Para ello, puede utilizar el siguiente código desde cualquier módulo.

Para conectar el objeto declarado con el objeto Documento

1. En la ventana Código del módulo principal, añada la siguiente línea a la sección de declaraciones:

   ```
   Dim X As New EventClassModule
   ```

2. En la misma ventana, añada la siguiente subrutina:

   ```
   ub InitializeEvents()
    Set X.Doc = ThisDrawing
   End Sub
   ```

3. En el código del módulo principal, añada una llamada a la subrutina `InitializeApp`:

   ```
   Call InitializeEvents
   ```

   Después de ejecutar el procedimiento InitializeEvents, el objeto Doc del módulo de clase señala al objeto de documento creado y los procedimientos de evento del módulo de clase se ejecutan siempre que se activa el evento.

Programación de eventos de documento en otros entornos (no VBA)

Una vez activados los eventos de documento, encontrará la variable de clase Doc en la lista desplegable de objetos del módulo Clase de la ventana de código. Seleccione la clase Doc y aparecerá la lista de eventos disponibles en la lista desplegable de procedimientos. Seleccione el evento para el que desea escribir un controlador y la estructura básica del mismo se creará automáticamente.

Programación de eventos de documento en VBA

Los eventos de documento no se activan de forma automática al cargar un proyecto VBA. Para escribir controladores para eventos de documento en VBA, seleccione AcadDocument en la lista desplegable de objetos de la ventana de código. Los eventos disponibles para el documento aparecen en la lista desplegable de procedimientos. Seleccione el evento para el que desea escribir un controlador y la estructura básica del mismo se creará automáticamente.

Tenga presente que los controladores de eventos creados de esta forma se aplican al dibujo que se encuentre activo. Para crear controladores de eventos para un dibujo específico, primero siga los pasos explicados en Activación de eventos de documento en otros entornos (no VBA). Podrá activar documentos concretos para los eventos.

En el siguiente ejemplo se utiliza el controlador de evento BeginShortcutMenuDefault para añadir la opción de menú “OpenDWG” al principio del menú contextual. Después, el controlador de evento EndShortcutMenu elimina la opción de menú adicional para que no se guarde permanentemente en la configuración de menús del usuario.

```
Private Sub AcadDocument_BeginShortcutMenuDefault _
                    (ShortcutMenu As AutoCAD.IAcadPopupMenu)
    On Error Resume Next
    ' Add a menu item to the cursor menu
    Dim newMenuItem As AcadPopupMenuItem
    Dim openMacro As String
    openMacro = Chr(vbKeyEscape) + Chr(vbKeyEscape) + "_open "
    Set newMenuItem = ShortcutMenu.AddMenuItem _
                                    (0, Chr(Asc("&")) _
                                     + "OpenDWG", openMacro)
End Sub
Private Sub AcadDocument_EndShortcutMenu _
                   (ShortcutMenu As AutoCAD.IAcadPopupMenu)
    On Error Resume Next
```

```
    ShortcutMenu.Item("OpenDWG").Delete
End Sub
```

¿Comentarios?

Control de eventos de nivel de objeto

El evento de nivel de objeto no es permanente en AutoCAD VBA. Es decir, que no se activan de forma automática al cargar un proyecto VBA Por tanto, es necesario activarlo para VBA y para todos los demás controladores de ActiveX Automation.

Una vez activados los eventos de objeto, podrá disponer del evento Modified. Este evento se activa después de modificar un objeto del dibujo.

- **Activación del evento de nivel de objeto**

Activación del evento de nivel de objeto

Para poder utilizar eventos del nivel de objeto es necesario crear un nuevo módulo de clase y declarar un tipo de objeto AcadApplicacion con eventos. Por ejemplo, suponga que se ha creado un módulo de clase nuevo denominado ModuloEventoClase. La nueva clase contiene la declaración de la aplicación con la palabra clave de VBA WithEvents.

Para crear una clase nueva y declarar Círculo como objeto con eventos

1. En VBA IDE, inserte un módulo de clase. En el menú Insertar, elija Módulo de clase.
2. En la ventana Proyecto, elija el módulo de clase que acaba de insertar.
3. En la ventana Propiedades, cambie el nombre de la clase por Modulo-EventoClase.
4. Abra la ventana Código de la clase utilizando F7 o elija Código en el menú Ver.
5. En la ventana de código de la clase, añada la siguiente línea:

```
Public WithEvents Object As AcadCircle
```

Después de declararlo, el nuevo objeto con eventos aparece en la lista desplegable de objetos del módulo de clase. Ahora puede escribir procedimientos de evento para el nuevo objeto del módulo de clase. Al seleccionar el nuevo objeto en la lista de objetos, los eventos válidos para dicho objeto aparecerán en la lista desplegable de procedimientos.

Antes de ejecutar los procedimientos, debe conectar el objeto declarado del módulo de clase con el objeto Circle. Para ello, puede utilizar el siguiente código desde cualquier módulo.

Para conectar el objeto declarado con el objeto Automation:

1. En la ventana Código del módulo principal, añada la siguiente línea a la sección de declaraciones:

```
Dim X As New EventClassModule
```

2. En la misma ventana, cree un círculo denominado “MyCircle” e inicialícelo como si contuviera eventos:

```
Sub InitializeEvents()
 Dim MyCircle As AcadCircle
 Dim centerPoint(0 To 2) As Double
 Dim radius As Double
 centerPoint(0) = 0#: centerPoint(1) = 0#: centerPoint(2) = 0
 radius = 5#
 Set MyCircle = ThisDrawing.ModelSpace.AddCircle(centerPoint,
 Set X.Object = MyCircle
End Sub
```

3. En el código del módulo principal, añada una llamada a la subrutina InitializeApp:

```
Call InitializeEvents
```

Después de ejecutar el procedimiento InitializeEvents, el objeto Circle del módulo de clase señala al objeto de círculo creado; los procedimientos de evento del módulo de clase se ejecutan siempre que se activa el evento.

Nota Cuando cree código en VBA, debe proporcionar un gestor de eventos para todos los objetos del evento Modified. De lo contrario, VBA podría terminar de forma inesperada.

Visualización del área de una polilínea cerrada cada vez que se actualiza

Este ejemplo crea una polilínea optimizada con eventos. Después el gestor del evento de la polilínea muestra el nuevo área que se crea al cambiar la polilínea. Para activar el evento, cambie el tamaño de la polilínea en AutoCAD. Recuerde que debe ejecutar la subrutina CreatePLineWithEvents para que se active el controlador de eventos.

```
Public WithEvents PLine As AcadLWPolyline
Sub CreatePLineWithEvents()
 ' This example creates a light weight polyline
```

```
 Dim points(0 To 9) As Double
 points(0) = 1: points(1) = 1
 points(2) = 1: points(3) = 2
 points(4) = 2: points(5) = 2
 points(6) = 3: points(7) = 3
 points(8) = 3: points(9) = 2
 Set PLine = ThisDrawing.ModelSpace. _
 AddLightWeightPolyline(points)
 PLine.Closed = True
 ThisDrawing.Application.ZoomAll
End Sub
Private Sub PLine_Modified _
 (ByVal pObject As AutoCAD.IAcadObject)
 ' This event is triggered when the polyline is resized.
 ' If the polyline is deleted the modified event is still
 ' triggered, so we use the error handler to avoid
 ' reading data from a deleted object.
 On Error GoTo ERRORHANDLER
 MsgBox "The area of " & pObject.ObjectName & " is: " _
 & pObject.Area
 Exit Sub
ERRORHANDLER:
 MsgBox Err.Description
End Sub
```

¿Comentarios?

Trabajo en espacio tridimensional

La mayor parte de los dibujos constan de vistas bidimensionales (2D) de objetos tridimensionales (3D). Aunque este método de dibujo está muy extendido entre las comunidades de arquitectura y la ingeniería, es limitado: los dibujos son representación en 2D de objetos en 3D, y deben interpretarse visualmente. Además, como las vistas se crean por separado, hay más posibilidades de errores y ambigüedades. Como resultado, tal vez quiera crear modelos 3D reales en lugar de las representaciones 2D. Puede utilizar las herramientas de dibujo de AutoCAD para crear objetos tridimensionales detallados y realistas, así como para manejarlos de diversas formas.

- **Definición de coordenadas 3D**
- **Definición de un sistema de coordenadas personales**
- **Conversión de coordenadas**
- **Creación de objetos 3D**
- **Tareas de edición en 3D**
- **Modificación de sólidos 3D**

Definición de coordenadas 3D

Introducir coordenadas 3D en el sistema de coordenadas universales (SCU) es similar a introducir coordenadas 2D en dicho sistema. Además de especificar los valores *X* e *Y*, el usuario especifica un valor *Z*. Al igual que ocurre con las coordenadas 2D, se utiliza una variante para pasar las coordenadas a los métodos y propiedades ActiveX® y para consultar las coordenadas.

Para obtener más información acerca de la definición de coordenadas 3D, véase "Introducción de coordenadas 3D" en el *Manual del usuario*.

Definición y consulta de coordenadas en polilíneas 2D y 3D

En este ejemplo se crean dos polilíneas, cada una con tres coordenadas. La primera es una polilínea 2D y la segunda 3D. Observe que la longitud de la matriz que contiene los vértices está ampliada para incluir las coordenadas *Z* en la creación de la polilínea 3D. El ejemplo termina con la consulta de las coordenadas de las polilíneas, que se muestran en un cuadro de mensaje.

```
Sub Ch8_Polyline_2D_3D()
    Dim pline2DObj As AcadLWPolyline
    Dim pline3DObj As AcadPolyline
    Dim points2D(0 To 5) As Double
    Dim points3D(0 To 8) As Double
    ' Define three 2D polyline points
   points2D(0) = 1: points2D(1) = 1
   points2D(2) = 1: points2D(3) = 2
   points2D(4) = 2: points2D(5) = 2
    ' Define three 3D polyline points
   points3D(0) = 1: points3D(1) = 1: points3D(2) = 0
   points3D(3) = 2: points3D(4) = 1: points3D(5) = 0
   points3D(6) = 2: points3D(7) = 2: points3D(8) = 0
    ' Create the 2D light weight Polyline
    Set pline2DObj = ThisDrawing.ModelSpace. _
                             AddLightWeightPolyline(points2D)
```

```
    pline2DObj.Color = acRed
    pline2DObj.Update
    ' Create the 3D polyline
    Set pline3DObj = ThisDrawing.ModelSpace. _
                             AddPolyline(points3D)
    pline3DObj.Color = acBlue
    pline3DObj.Update
    ' Query the coordinates of the polylines
    Dim get2Dpts As Variant
    Dim get3Dpts As Variant
    get2Dpts = pline2DObj.Coordinates
    get3Dpts = pline3DObj.Coordinates
    ' Display the coordinates
    MsgBox ("2D polyline (red): " & vbCrLf & _
            get2Dpts(0) & ", " & get2Dpts(1) & vbCrLf & _
            get2Dpts(2) & ", " & get2Dpts(3) & vbCrLf & _
            get2Dpts(4) & ", " & get2Dpts(5))
    MsgBox ("3D polyline (blue): " & vbCrLf & _
            get3Dpts(0) & ", " & get3Dpts(1) & ", " & _
            get3Dpts(2) & vbCrLf & _
            get3Dpts(3) & ", " & get3Dpts(4) & ", " & _
            get3Dpts(5) & vbCrLf & _
            get3Dpts(6) & ", " & get3Dpts(7) & ", " & _
            get3Dpts(8))
End Sub
```

¿Comentarios?

Definición de un sistema de coordenadas personales

Puede definir un sistema de coordenadas personales (SCP) para cambiar el emplazamiento del punto de origen (0, 0, 0) y la orientación del plano *XY* y del eje *Z*. Un SCP se puede colocar y orientar en cualquier punto del espacio tridimensional. Se pueden definir, guardar y utilizar tantos sistemas de coordenadas como se necesiten. La introducción y visualización de las coordenadas depende del sistema SCP que esté activo.

Para indicar el origen y la orientación del SCP, puede mostrar el icono SCP en el punto de origen del SCP mediante la propiedad UCSIconAtOrigin. Si el icono SCP está activado (véase la propiedad UCSIconOn) pero no aparece en el origen, se muestra en la coordenada del SCU definida por la variable de sistema UCSORG.

Puede crear un sistema de coordenadas personales con el método Add. Este método requiere cuatro valores de entrada: la coordenada del origen, una coordenada en los ejes *X* e *Y*, y el nombre del SCP.

Todas las coordenadas de ActiveX Automation de AutoCAD® se introducen en el sistema de coordenadas universales. Utilice el método GetUCSMatrix para volver a la matriz de transformación de un SCP concreto. Utilice esta matriz de transformación para buscar las coordenadas SCU equivalentes.

Para activar un SCP, utilice la propiedad ActiveUCS del objeto Document. Si se realizan cambios en el SCP activo, el nuevo objeto de SCP debe restablecerse como SCP activo para que los cambios se vean. Para restablecer el SCP activo, sólo hay que llamar a la propiedad ActiveUCS de nuevo con el objeto de SCP actualizado.

Para obtener más información sobre la definición del SCP, véase “Control del sistema de coordenadas personales (SCP) en 3D” en el *Manual del usuario*.

Creación de un SCP nuevo, activación y traducción de las coordenadas de un punto a SCP

La siguiente subrutina crea un nuevo SCP y lo establece como el SCP activo del dibujo. A continuación, pide al usuario que designe un punto del dibujo y devuelve las coordenadas SCU y SCP del punto.

```
Sub Ch8_NewUCS()
    ' Define the variables we will need
    Dim ucsObj As AcadUCS
    Dim origin(0 To 2) As Double
    Dim xAxisPnt(0 To 2) As Double
    Dim yAxisPnt(0 To 2) As Double
    ' Define the UCS points
    origin(0) = 4: origin(1) = 5: origin(2) = 3
    xAxisPnt(0) = 5: xAxisPnt(1) = 5: xAxisPnt(2) = 3
    yAxisPnt(0) = 4: yAxisPnt(1) = 6: yAxisPnt(2) = 3
    ' Add the UCS to the
    ' UserCoordinatesSystems collection
    Set ucsObj = ThisDrawing.UserCoordinateSystems. _
             Add(origin, xAxisPnt, yAxisPnt, "New_UCS")
    ' Display the UCS icon
    ThisDrawing.ActiveViewport.UCSIconAtOrigin = True
    ThisDrawing.ActiveViewport.UCSIconOn = True
    ' Make the new UCS the active UCS
    ThisDrawing.ActiveUCS = ucsObj
    MsgBox "The current UCS is : " & ThisDrawing.ActiveUCS.Name _
            & vbCrLf & " Pick a point in the drawing."
    ' Find the WCS and UCS coordinate of a point
    Dim WCSPnt As Variant
    Dim UCSPnt As Variant
    WCSPnt = ThisDrawing.Utility.GetPoint(, "Enter a point: ")
    UCSPnt = ThisDrawing.Utility.TranslateCoordinates _
             (WCSPnt, acWorld, acUCS, False)
    MsgBox "The WCS coordinates are: " & WCSPnt(0) & ", " _
             & WCSPnt(1) & ", " & WCSPnt(2) & vbCrLf & _
             "The UCS coordinates are: " & UCSPnt(0) & ", " _
             & UCSPnt(1) & ", " & UCSPnt(2)
End Sub
```

Conversión de coordenadas

El método TranslateCoordinates convierte un punto o un desplazamiento de un sistema de coordenadas a otro. Un argumento de punto, llamado `OriginalPoint`, puede interpretarse como un punto 3D o como un vector de desplazamiento 3D. Este argumento se distingue por el argumento `Boolean`, `Disp`. Si el argumento `Disp` se define como `TRUE`, el argumento `OriginalPoint` se trata como un vector de desplazamiento; en caso contrario, se trata como un punto. Otros dos argumentos determinan el sistema de coordenadas al que pertenece `OriginalPoint`, así como el sistema de coordenadas al que `OriginalPoint` debe convertirse. Los siguientes sistemas de coordenadas de AutoCAD pueden especificarse en los argumentos `From` y `To`:

SCU

Sistema de coordenadas universales: el sistema de coordenadas de referencia. Todos los demás sistemas de coordenadas se definen en función del SCU, que nunca cambia. Los valores medidos con el SCU permanecen estables aunque se realicen cambios en otros sistemas de coordenadas. Todos los puntos que se transfieren a los métodos y propiedades de ActiveX, o que éstos devuelven, están expresados en el SCU, salvo que se especifique lo contrario.

SCP

Sistema de coordenadas personales (SCP): el sistema de coordenadas de trabajo. El usuario especifica un SCP para crear tareas de dibujos con más facilidad. Todos los puntos que se transmiten a los comandos de AutoCAD, incluidos los devueltos por funciones externas y rutinas de AutoLISP, están en el sistema SCP activo (a menos que el usuario los preceda de un * en la solicitud de comando). Si desea que su aplicación envíe a los comandos de AutoCAD coordenadas en los sistemas SCU, SCO o SCV, primero debe convertirlas al sistema SCP mediante el método TranslateCoordinates.

SCO

Sistema de coordenadas de objeto: los valores de punto especificados por determinados métodos y propiedades para los objetos Polyline y LightweightPolyline se expresan en este sistema de coordenadas relativo al objeto. Dependiendo del uso que se le vaya a dar al objeto, estos puntos suelen convertirse al SCU, al SCP activo o al SCV activo. Del mismo modo, los puntos del SCU, SCP o SCV que vayan a incluirse en la base de datos deben convertirse antes al SCO por medio de las mismas propiedades. Para conocer los métodos y propiedades que usan este sistema de coordenadas, véase *ActiveX and VBA Reference* de AutoCAD.

Cuando convierta coordenadas al SCO, o desde éste, introduzca la normal del SCO en el último argumento de la función TranslateCoordinates.

SCV

Sistema de coordenadas de visualización: el sistema de coordenadas donde se transforman los objetos antes de mostrarse. El origen del SCV es el punto almacenado en la variable de sistema TARGET de AutoCAD y su eje *Z* es la línea de mira. En otras palabras, una ventana gráfica siempre es una vista en planta de su SCV. Pueden utilizarse estas coordenadas para determinar el lugar de la pantalla donde el usuario de AutoCAD verá un objeto concreto.

SCVEP

SCV en espacio papel: este sistema de coordenadas sólo puede transformarse al SCV de la ventana gráfica del espacio modelo actualmente activa y desde el mismo. Se trata básicamente de una transformación 2D, donde las coordenadas *X* e *Y* cambian su escala o se desfasan si el argumento `Disp` es `FALSE`. La coordenada *Z* cambia de escala, pero jamás se convierte. Por lo tanto, puede usarse para hallar el factor de escala entre los dos sistemas de coordenadas. El SCVEP sólo puede transformarse en la ventana gráfica actual del espacio modelo actual. Si el argumento `From` es igual a SCVEP, el argumento `To` debe ser igual a SCV y viceversa.

Conversión de coordenadas SCO en coordenadas SCU

En este ejemplo se crea una polilínea en espacio modelo. Su primer vértice se muestra en los sistemas de coordenadas SCO y SCU. La conversión de SCO a SCU requiere situar la normal del SCO en el último argumento del método TranslateCoordinates.

```
Sub Ch8_TranslateCoordinates()
    ' Create a polyline in model space.
    Dim plineObj As AcadPolyline
    Dim points(0 To 14) As Double
    ' Define the 2D polyline points
    points(0) = 1: points(1) = 1: points(2) = 0
    points(3) = 1: points(4) = 2: points(5) = 0
    points(6) = 2: points(7) = 2: points(8) = 0
    points(9) = 3: points(10) = 2: points(11) = 0
    points(12) = 4: points(13) = 4: points(14) = 0
    ' Create a light weight Polyline object in model space
    Set plineObj = ThisDrawing.ModelSpace.AddPolyline(points)
    ' Find the X and Y coordinates of the
    ' first vertex of the polyline
    Dim firstVertex As Variant
    firstVertex = plineObj.Coordinate(0)
    ' Find the Z coordinate for the polyline
    ' using the elevation property
    firstVertex(2) = plineObj.Elevation
    ' Change the normal for the pline so that the
    ' difference between the coordinate systems
    ' is obvious.
    Dim plineNormal(0 To 2) As Double
    plineNormal(0) = 0#
    plineNormal(1) = 1#
    plineNormal(2) = 2#
    plineObj.Normal = plineNormal
    ' Translate the OCS coordinate into WCS
    Dim coordinateWCS As Variant
    coordinateWCS = ThisDrawing.Utility.TranslateCoordinates _
          (firstVertex, acOCS, acWorld, False, plineNormal)
    ' Display the coordinates of the point
    MsgBox "The first vertex has the following coordinates:" _
           & vbCrLf & "OCS: " & firstVertex(0) & ", " & _
           firstVertex(1) & ", " & firstVertex(2) & vbCrLf & _
           "WCS: " & coordinateWCS(0) & ", " & _
           coordinateWCS(1) & ", " & coordinateWCS(2)
End Sub
```

¿Comentarios?

Creación de objetos 3D

Crear una malla policara Crear un sólido en cuña AutoCAD admite tres tipos de modelado en 3D: alámbrico, de superficie y sólido. Cada uno de ellos se distingue de los demás por sus técnicas de creación y de modificación.

Para obtener más información acerca de la creación de objetos 3D, véase “Creación de objetos 3D” en el *Manual del usuario*.

- **Creación de modelos alámbricos**
- **Creación de mallas**
- **Creación de mallas policara**
- **Creación de sólidos**

Creación de modelos alámbricos

Con AutoCAD se pueden crear modelos alámbricos situando los objetos planos 2D en una ubicación cualquiera de un espacio 3D. Los objetos 2D pueden colocarse en espacios 3D mediante uno de los siguientes métodos:

- Crear el objeto definiendo puntos 3D. Las coordenadas que introduzca definen la posición *X*, *Y* y *Z* del punto.
- Definir el plano de construcción por defecto (*XY*) sobre el que desea dibujar el objeto mediante la definición de un SCP.
- Mover el objeto hasta la posición adecuada en el espacio 3D una vez creado.

También pueden crearse objetos alámbricos, como polilíneas, que pueden existir en las tres dimensiones. Utilice el método Add3DPoly para crear polilíneas 3D.

Para obtener más información acerca de la creación de estructuras alámbricas, véase “Creación de modelos alámbricos” en el *Manual del usuario*.

Creación de mallas

Una malla rectangular (objeto PolygonMesh) representa la superficie de un objeto mediante facetas planas. La densidad de malla, o número de facetas, se define en función de una matriz de vértices *M* y *N*, similar a una rejilla con filas y columnas. Mediante *M* y *N* se determina la posición de las columnas y filas, respectivamente, de cualquier vértice dado. Estos objetos se pueden crear en espacio 2D y 3D, si bien su uso es más apropiado para 3D.

Utilice el método Add3DMesh para crear mallas rectangulares. Este método requiere tres valores de entrada: el número de vértices de la dirección *M*, el número de vértices de la dirección *N* y una matriz de variantes que contenga las coordenadas de todos los vértices de la malla.

Una vez establecido PolygonMesh, utilice las propiedades MClose y NClose para cerrar la malla.

Para obtener más información acerca de la creación de mallas, véase “Creación de superficies” en el *Manual del usuario*.

Creación de una malla poligonal

En este ejemplo se crea una malla poligonal de “. La dirección de la ventana gráfica activa se ajusta de forma que la naturaleza tridimensional de la malla se visualiza con más facilidad.

```
Sub Ch8_Create3DMesh()
    Dim meshObj As AcadPolygonMesh
    Dim mSize, nSize, Count As Integer
    Dim points(0 To 47) As Double
    ' create the matrix of points
    points(0) = 0: points(1) = 0: points(2) = 0
    points(3) = 2: points(4) = 0: points(5) = 1
    points(6) = 4: points(7) = 0: points(8) = 0
```

```
    points(9) = 6: points(10) = 0: points(11) = 1
    points(12) = 0: points(13) = 2: points(14) = 0
    points(15) = 2: points(16) = 2: points(17) = 1
    points(18) = 4: points(19) = 2: points(20) = 0
    points(21) = 6: points(22) = 2: points(23) = 1
    points(24) = 0: points(25) = 4: points(26) = 0
    points(27) = 2: points(28) = 4: points(29) = 1
    points(30) = 4: points(31) = 4: points(32) = 0
    points(33) = 6: points(34) = 4: points(35) = 0
    points(36) = 0: points(37) = 6: points(38) = 0
    points(39) = 2: points(40) = 6: points(41) = 1
    points(42) = 4: points(43) = 6: points(44) = 0
    points(45) = 6: points(46) = 6: points(47) = 0
    mSize = 4: nSize = 4
    ' creates a 3Dmesh in model space
    Set meshObj = ThisDrawing.ModelSpace. _
                        Add3DMesh(mSize, nSize, points)
    ' Change the viewing direction of the viewport
    ' to better see the cylinder
    Dim NewDirection(0 To 2) As Double
    NewDirection(0) = -1
    NewDirection(1) = -1
    NewDirection(2) = 1
    ThisDrawing.ActiveViewport.direction = NewDirection
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
    ZoomAll
End Sub
```

¿Comentarios?

Creación de mallas policara

Utilice el método AddPolyfaceMesh para crear una malla policara en la que las caras puedan tener numerosos vértices.

El proceso de creación de una malla policara es muy similar al empleado para crear una malla rectangular. Para crear una malla policara, indique las coordenadas de todos sus vértices y, a continuación, defina cada cara introduciendo los números de vértice de todos los vértices de esa cara. A medida que va creando la malla policara, puede optar por ocultar determinados lados, asignarlos a capas o asignarles color.

Si desea que los lados sean invisibles, indique el número de vértice correspondiente como un valor negativo. Para obtener más información acerca de la creación de mallas policara, véase el método AddPolyfaceMesh en *ActiveX and VBA Reference*.

Creación de una malla policara

En este ejemplo se crea una malla policara en espacio modelo. Después se actualiza la dirección de visualización de la ventana gráfica activa para permitir una mejor visión de la naturaleza tridimensional de la malla.

```
Sub Ch8_CreatePolyfaceMesh()
    'Define the mesh vertices
    Dim vertex(0 To 17) As Double
    vertex(0) = 4: vertex(1) = 7: vertex(2) = 0
    vertex(3) = 5: vertex(4) = 7: vertex(5) = 0
    vertex(6) = 6: vertex(7) = 7: vertex(8) = 0
    vertex(9) = 4: vertex(10) = 6: vertex(11) = 0
    vertex(12) = 5: vertex(13) = 6: vertex(14) = 0
    vertex(15) = 6: vertex(16) = 6: vertex(17) = 1
    ' Define the face list
    Dim FaceList(0 To 7) As Integer
```

```
    FaceList(0) = 1
    FaceList(1) = 2
    FaceList(2) = 5
    FaceList(3) = 4
    FaceList(4) = 2
    FaceList(5) = 3
    FaceList(6) = 6
    FaceList(7) = 5
    ' Create the polyface mesh
    Dim polyfaceMeshObj As AcadPolyfaceMesh
    Set polyfaceMeshObj = ThisDrawing.ModelSpace.AddPolyfaceMesh _
                                    (vertex, FaceList)
    ' Change the viewing direction of the viewport to
    ' better see the polyface mesh
    Dim NewDirection(0 To 2) As Double
    NewDirection(0) = -1
    NewDirection(1) = -1
    NewDirection(2) = 1
    ThisDrawing.ActiveViewport.direction = NewDirection
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
    ZoomAll
End Sub
```

Creación de sólidos

Un objeto sólido (objeto 3DSolid) representa todo el volumen de un objeto. Los sólidos son probablemente los objetos menos ambiguos y más completos de todos los tipos de modelizado 3D. La creación de formas sólidas complejas es más fácil que la de mallas y representaciones alámbricas.

La creación de sólidos puede llevarse a cabo a partir de una de las formas sólidas básicas como, por ejemplo, un prisma rectangular, un cono, un cilindro, una esfera, un toroide o una cuña, mediante la extrusión de un objeto 2D a lo largo de una trayectoria o mediante su rotación sobre un eje. Utilice uno de los siguientes métodos para crear sólidos:

AddBox, AddCone, AddCylinder, AddEllipticalCone, AddEllipticalCylinder, AddExtrudedSolid, AddExtrudedSolidAlongPath, AddRevolvedSolid, AddSolid, AddSphere, AddTorus, o AddWedge.

Al igual que ocurre con las mallas, los sólidos se muestran en pantalla como representaciones alámbricas hasta que se decide ocultarlos, sombrearlos o modelizarlos. Además, es posible analizar las propiedades físicas de los sólidos (volumen, momentos de inercia, centro de gravedad, etc). Utilice las siguientes propiedades para analizar sólidos: MomentOfInertia, PrincipalDirections, PrincipalMoments, ProductOfInertia, RadiiOfGyration, y Volume.

La propiedad ContourlinesPerSurface controla el número de líneas de triangulación para ver las partes curvas de la representación alámbrica. La propiedad RenderSmoothness controla la suavidad de objetos sombreados y con líneas ocultas.

Para obtener más información acerca de la creación de sólidos, véase “Creación de objetos 3D” en el *Manual del usuario*.

Creación de una cuña sólida

En el siguiente ejemplo se crea un sólido con forma de cuña en espacio modelo. Después se actualiza la dirección de visualización de la ventana gráfica activa para permitir una mejor visión de la naturaleza tridimensional de la cuña.

```
Sub Ch8_CreateWedge()
    Dim wedgeObj As Acad3DSolid
    Dim center(0 To 2) As Double
    Dim length As Double
    Dim width As Double
    Dim height As Double
    ' Define the wedge
    center(0) = 5#: center(1) = 5#: center(2) = 0
    length = 10#: width = 15#: height = 20#
    ' Create the wedge in model space
    Set wedgeObj = ThisDrawing.ModelSpace. _
                   AddWedge(center, length, width, height)
    ' Change the viewing direction of the viewport
    Dim NewDirection(0 To 2) As Double
    NewDirection(0) = -1
    NewDirection(1) = -1
    NewDirection(2) = 1
    ThisDrawing.ActiveViewport.direction = NewDirection
    ThisDrawing.ActiveViewport = ThisDrawing.ActiveViewport
    ZoomAll
End Sub
```

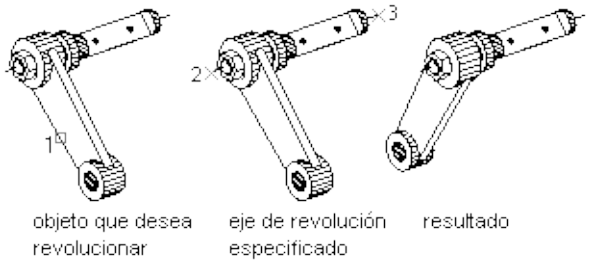
Tareas de edición en 3D

Este apartado trata sobre la realización de diferentes tareas de edición en 3D como girar, disponer en forma de matriz y reflejar en simetría.

- **Rotación de objetos en 3D**
- **Disposición en matriz en 3D**
- **Reflexión en simetría de objetos a lo largo de un plano**

Rotación de objetos en 3D

Con el método Rotate puede girar objetos en 2D alrededor de un punto precisado. La dirección de la rotación está determinada por el SCU. El método Rotate3D gira los objetos en 3D alrededor de un eje precisado. El método Rotate3D requiere tres valores de entrada: las coordenadas SCU de los dos puntos que definen el eje de rotación y el ángulo de rotación en radianes.



Puede girar objetos 3D mediante los métodos Rotate o Rotate3D.

Para obtener más información acerca de la rotación en 3D, véase “Rotación de objetos“ en el *Manual del usuario*.

Creación de un prisma rectangular 3D y rotación sobre un eje

En este ejemplo se crea un prisma rectangular 3D. Después se define el eje de rotación y, por último, se gira el cuadrado 30 grados alrededor del eje.

```
Sub Ch8_Rotate_3DBox()
    Dim boxObj As Acad3DSolid
    Dim length As Double
    Dim width As Double
    Dim height As Double
    Dim center(0 To 2) As Double
    ' Define the box
    center(0) = 5: center(1) = 5: center(2) = 0
    length = 5
```

```
    width = 7
    height = 10
    ' Create the box object in model space
    Set boxObj = ThisDrawing.ModelSpace. _
                 AddBox(center, length, width, height)
    ' Define the rotation axis with two points
    Dim rotatePt1(0 To 2) As Double
    Dim rotatePt2(0 To 2) As Double
    Dim rotateAngle As Double
    rotatePt1(0) = -3: rotatePt1(1) = 4: rotatePt1(2) = 0
    rotatePt2(0) = -3: rotatePt2(1) = -4: rotatePt2(2) = 0
    rotateAngle = 30
    rotateAngle = rotateAngle * 3.141592 / 180#
    ' Rotate the box
    boxObj.Rotate3D rotatePt1, rotatePt2, rotateAngle
    ZoomAll
End Sub
```

¿Comentarios?

Disposición en matriz en 3D

Con el método ArrayRectangular puede crear una matriz rectangular en 3D. Además de especificar el número de columnas (dirección *X*) y filas (dirección *Y*), también puede especificar el número de niveles (dirección *Z*).

Para obtener más información acerca de la utilización de matrices de objetos en 3D, véase “Creación de una matriz de objetos” en el *Manual del usuario*.

Creación de una matriz rectangular 3D

En este ejemplo se crea un círculo y, a continuación, se utiliza para crear una matriz rectangular de cuatro filas, cuatro columnas y tres niveles de círculos.

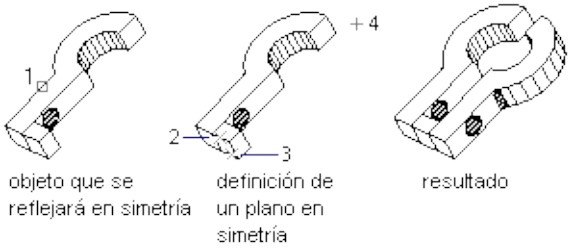
```
Sub Ch8_CreateRectangularArray()
    ' Create the circle
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 2: center(1) = 2: center(2) = 0
    radius = 0.5
    Set circleObj = ThisDrawing.ModelSpace. _
                                AddCircle(center, radius)
    ' Define the rectangular array
    Dim numberOfRows As Long
    Dim numberOfColumns As Long
    Dim numberOfLevels As Long
    Dim distanceBwtnRows As Double
    Dim distanceBwtnColumns As Double
    Dim distanceBwtnLevels As Double
    numberOfRows = 4
    numberOfColumns = 4
    numberOfLevels = 3
    distanceBwtnRows = 1
    distanceBwtnColumns = 1
    distanceBwtnLevels = 4
```

```
    ' Create the array of objects
    Dim retObj As Variant
    retObj = circleObj.ArrayRectangular _
        (numberOfRows, numberOfColumns, _
         numberOfLevels, distanceBwtnRows, _
         distanceBwtnColumns, distanceBwtnLevels)
    ZoomAll
End Sub
```

¿Comentarios?

Reflexión en simetría de objetos a lo largo de un plano

Con el método Mirror3D se pueden reflejar objetos a lo largo de un plano de simetría precisado mediante la definición de tres puntos.



Para obtener más información acerca de la reflexión de objetos en simetría en 3D, véase “Reflejo de objetos” en el *Manual del usuario*.

Reflexión en simetría en 3D

En este ejemplo se crea un prisma rectangular en espacio modelo. Después se refleja en simetría con respecto a un plano y se asigna el color rojo al prisma reflejado.

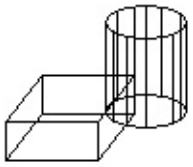
```
Sub Ch8_MirrorABox3D()
    ' Create the box object
    Dim boxObj As Acad3DSolid
    Dim length As Double
    Dim width As Double
    Dim height As Double
    Dim center(0 To 2) As Double
    center(0) = 5#: center(1) = 5#: center(2) = 0
    length = 5#: width = 7: height = 10#
    ' Create the box (3DSolid) object in model space
    Set boxObj = ThisDrawing.ModelSpace. _
                 AddBox(center, length, width, height)
    ' Define the mirroring plane with three points
    Dim mirrorPt1(0 To 2) As Double
```

```
    Dim mirrorPt2(0 To 2) As Double
    Dim mirrorPt3(0 To 2) As Double
    mirrorPt1(0) = 1.25: mirrorPt1(1) = 0: mirrorPt1(2) = 0
    mirrorPt2(0) = 1.25: mirrorPt2(1) = 2: mirrorPt2(2) = 0
    mirrorPt3(0) = 1.25: mirrorPt3(1) = 2: mirrorPt3(2) = 2
    ' Mirror the box
    Dim mirrorBoxObj As Acad3DSolid
    Set mirrorBoxObj = boxObj.Mirror3D _
                (mirrorPt1, mirrorPt2, mirrorPt3)
    mirrorBoxObj.Color = acRed
    ZoomAll
End Sub
```

¿Comentarios?

Modificación de sólidos 3D

Una vez creado un sólido, puede proceder a la creación de formas sólidas más complejas mediante la combinación de distintos objetos sólidos. Puede optar por unir sólidos, sustraerlos o localizar su volumen común (partes superpuestas). Utilice el método Boolean o CheckInterference para efectuar dichas combinaciones.



Los sólidos se pueden modificar también mediante la obtención de la sección transversal bidimensional de un sólido o el corte de un sólido en dos partes. Utilice el método SectionSolid para buscar secciones transversales de sólidos, y el método SliceSolid para cortar un sólido en dos partes.

Búsqueda de la interferencia entre dos sólidos

En este ejemplo se crea un prisma rectangular y un cilindro en espacio modelo. A continuación, se localiza la interferencia entre los dos sólidos y se crea un sólido nuevo a partir de ella. Para facilitar la visualización, el prisma se colorea en blanco, el cilindro en cián y el sólido de interferencia en rojo.

```
Sub Ch8_FindInterferenceBetweenSolids()
    ' Define the box
    Dim boxObj As Acad3DSolid
    Dim length As Double
    Dim width As Double
    Dim height As Double
    Dim center(0 To 2) As Double
```

```
    center(0) = 5: center(1) = 5: center(2) = 0
    length = 5
    width = 7
    height = 10
    ' Create the box object in model space
    ' and color it white
    Set boxObj = ThisDrawing.ModelSpace. _
                 AddBox(center, length, width, height)
    boxObj.Color = acWhite
    ' Define the cylinder
    Dim cylinderObj As Acad3DSolid
    Dim cylinderRadius As Double
    Dim cylinderHeight As Double
    center(0) = 0: center(1) = 0: center(2) = 0
    cylinderRadius = 5
    cylinderHeight = 20
    ' Create the Cylinder and
    ' color it cyan
    Set cylinderObj = ThisDrawing.ModelSpace.AddCylinder _
                      (center, cylinderRadius, cylinderHeight)
    cylinderObj.Color = acCyan
    ' Find the interference between the two solids
    ' and create a new solid from it. Color the
    ' new solid red.
    Dim solidObj As Acad3DSolid
    Set solidObj = boxObj.CheckInterference(cylinderObj, True)
    solidObj.Color = acRed
    ZoomAll
End Sub
```

Corte de un sólido en dos sólidos

En este ejemplo se crea un prisma rectangular en espacio modelo. Después se corta tomando como referencia un plano definido por tres puntos. La sección se devuelve como sólido 3D.

```
Sub Ch8_SliceABox()
    ' Create the box object
    Dim boxObj As Acad3DSolid
    Dim length As Double
    Dim width As Double
    Dim height As Double
    Dim center(0 To 2) As Double
    center(0) = 5#: center(1) = 5#: center(2) = 0
    length = 5#: width = 7: height = 10#
    ' Create the box (3DSolid) object in model space
    Set boxObj = ThisDrawing.ModelSpace. _
```

```
                AddBox(center, length, width, height)
    boxObj.Color = acWhite
    ' Define the section plane with three points
    Dim slicePt1(0 To 2) As Double
    Dim slicePt2(0 To 2) As Double
    Dim slicePt3(0 To 2) As Double
    slicePt1(0) = 1.5: slicePt1(1) = 7.5: slicePt1(2) = 0
    slicePt2(0) = 1.5: slicePt2(1) = 7.5: slicePt2(2) = 10
    slicePt3(0) = 8.5: slicePt3(1) = 2.5: slicePt3(2) = 10
    ' slice the box and color the new solid red
    Dim sliceObj As Acad3DSolid
    Set sliceObj = boxObj.SliceSolid _
                   (slicePt1, slicePt2, slicePt3, True)
    sliceObj.Color = acRed
    ZoomAll
End Sub
```

¿Comentarios?

Definición de presentaciones e impresión

Una vez creado el dibujo con AutoCAD, habitualmente se traza en papel. Un dibujo trazado puede contener una única vista del dibujo o varias, dotándolo de una organización más compleja. En espacio papel, se pueden crear ventanas gráficas flotantes, que muestran distintas vistas del dibujo. Según sea necesario, se puede imprimir una o varias ventanas gráficas o establecer las opciones que determinen lo que debe trazarse y cómo debe ajustarse la imagen al papel.

El espacio modelo y el espacio papel

El espacio modelo es el entorno de dibujo donde el usuario crea la geometría del modelo. Por lo general, al comenzar un dibujo en el espacio modelo se designan los límites del dibujo para determinar su extensión, y se dibuja a escala real.

El espacio papel representa el modelo conforme quedará trazado en papel. En el espacio papel es posible presentar distintas vistas del dibujo y vistas a escala independientes entre sí, y disponer las distintas vistas del dibujo como se desea que se tracen. El mismo dibujo puede tener distintas representaciones de espacio papel.

Presentaciones

Las presentaciones contienen toda la geometría del dibujo. La geometría del espacio modelo se incluye en una presentación denominada Modelo. Ni se puede cambiar el nombre de la presentación del espacio modelo, ni se puede crear más de una presentación del espacio modelo. Sólo puede haber una presentación de espacio modelo por dibujo.

La geometría del espacio papel también se incluye en las presentaciones. Es posible tener varias presentaciones de espacio papel para cada dibujo. Cada una representa una configuración de impresión distinta. El nombre de las presentaciones del espacio papel se puede modificar.

En ActiveX® Automation, toda la geometría de la presentación del espacio modelo está incluida en la colección ModelSpace. Como en un dibujo puede haber más de una presentación del espacio papel, la colección PaperSpace apunta a la última presentación del espacio papel activa.

Para obtener más información acerca de las presentaciones de espacio papel, véase “Sombreados, rellenos y coberturas” en el *Manual del usuario*.

- **Presentaciones y bloques**
- **Configuraciones de impresión**
- **Parámetros de presentación**

Presentaciones y bloques

El contenido de cualquier presentación se distribuye entre dos objetos ActiveX diferentes: el objeto Layout y el objeto Block. El objeto Layout contiene los ajustes de trazado y las propiedades visuales de la presentación tal como aparece en la interfaz de usuario. El objeto Block contiene la geometría de la presentación.

Cada objeto Layout está asociado a un solo objeto Block. Si desea acceder al objeto Block asociado a una presentación determinada, utilice la propiedad Block. A su vez, cada objeto Block está asociado a un solo objeto Layout. Para acceder al objeto Layout asociado a un bloque determinado, utilice la propiedad Layout del bloque.

Configuraciones de impresión

Los objetos PlotConfiguration se parecen a los objetos Layout en que contienen la misma información de impresión. Se diferencian en que el objeto Layout tiene asociado un objeto Block que contiene la geometría que debe imprimirse. Los objetos PlotConfiguration no se asocian a un objeto Block particular. Simplemente, es un conjunto guardado de configuraciones de trazadores que se puede utilizar con cualquier geometría.

Parámetros de presentación

Los parámetros de presentación determinan la salida impresa. Estos parámetros afectan al tamaño de papel, la escala de impresión, el área de impresión, el origen de impresión y el nombre del dispositivo de impresión. Comprender el modo de utilizar los parámetros de presentación garantiza que la presentación se trazará de la forma esperada. Todos los parámetros de una presentación pueden cambiarse por medio de las propiedades y los métodos del objeto Layout.

- **Tamaño y unidades de papel**
- **Ajuste del origen de impresión**
- **Definición del área de trazado**
- **Establecimiento de la escala de impresión**
- **Definición de la escala del grosor de línea**
- **Configuración del dispositivo de impresión**

Tamaño y unidades de papel

La elección del tamaño del papel depende del trazador configurado para el sistema. Cada modelo de trazador tiene su propia lista estándar de tamaños de papel disponibles. El tamaño del papel de una presentación se puede cambiar por medio de la propiedad CanonicalMediaName.

También se pueden especificar las unidades de la presentación mediante la propiedad PaperUnits. La propiedad requiere tres valores de entrada: `acInches`, `acMillimeters` o `acPixels`. Si el trazador se encuentra configurado para salida ráster, deberá precisar el tamaño de salida en píxeles.

Ajuste del origen de impresión

El trazado comienza en la esquina inferior izquierda del área de impresión indicada y se controla con la propiedad PlotOrigin. Habitualmente, el origen de trazado es (0, 0). No obstante, es posible centrar el trazado en la hoja de papel, asignando a la propiedad CenterPlot el valor TRUE. Al centrarse el trazado se modifica su origen.

Definición del área de trazado

Al preparar el trazado de una presentación, se puede especificar el área de impresión para determinar qué se incluirá en la impresión. Para precisar el área del trazado, utilice la propiedad PlotType Esta propiedad requiere la entrada de uno de los siguientes valores:

acDisplay

Imprime todo lo que se encuentra en la pantalla del espacio modelo actual. Esta opción no está disponible cuando se imprime desde una presentación de espacio papel.

acExtents

Imprime todo lo que se encuentra dentro del contorno del espacio seleccionado.

acLimits

Imprime todo lo que está dentro de los límites del espacio actual.

acView

Imprime la vista guardada por la propiedad ViewToPlot.

acWindow

Imprime todo lo que está en la ventana especificada en el método SetWindowToPlot.

acLayout

Imprime todo lo que se encuentra dentro de los márgenes del tamaño de papel especificado. Esta opción no se encuentra disponible cuando se imprime desde el espacio modelo.

Cuando se crea una nueva presentación de espacio papel, la opción por defecto es `acLayout`.

¿Comentarios?

Establecimiento de la escala de impresión

Por lo general, los objetos se dibujan en su tamaño real. Cuando se imprime el dibujo, se le atribuye una escala o bien se ajusta la imagen a la página. Para precisar una escala, introduzca una escala de trazado predefinida o personalizada.

Para introducir una escala estándar, primero establezca la propiedad UseStandardScale como `TRUE`. Luego puede escribir la escala deseada utilizando la propiedad StandardScale.

Para escribir una escala personalizada, primer defina la propiedad UseStandardScale como `FALSE`. Luego puede introducir la escala personalizada utilizando el método SetCustomScale.

Durante la fase de revisión de una vista de dibujo, la aplicación de la escala precisa no siempre es importante. Puede utilizar el valor `acScaleToFit` de la propiedad StandardScale para imprimir la presentación con el máximo tamaño que permita el formato del papel.

Definición de la escala del grosor de línea

Se puede ajustar la escala de los grosores de línea (lineweight) de forma proporcional a la escala de trazado en las presentaciones. Normalmente, los grosores de línea precisan el ancho de línea de los objetos trazados, y se trazan con este ancho de línea independientemente de la escala de trazado. La mayor parte de las veces, se utiliza la escala de trazado por defecto 1:1 al trazar una presentación. Sin embargo, si desea trazar una presentación tamaño E que se haya ajustado a escala para una hoja de papel tamaño A, puede especificar que los grosores de línea se ajusten a escala en proporción con la nueva escala de trazado.

Para ajustar la escala de los grosores de línea, asigne a la propiedad ScaleLineweights el valor TRUE. Si no desea ajustar la escala, asigne a esta propiedad el valor FALSE.

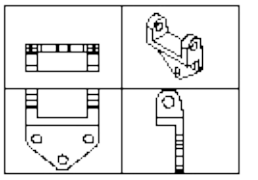
Configuración del dispositivo de impresión

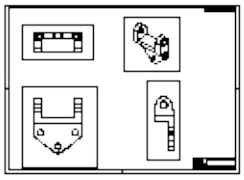
El nombre del dispositivo de trazado se especifica en la propiedad ConfigName. Puede definir este nombre para cualquier nombre de dispositivo válido de su sistema. Si no se establece la propiedad, el trazado se envía al dispositivo predeterminado del sistema.

Ventanas gráficas

En el espacio modelo, la geometría se dibuja en ventanas gráficas dispuestas en mosaico (citadas en ActiveX Automation como objetos Viewport). En la pantalla se pueden tener abiertas una o más de estas ventanas. Cuando hay varias en pantalla, las modificaciones que se realicen en una afectan a todas las demás. Sin embargo, los parámetros de ampliación, punto de vista, rejilla y resolución sí pueden definirse en cada ventana por separado.

En el espacio papel se trabaja en ventanas gráficas de espacio papel flotantes (mencionadas en ActiveX Automation como objetos PViewport) que incluyen distintas vistas del modelo. Las ventanas flotantes, que reciben el mismo tratamiento que cualquier objeto, pueden desplazarse o variar de tamaño y forma para presentar el dibujo como convenga. También es posible dibujar objetos, como anotaciones o cuadros de rotulación, directamente en la vista de espacio papel sin que afecte al modelo.


Ventanas gráf. en mosaico


Ventanas gráf. flotantes

Para obtener más información acerca de ventanas gráficas, consulte “Definición de las ventanas gráficas del espacio modelo” y

“Presentación de varias vistas en espacio modelo” en el *Manual del usuario*.

- **Ventanas flotantes**
- **Cambio a una presentación de espacio papel**
- **Cambio a una presentación de espacio modelo**
- **Creación de ventanas gráficas de espacio papel.**

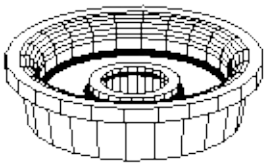
- **Modificación de vistas y contenido de las ventanas gráficas**
- **Ajuste de escala de las vistas en relación con el espacio papel**
- **Ajuste de escala de tipos de línea de patrones en espacio papel**
- **Uso de ventanas gráficas sombreadas**

¿Comentarios?

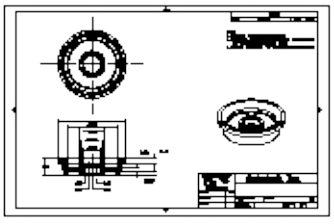
Ventanas flotantes

El modelo no puede modificarse en espacio papel. Para acceder al modelo de un objeto PViewport, utilice la propiedad ActiveSpace para cambiar del espacio papel al espacio modelo. De esta forma podrá trabajar con el modelo sin perder de vista su presentación global. En objetos PViewport, las capacidades de edición y cambio de visualización son casi las mismas que en objetos Viewport. Sin embargo, tiene más control sobre las vistas individuales. Por ejemplo, se puede inutilizar o desactivar capas en algunas ventanas gráficas de presentación sin que las otras se vean afectadas. Se puede activar y desactivar la visualización de una ventana gráfica entera. También es posible alinear vistas entre ventanas gráficas y ajustar sus escalas en relación con la presentación global.

La ilustración siguiente muestra las diferentes vistas de un modelo que se pueden presentar en espacio papel. Las imágenes de espacio papel representan distintas vistas de un objeto PViewport. En una de las vistas, la capa de cotas está inutilizada. Observe que el cuadro de título, el marco y la anotación, dibujados en espacio papel, no aparecen en la vista del espacio modelo. Además, la capa que contiene el marco de la ventana gráfica está desactivada.



el modelo



el modelo mostrado en ventanas flotantes

Cuando se trabaja en un objeto Viewport, la propiedad ActiveSpace debe tener siempre el valor `acModelSpace`. Cuando se trabaja en un objeto PViewport, la propiedad ActiveSpace puede tener el valor `acModelSpace` o `acPaperSpace`, por lo que es posible cambiar entre el espacio papel y el

espacio modelo según las necesidades.

Parámetros de los objetos PViewport y Viewport, y de la propiedad ActiveSpace

Tipo de ventana gráfica	Estado	Descripción
PViewport	ActiveSpace = `acPaperspace`	Disposición de la presentación en ventanas gráficas flotantes y posibilidad de añadir cuadros de rotulación, marcos y anotaciones. La edición no afecta al modelo.
PViewport	ActiveSpace = `acModelspace`	Trabajo con ventanas gráficas flotantes para modificar el modelo o cambiar las vistas. Se pueden inutilizar o desactivar capas determinadas en las distintas ventanas gráficas.
Ventana gráfica	ActiveSpace = `acModelspace`	División de la pantalla en ventanas gráficas en mosaico para editar varias vistas del modelo.

En ActiveX Automation de AutoCAD®, la propiedad ActiveSpace se utiliza para controlar la variable de sistema TILEMODE. Establecer `ThisDrawing.ActiveSpace = acModelSpace` es equivalente a

establecer TILEMODE = como activado, y establecer ThisDrawing.ActiveSpace = acPaperSpace es equivalente a establecer TILEMODE = como desactivado.

De igual forma, la propiedad MSpace equivale a los comandos MSPACE y PSPACE de AutoCAD. Establecer ThisDrawing.MSpace = TRUE es igual a utilizar el comando MSPACE: conmuta a espacio modelo. Establecer ThisDrawing.MSpace = FALSE es igual que utilizar el comando PSPACE: conmuta a espacio papel.

Por otro lado, se requiere el uso del método Display antes de definir TRUE en la propiedad MSpace. El método Display inicializa determinados parámetros gráficos que deben fijarse antes de cambiar al espacio modelo. En AutoCAD, esto se lleva a cabo “entre bastidores. ” Sin embargo, en la interfaz de ActiveX Automation, el programador es el responsable de esta inicialización.

Nota Recuerde que, para poder asignar TRUE a la propiedad MSpace, debe haber activado la visualización de al menos un objeto PViewport con el método Display. Si no se activa la visualización, se producirá un error cuando se intente definir la propiedad MSpace.

Cambio a una presentación de espacio papel

Desde el espacio modelo se puede cambiar a la última presentación del espacio papel activo.

Para cambiar a la última presentación del espacio papel activo

1. Asigne a la propiedad ActiveSpace el valor `acPaperSpace`:

```
ThisDrawing.ActiveSpace = acPaperSpace
```

2. Conmute la propiedad MSpace a `FALSE`:

```
ThisDrawing.MSpace = FALSE
```

Cuando se está en el espacio papel, AutoCAD muestra el icono del sistema de coordenadas personales (SCP) del espacio papel en la esquina inferior izquierda del área gráfica. El cursor en cruz indica que se pueden realizar cambios en el área de presentación de espacio papel (no en las vistas de las ventanas gráficas).

Cambio a una presentación de espacio modelo

Desde el espacio papel se puede pasar a las ventanas gráficas flotantes o en mosaico del espacio modelo.

Para cambiar a las ventanas gráficas flotantes

1. Utilice el método Display para inicializar los parámetros gráficos:

```
ThisDrawing.ActivePViewport.Display TRUE
```

2. Conmute la propiedad MSpace a `TRUE`:

```
ThisDrawing.MSpace = TRUE
```

Se encontrará en el espacio modelo, con ventanas flotantes.

Nota Debe crear ventanas flotantes antes de intentar cambiar al espacio modelo.

Para cambiar a las ventanas gráficas en mosaico

Para cambiar a las ventanas gráficas en mosaico, efectúe este paso adicional:

- Asigne a la propiedad ActiveSpace el valor `acModelSpace`:

```
ThisDrawing.ActiveSpace = acModelSpace
```

Creación de ventanas gráficas de espacio papel.

Las ventanas gráficas de espacio papel se crean con el método AddPViewport. Este método requiere un punto central y la anchura y altura de la nueva ventana gráfica. Antes de crear la ventana gráfica, utilice la propiedad ActiveSpace para especificar el espacio papel como espacio actual (normalmente se hace estableciendo TILEMODE en 0).

Después de crear el objeto PViewport se pueden configurar las propiedades de la vista, como la dirección de la vista (propiedad Direction), la longitud de la lente para vistas en perspectiva (propiedad LensLength) y la presentación de la rejilla (propiedad GridOn). También se pueden controlar propiedades de la ventana gráfica, como la capa (propiedad Layer), el tipo de línea (propiedad Linetype) y la escala del tipo de línea (propiedad LinetypeScale).

Creación y activación de una ventana gráfica flotante

En este ejemplo se conmuta AutoCAD a espacio papel, se crea una ventana flotante, se configura la vista y se activa la ventana.

```
Sub Ch9_SwitchToPaperSpace()
   ' Set the active space to paper space
    ThisDrawing.ActiveSpace = acPaperSpace
   ' Create the paperspace viewport
    Dim newVport As AcadPViewport
    Dim center(0 To 2) As Double
    center(0) = 3,25
    center(1) = 3
    center(2) = 0
    Set newVport = ThisDrawing.PaperSpace. _
                                AddPViewport(center, 6, 5)
   ' Change the view direction for the viewport
    Dim viewDir(0 To 2) As Double
    viewDir(0) = 1
    viewDir(1) = 1
```

```
    viewDir(2) = 1
    newVport.direction = viewDir
   ' Enable the viewport
    newVport.Display True
   ' Switch to model space
    ThisDrawing.MSpace = True
   ' Set newVport current
   ' (not always necessary but a good idea)
    ThisDrawing.ActivePViewport = newVport
   ' Zoom Extents in model space
    ZoomExtents
   ' Turn model space editing off
    ThisDrawing.MSpace = False
    ' ZoomExtents in paperspace
    ZoomExtents
End Sub
```

El orden de los pasos del código de este ejemplo es muy importante. Como norma, las sentencias deben establecerse en el mismo orden en el que se haría desde la línea de comando de AutoCAD. Las únicas acciones inesperadas son la definición de la vista y la activación de la ventana gráfica.

Nota Para configurar o modificar aspectos de la vista (dirección, distancia focal, etc.), el método Display del objeto Viewport debe estar desactivado (FALSE); para establecer una ventana gráfica como actual, el método Display debe estar activado (TRUE).

Creación de cuatro ventanas flotantes

En este ejemplo se amplía el ejemplo de "Crear y activar una ventana gráfica flotante" para crear cuatro ventanas flotantes y establecer sus vistas en superior, frontal, derecha e isométrica, respectivamente. La escala de las vistas se ajusta a la mitad de la escala del espacio papel. Para asegurarse de que estas ventanas tienen algo que mostrar, puede crear una esfera sólida 3D antes de probar el ejemplo.

```
Sub Ch9_FourPViewports()
    Dim topVport, frontVport As AcadPViewport
    Dim rightVport, isoVport As AcadPViewport
    Dim pt(0 To 2) As Double
    Dim viewDir(0 To 2) As Double
    ThisDrawing.ActiveSpace = acPaperSpace
    ThisDrawing.MSpace = True
' Take the existing PViewport and make it the topVport
    pt(0) = 2.5: pt(1) = 5.5: pt(2) = 0
```

```
    Set topVport = ThisDrawing.ActivePViewport
    'No need to set Direction for top view
    topVport.center = pt
    topVport.width = 2.5
    topVport.height = 2.5
    topVport.Display True
    ThisDrawing.MSpace = True
    ThisDrawing.ActivePViewport = topVport
    ZoomExtents
    ZoomScaled 0.5, acZoomScaledRelativePSpace
'Create and setup frontVport
    pt(0) = 2.5: pt(1) = 2.5: pt(2) = 0
    Set frontVport = ThisDrawing.PaperSpace. _
            AddPViewport(pt, 2.5, 2.5)
    viewDir(0) = 0: viewDir(1) = 1: viewDir(2) = 0
    frontVport.direction = viewDir
    frontVport.Display acOn
    ThisDrawing.MSpace = True
    ThisDrawing.ActivePViewport = frontVport
    ZoomExtents
    ZoomScaled 0.5, acZoomScaledRelativePSpace
'Create and setup rightVport
    pt(0) = 5.5: pt(1) = 5.5: pt(2) = 0
    Set rightVport = ThisDrawing.PaperSpace. _
            AddPViewport(pt, 2.5, 2.5)
    viewDir(0) = 1: viewDir(1) = 0: viewDir(2) = 0
    rightVport.direction = viewDir
    rightVport.Display acOn
    ThisDrawing.MSpace = True
    ThisDrawing.ActivePViewport = rightVport
    ZoomExtents
    ZoomScaled 0.5, acZoomScaledRelativePSpace
'Create and set up isoVport
    pt(0) = 5.5: pt(1) = 2.5: pt(2) = 0
    Set isoVport = ThisDrawing.PaperSpace. _
                             AddPViewport(pt, 2.5, 2.5)
    viewDir(0) = 1: viewDir(1) = 1: viewDir(2) = 1
    isoVport.direction = viewDir
    isoVport.Display acOn
    ThisDrawing.MSpace = True
    ThisDrawing.ActivePViewport = isoVport
    ZoomExtents
    ZoomScaled 0.5, acZoomScaledRelativePSpace
'Finish: Perform a regen in all viewports
    ThisDrawing.Regen True
End Sub
```

Modificación de vistas y contenido de las ventanas gráficas

Para cambiar la vista de un objeto Viewport, debe encontrarse en el espacio modelo y tener activa la ventana gráfica.

Para editar un dibujo en una ventana gráfica flotante

1. En el espacio modelo, utilice la propiedad ActiveViewport para activar la ventana gráfica:

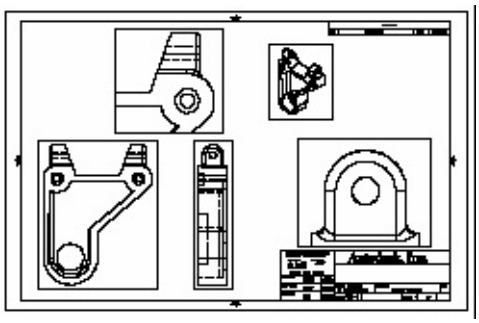
   ```
   Thisdrawing.ActiveViewport = MyViewportObject
   ```

2. Modificación del dibujo.

También puede crear objetos, como anotaciones, cotas o cuadros de rotulación, en el espacio papel. No obstante, es necesario asignar a la propiedad ActiveSpace el valor FALSE y activar el espacio papel por medio de la propiedad MSpace. Los objetos que se crean en espacio papel sólo son visibles en el espacio papel.

Ajuste de escala de las vistas en relación con el espacio papel

Antes de imprimir, se pueden establecer factores de escala de ampliación exactos para cada sección del dibujo. Al ajustar la escala de las vistas en relación con el espacio papel, se establece una escala sistemática para cada una de las vistas. Por ejemplo, la siguiente ilustración muestra una vista en espacio papel con varias ventanas gráficas, en las que se ofrecen vistas distintas a escalas diferentes. Para ajustar la escala del dibujo trazado con precisión, es imprescindible que se ajusten las escalas de las vistas en relación con el espacio papel, y no en función de la vista anterior ni del tamaño real del modelo.



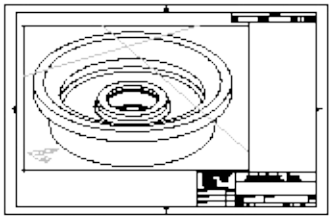
Cuando se trabaja en espacio papel, el factor de escala indica la relación entre el tamaño del dibujo trazado y el tamaño real del modelo exhibido en las ventanas gráficas. Para obtener esta escala, se dividen las unidades de espacio papel por las unidades de espacio modelo. Para un dibujo en escala de un cuarto, por ejemplo, se especifica un factor de escala de una unidad de espacio papel por cuatro unidades de espacio modelo (1:4).

Utilice el método ZoomScaled para atribuir una escala a las ventanas gráficas en relación con las unidades de espacio papel. Este método requiere tres valores de entrada: la ventana gráfica que vamos a escalar, el factor de escala y cómo queremos que se aplique dicho factor. Este tercer valor es optativo y determina cómo se aplica la escala:

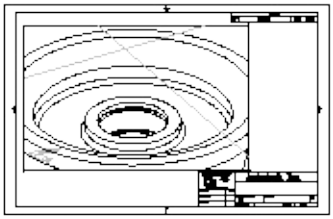
- En relación con los límites del dibujo
- En relación con la vista actual
- En relación con las unidades del espacio papel

Para atribuir una escala a una vista en relación con las unidades de espacio papel, introduzca la constante `acZoomScaled-RelativePSpace` para este valor.

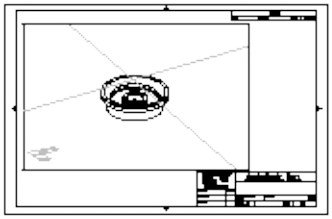
Como muestran las ilustraciones, una escala de 2 en relación con las unidades de espacio papel, la escala de la ventana gráfica aumenta al doble el tamaño de las unidades de espacio papel. Una escala de 0,5 relativa a las unidades de espacio papel reduce a la mitad el tamaño de las unidades de espacio papel. El modelo se imprime a la mitad de su tamaño real.



vista actual



ampliación a 2xp

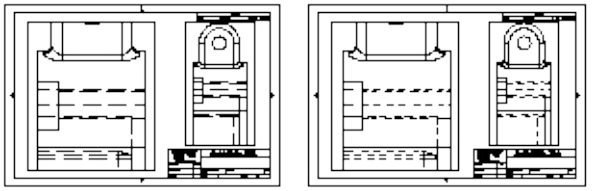


ampliación a 0,5xp

Ajuste de escala de tipos de línea de patrones en espacio papel

En espacio papel, hay dos formas de atribuir escala a un tipo de línea. La escala se puede basar en las unidades de dibujo del espacio en que se creó el objeto (modelo o papel), o puede tratarse de una escala uniforme basada en unidades de espacio papel. La escala de tipo de línea también puede ser una escala uniforme basada en unidades de espacio papel. La variable de sistema PSLTSCALE permite mantener la misma escala de los tipos de línea para los objetos mostrados a diferentes escalas en distintas ventanas gráficas. Afecta también a la presentación de las líneas en las vistas 3D.

En la siguiente ilustración se ha ajustado uniformemente en espacio papel la escala de los tipos de línea utilizados para crear las líneas en espacio modelo, mediante la variable de sistema PSLTSCALE. Observe que el tipo de línea muestra la misma escala en las dos ventanas gráficas, incluso en aquellos objetos que tienen distintos factores de ampliación.



psltscale=1, trazos ajustados al espacio papel

psltscale=0, trazos ajustados al espacio en que se crearon

El método SetVariable permite ajustar el valor de la variable de sistema PSLTSCALE.

Uso de ventanas gráficas sombreadas

Si el dibujo contiene caras 3D, mallas, objetos de extrusión, superficies o sólidos, se puede trazar desde el espacio papel mediante las opciones Como se muestra, Estructura alámbrica, Oculto y Modelizado. Es posible obtener una vista preliminar de las ventanas sombreadas y modelizadas, trazarlas e imprimirlas en archivo con sombreado y modelizado completos.

Para establecer una opción para trazar ventanas gráficas sombreadas en el espacio papel, utilice la propiedad ShadePlot del objeto PViewport.

Nota Para ocultar líneas al trazar ventanas gráficas del espacio modelo (objetos Viewport), utilice la propiedad PlotHidden del objeto Layout. Esta propiedad requiere un valor booleano: TRUE para quitar las líneas ocultas, FALSE para dibujarlas.

Impresión de dibujos

Los dibujos se pueden imprimir directamente, tal como se ven en el espacio modelo, o se pueden preparar primero en una presentación de espacio papel. Suele ser preferible trazar desde el espacio modelo cuando se desea ver o comprobar el dibujo antes de crear una presentación del espacio papel. Una vez que el modelo esté listo, se puede preparar e imprimir una presentación de espacio papel.

Nota La variable de sistema BACKGROUNDPLOT debe establecerse en 0 para que un archivo de comandos pueda trazar varios trabajos.

El trazado implica trabajar con dos objetos ActiveX Automation: el objeto Layout y el objeto Plot. El objeto Layout contiene la configuración de los trazadores de la presentación. El objeto Plot contiene los métodos y propiedades que inician y supervisan una secuencia de trazado.

- **Realización de una impresión básica**
- **Impresión desde el espacio modelo**
- **Impresión desde el espacio papel**

Realización de una impresión básica

Desde el objeto Plot se pueden utilizar los siguientes métodos y propiedades:

PlotToFile

Imprime en un archivo.

PlotToDevice

Traza en una impresora o trazador.

DisplayPlotPreview

Muestra una vista preliminar del trazado especificado.

QuietErrorMode

Conmuta el modo de error silencioso para informar de errores de trazado.

Impresión desde el espacio modelo

Normalmente, cuando se traza un dibujo grande, como el plano de una planta, se puede precisar una escala para convertir las unidades reales del dibujo en pulgadas o milímetros de trazado. Sin embargo, cuando se traza desde el espacio modelo, las opciones por defecto que se utilizan si no se han especificado otras incluyen trazar a la impresora del sistema, trazar la visualización actual, ajustar escala al papel, y desfase 0,0. Para modificar los parámetros de trazado, cambie las propiedades del objeto Layout asociado con el espacio modelo.

Impresión de la extensión de una presentación del espacio modelo activo

En este ejemplo, en primer lugar se comprueba que el espacio activo es el espacio modelo. A continuación establece varios configuraciones de trazadores. Por último, el trabajo se envía para su trazado mediante el método PlotToDevice.

```
Sub Ch9_PrintModelSpace()
    ' Verify that the active space is model space
    If ThisDrawing.ActiveSpace = acPaperSpace Then
        ThisDrawing.MSpace = True
        ThisDrawing.ActiveSpace = acModelSpace
    End If
    ' Set the extents and scale of the plot area
    ThisDrawing.ModelSpace.Layout.PlotType = acExtents
    ThisDrawing.ModelSpace.Layout. _
                          StandardScale = acScaleToFit
    ' Set the number of copies to one
    ThisDrawing.Plot.NumberOfCopies = 1
    ' Initiate the plot
    ThisDrawing.Plot.PlotToDevice
End Sub
```

El nombre del dispositivo se puede especificar mediante la propiedad ConfigName. Este dispositivo se puede reemplazar en el método PlotToDevice si se especifica un archivo PC3.

¿Comentarios?

Impresión desde el espacio papel

Se puede trazar una presentación del espacio papel. Es posible trazar la presentación activa, como se explica en "Trazar desde el espacio modelo", o especificar el nombre de la presentación que se desea trazar.

<$nopage>imágenes bitonales. <$nopage>bloque (referencias): <$nopage>BlockReference (objeto):<$nopage>refx.

Técnicas avanzadas de dibujo y organización

A medida que gane experiencia, podrá aprovechar las diversas funciones avanzadas de AutoCAD para perfeccionar sus aplicaciones.

Puede incluir imágenes ráster en el dibujo, como fotografías digitales, aéreas y por satélite, así como imágenes modelizadas por ordenador. Para obtener información sobre las imágenes ráster, además de lo expuesto en esta sección, consulte el *Manual del usuario*.

Además de mejorar la imagen visual del dibujo, AutoCAD proporciona varias funciones para ayudarle a organizar datos, lo que permite ampliar aún más la inteligencia de los objetos del dibujo.

- **Trabajo con imágenes ráster**
- **Utilización de bloques y atributos**
- **Utilización de referencias externas**
- **Asignación y recuperación de datos extendidos**

<$nopage>imágenes bitonales.

Trabajo con imágenes ráster

Con AutoCAD® se pueden añadir imágenes ráster a los dibujos vectoriales de AutoCAD y, a continuación, ver y trazar el archivo resultante.

- **Enlazar y ajustar la escala de una imagen ráster**
- **Gestión de imágenes ráster**
- **Modificar imágenes y contornos de imagen**
- **Delimitación de imágenes**

Enlazar y ajustar la escala de una imagen ráster

Las imágenes pueden colocarse en un archivo de dibujo, pero no son realmente parte del archivo. La imagen se vincula al archivo de dibujo a través de un nombre de ruta o de un ID de documento de gestión de datos. Las rutas de imágenes vinculadas se pueden cambiar o eliminar en cualquier momento. Para enlazar una imagen, se crea un objeto Raster en el dibujo utilizando el método AddRaster. Este método requiere cuatro valores de entrada: el nombre del archivo de imagen que enlazar, el punto de inserción en el dibujo para colocarla, el factor de escala y el ángulo de rotación. Recuerde que el objeto Raster representa un vínculo independiente a la imagen, no la imagen en sí.

Una vez que haya enlazado una imagen, puede volver a enlazarla varias veces, creando un nuevo objeto Raster para cada enlace. Cada enlace tiene su propio contorno de recorte y su propia configuración de brillo, contraste, difuminación y transparencia. Una sola imagen puede cortarse en múltiples piezas y reorganizarse independientemente en el dibujo.

Puede definir el factor de escala de la imagen ráster al crear el objeto Raster de forma que la geometría de la imagen coincida con la escala de la geometría creada en el dibujo de AutoCAD. Cuando selecciona una imagen para enlazar, la imagen se inserta con un factor de escala de 1 unidad de medida de imagen a 1 unidad de medida de AutoCAD. Para definir el factor de escala de la imagen, necesita saber la escala de la geometría de la imagen, y necesita saber qué unidad de medida (pulgadas, pies, etc.) desea utilizar para definir 1 unidad de medida de AutoCAD. El archivo de imagen debe contener información de resolución que defina los PPP (puntos por pulgada) y el número de píxeles de la imagen.

Si una imagen tiene información de resolución, AutoCAD la combina con el factor de escala y con la unidad de medida de AutoCAD que usted proporcione para ajustar la escala de la imagen en el dibujo. Por ejemplo, supongamos que la

imagen ráster es un plano técnico digitalizado en el que la escala es de 1 pulgada igual a 50 pies, o 1:600, y que el dibujo de AutoCAD está definido para que una unidad represente una pulgada. Para definir el factor de escala de la imagen debe escribir 600 para el parámetro `ScaleFactor` del método AddRaster. AutoCAD inserta entonces la imagen a una escala que haga corresponder la geometría de la imagen con la geometría vectorial del dibujo.

Nota Si no se define ninguna información de resolución en el archivo de imagen enlazado, AutoCAD calcula la anchura de la imagen original como una unidad. Después de la inserción, la anchura de la imagen en unidades de AutoCAD es igual al factor de escala.

Enlazar una imagen ráster

Este ejemplo crea una cota radial en espacio modelo. Este ejemplo utiliza el archivo *watch.jpg*, que se encuentra en el directorio de ejemplos. Si no tiene esta imagen o si está situada en un directorio diferente, escriba una ruta y un nombre de archivo válidos para la variable `imageName`.

```
Sub Ch10_AttachingARaster()
    Dim insertionPoint(0 To 2) As Double
    Dim scalefactor As Double
    Dim rotationAngle As Double
    Dim dwgName As String
    Dim rasterObj As AcadRasterImage
    imageName = "C:/Program Files/AutoCAD Directory/sample/watch.jpg
insertionPoint(0) = 5
insertionPoint(1) = 5
insertionPoint(2) = 0
scalefactor = 2
rotationAngle = 0
  On Error GoTo ERRORHANDLER
   ' Attach the raster image in model space
  Set xlineObj = ThisDrawing.ModelSpace.AddXLine _
        (imageName, insertionPoint, _
               scalefactor, rotationAngle)
   ZoomAll
   Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

Gestión de imágenes ráster

Puede gestionar el nombre de imagen ráster, del archivo y su ruta utilizando las propiedades del objeto Raster.

- **Cambio de rutas de archivo de imagen**
- **Nombrar imágenes**

Cambio de rutas de archivo de imagen

La ruta y el nombre de archivo pueden consultarse o cambiarse utilizando la propiedad ImageFile. La ruta definida por esta propiedad es la ruta real en que AutoCAD busca la imagen.

Si AutoCAD no puede encontrar el dibujo (por ejemplo, si ha movido el archivo a un directorio diferente al que guardó con la propiedad ImageFile), elimina la información de ruta relativa o absoluta del nombre (por ejemplo, *\images\tree.tga* o *c:\my project\images\tree.tga* se convierte en *tree.tga*) y busca las rutas que ha definido usando el método SetProjectFilePath en el objeto Preferences. Si el dibujo no está situado en las rutas, vuelve a intentar con la primera ruta de búsqueda.

Puede eliminar la ruta del nombre del archivo o especificar una ruta relativa redefiniendo la propiedad ImageFile.

El cambio de ruta en la propiedad ImageFile no afecta a la configuración de ruta de búsqueda de los archivos del proyecto.

Nombrar imágenes

Los nombres de imagen no son necesariamente los mismos que los nombres de archivo. Cuando enlaza una imagen a un dibujo, AutoCAD utiliza el nombre de archivo sin la extensión de archivo como nombre de imagen. Puede cambiar el nombre de la imagen sin que esto afecte al nombre del archivo.

El archivo de imagen se representa mediante la propiedad ImageFile en el objeto Raster. El cambio de la propiedad ImageFile cambiará la imagen del dibujo. El nombre de la imagen es representado por la propiedad Name, y el cambio de la propiedad Name cambiará solamente el nombre de la imagen, no el archivo asociado a ella.

<$nopage>imágenes bitonales.

Modificar imágenes y contornos de imagen

Todas las imágenes tienen un contorno de imagen. Cuando enlaza una imagen a un dibujo, el contorno de imagen hereda la configuración de propiedades actual, incluyendo el color, la capa, el tipo de línea y la escala de tipo de línea. Si la imagen es una imagen bitonal, el color de la imagen y del contorno son iguales.

Al igual que con otros objetos de AutoCAD, puede modificar las imágenes y sus propiedades de contorno. Por ejemplo, puede:

- Mostrar u ocultar el contorno de imagen
- Modificar la capa, color de contorno y tipo de línea de la imagen
- Cambiar la ubicación de la imagen
- Ajustar la escala de la imagen, girarla y cambiar su altura y anchura.
- Hacer la imagen visible o invisible
- Cambiar la transparencia de la imagen
- Cambiar el brillo de la imagen, el contraste y la difuminación
- Modificar la calidad y la velocidad de visualización de la imagen
- **Visualización y ocultación de los contornos de la imagen**
- **Modificación de la capa, el color del contorno y el tipo de línea del contorno de la imagen**
- **Modificación de la escala, rotación, posición, anchura y altura de la imagen**
- **Cambio de la visibilidad de la imagen**
- **Modificación del color y la transparencia de imágenes bitonales**

- **Ajuste del brillo, el contraste y el difuminado de la imagen**

¿Comentarios?

Visualización y ocultación de los contornos de la imagen

Con ello se garantiza que la imagen no se mueva o modifique por error, y se evita el trazado o visualización del contorno. Cuando los contornos de la imagen están ocultos, las imágenes delimitadas siguen visualizándose hasta los límites especificados por el contorno y sólo este último se ve afectado. La visualización y ocultación de los contornos de la imagen afecta a todas las imágenes enlazadas al dibujo.

Para mostrar u ocultar los contornos de una imagen, utilice la propiedad ClippingEnabled.

Nota Esta propiedad solo afecta al contorno de la imagen. Para ver un cambio en la imagen al activar o desactivar esta propiedad, observe detenidamente el pequeño contorno que rodea la imagen.

Modificación de la capa, el color del contorno y el tipo de línea del contorno de la imagen

El color y el tipo de línea de los contornos de una imagen, así como la capa de donde esta se encuentra, pueden cambiarse con las siguientes propiedades:

Layer

Especifica la capa de la imagen.

Color

Especifica el color del contorno de la imagen.

Linetype

Especifica el tipo de línea de la imagen.

Modificación de la escala, rotación, posición, anchura y altura de la imagen

La escala, rotación, posición, anchura y altura de una imagen se pueden cambiar con los siguientes métodos y propiedades:

ScaleEntity

Asigna una escala a la imagen

Rotate

Rota la imagen.

Origin

Especifica la posición de la imagen.

Width

Especifica la anchura de la imagen en píxeles.

Height

Especifica la altura de la imagen en píxeles.

ImageWidth

Especifica la anchura de la imagen en unidades de base de datos.

ImageHeight

Determina la altura de la imagen en unidades de base de datos

ShowRotation

Determina si la imagen ráster se muestra girada

¿Comentarios?

Cambio de la visibilidad de la imagen

La visibilidad de las imágenes afecta a la velocidad de redibujo al ocultar imágenes en la sesión de dibujo actual. Las imágenes ocultas no se muestran ni se trazan, sólo se muestra el contorno del dibujo. Para ocultar imágenes, asigne a la propiedad ImageVisibility el valor FALSE. Para volver a visualizar las imágenes, asigne a la propiedad ImageVisibility el valor TRUE.

<$nopage>imágenes bitonales.

Modificación del color y la transparencia de imágenes bitonales

Las imágenes ráster bitonales son imágenes que constan únicamente de un color de primer plano y un color de fondo. Al enlazar una imagen bitonal, los píxeles de primer plano de la imagen heredan los parámetros de capa actuales del color. Además de las modificaciones que pueden efectuarse en cualquier imagen enlazada, las imágenes bitonales pueden modificarse mediante el cambio del color de primer plano y la activación o desactivación de la transparencia del fondo.

Nota Las imágenes bitonales y los contornos de imágenes bitonales siempre tienen el mismo color.

Para cambiar el color de fondo de una imagen bitonal, utilice la propiedad Color. Para activar o desactivar la transparencia, utilice la propiedad Transparency.

Ajuste del brillo, el contraste y el difuminado de la imagen

Es posible ajustar el brillo, el contraste y el difuminado de las imágenes de AutoCAD para su visualización en pantalla y para su impresión, sin alterar el archivo de la imagen ráster original.

Utilice las siguientes propiedades para ajustar el brillo, el contraste y el difuminado:

Brightness

Especifica el nivel de brillo de una imagen.

Contrast

Especifica el grado de contraste de una imagen.

Fade

Especifica la intensidad de difuminado de una imagen.

Delimitación de imágenes

La delimitación de las imágenes permite definir la zona de la imagen que se desee ver e imprimir. El contorno delimitador debe ser un polígono 2D o un rectángulo con los vértices dentro del contorno de la imagen. Varias copias de la misma imagen pueden tener distintos contornos

Para delimitar una imagen:

1. Active los contornos de la imagen con la propiedad ClippingEnabled
2. Especifique el contorno delimitador y realice la delimitación con el método ClipBoundary. Este método requiere un valor de entrada: una matriz variante de un sistema de coordenadas 2D que especifique el contorno delimitador de una imagen ráster.

- **Modificación del contorno delimitador**
- **Visualización y ocultación del contorno delimitador**

Modificación del contorno delimitador

Para modificar un contorno delimitador existente, sencillamente repita los pasos anteriores. Se borrará el contorno antiguo y aparecerá el nuevo en su lugar.

Visualización y ocultación del contorno delimitador

Las imágenes delimitadas pueden mostrarse con el contorno delimitador, o bien, puede ocultarse este contorno y mostrar los contornos originales de la imagen. Para ocultar un contorno delimitador y mostrar la imagen original, asigne a la propiedad ClippingEnabled el valor FALSE. Para visualizar la imagen delimitada, establezca la propiedad ClippingEnabled en TRUE.

- **Delimitación del contorno delimitador de una imagen ráster**

Delimitación del contorno delimitador de una imagen ráster

Este ejemplo crea una cota radial en espacio modelo. Después delimita la imagen basándose en un contorno delimitador. Este ejemplo utiliza el archivo *downtown.jpg*, que se encuentra en el directorio de ejemplos. Si no tiene esta imagen o si está situada en un directorio diferente, escriba una ruta y un nombre de archivo válidos para la variable imageName.

```
Sub Ch10_ClippingRasterBoundary()
    Dim insertionPoint(0 To 2) As Double
    Dim scalefactor As Double
    Dim rotationAngle As Double
    Dim dwgName As String
    Dim rasterObj As AcadRasterImage
    imageName = "C:\AutoCAD\sample\downtown.jpg"
    insertionPoint(0) = 5
    insertionPoint(1) = 5
    insertionPoint(2) = 0
    scalefactor = 2
    rotationAngle = 0
    On Error GoTo ERRORHANDLER
    ' Creates a raster image in model space
    Set xlineObj = ThisDrawing.ModelSpace.AddXLine _
                    (imageName, insertionPoint, _
                     scalefactor, rotationAngle)
   ZoomAll
    ' Establish the clip boundary with an array of points
    Dim clipPoints(0 To 9) As Double
    clipPoints(0) = 6: clipPoints(1) = 6.75
    clipPoints(2) = 7: clipPoints(3) = 6
    clipPoints(4) = 6: clipPoints(5) = 5
    clipPoints(6) = 5: clipPoints(7) = 6
    clipPoints(8) = 6: clipPoints(9) = 6,75
    ' Clip the image
```

```
    rasterObj.ClipBoundary clipPoints
    ' Enable the display of the clip
    rasterObj.ClippingEnabled = True
    ThisDrawing.Regen acActiveViewport
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

<$nopage>bloque (referencias):<$nopage>BlockReference (objeto):

Utilización de bloques y atributos

AutoCAD dispone de varias características que ayudan a gestionar los objetos del dibujo. Los bloques facilitan la organización y manipulación de varios objetos como un solo componente. Los atributos asocian elementos de información con los bloques del dibujo; por ejemplo, números de referencia de piezas y precios.

Con las referencias externas de AutoCAD, o refX, se pueden superponer o enlazar dibujos enteros con el dibujo actual. Al abrir el dibujo actual, cualquier modificación efectuada en el dibujo al que se hace referencia quedará reflejada en el dibujo actual.

- **Utilización de bloques**
- **Utilización de atributos**

<$nopage>bloque (referencias):<$nopage>BlockReference (objeto):

Utilización de bloques

Un bloque es una colección de objetos que pueden agruparse para formar un único objeto o una referencia de bloque. Las referencias de bloque de los dibujos pueden insertarse, ajustar su escala y girarse. También puede descomponer la referencia a bloque en sus objetos, modificarlos y definir de nuevo el bloque. AutoCAD actualiza todas las futuras y actuales copias de la referencia de bloque basándose en la definición del bloque.

Los bloques se pueden definir a partir de objetos dibujados originalmente en diferentes capas con distintos colores y tipos de línea. Se puede conservar la información sobre la capa, el color y el tipo de línea de un bloque. Esto permite que cada vez que se inserte un bloque, los objetos del mismo se dibujen en su capa con el color y tipo de línea originales.

Para obtener más información acerca del uso de bloques, véase “Creación e inserción de símbolos (bloques)” en el *Manual del usuario*.

- **Definición de bloques**
- **Inserción de bloques**
- **Descomposición de referencias de bloque**
- **Redefinición de bloques**

Definición de bloques

Para crear un bloque nuevo, utilice el método Add. Este método requiere dos valores de entrada: la ubicación del dibujo en donde se añade el bloque y el nombre del bloque que se crea.

Una vez creado, se le pueden añadir otros objetos geométricos u otro bloque. Después, puede insertar una instancia del bloque en el dibujo. Al bloque que se inserta se le conoce como objeto de referencia de bloque.

También puede crear un bloque utilizando el método WBlock para agrupar objetos en un archivo de dibujo independiente. El archivo de dibujo puede entonces utilizarse como definición de bloque para otros dibujos. AutoCAD considera como bloque cualquier dibujo que se inserte en otro dibujo.

Para obtener más información acerca de la definición de bloques, véase “Creación de bloques” en el *Manual del usuario*.

<$nopage>bloque (referencias):

Inserción de bloques

Se puede utilizar el método InsertBlock para insertar bloques o dibujos completos en el dibujo actual. El método InsertBlock requiere seis valores como entrada: el punto de inserción, el nombre del bloque o dibuje que se inserta, el factor de escala X, el factor de escala Y, el factor de escala Z, y el ángulo de rotación.

Los dibujos insertados en otros dibujos son considerados por AutoCAD como referencias a bloques. Las siguientes inserciones refieren a la definición de bloque (que contiene la descripción geométrica del bloque) con parámetros de posición, escala y rotación distintos. Si se modifica el dibujo original una vez insertado, las modificaciones no tendrán efecto en el bloque insertado. Si quiere que el bloque insertado refleje las modificaciones realizadas en el dibujo original, puede definir de nuevo el bloque mediante la reinserción del dibujo original. Para ello puede utilizar el método InsertBlock.

Si inserta un dibujo a modo de bloque, el bloque adquiere automáticamente el nombre del archivo. Después de crear el bloque, puede cambiar su nombre mediante la propiedad Name.

Por defecto, el punto base para la inserción de dibujos en AutoCAD tiene las coordenadas (0, 0, 0). Se puede modificar el punto base de un dibujo, si abre el dibujo original especifica un punto base de inserción diferente para la variable de sistema INSBASE con el método SetVariable. AutoCAD empleará el nuevo punto base la próxima vez que inserte el dibujo.

Si el dibujo insertado contiene objetos PaperSpace, dichos objetos no se incluyen en la definición de bloque del dibujo actual. Para usar los objetos dibujados en espacio papel en otros dibujos, abra el dibujo original y defina los objetos del espacio papel como un bloque con el método Add. La inserción de un dibujo en otro se puede realizar tanto en espacio papel como en espacio modelo.

Una referencia a bloque no puede iterarse para encontrar los objetos originales que la componen. Sin embargo, se puede iterar la definición de bloque original, o se puede descomponer la referencia a bloque en sus componentes originales.

También se puede insertar una matriz de bloques con el método AddMInsertBlock. Este método no inserta un bloque en el dibujo, como hace InsertBlock, sino que inserta una matriz del bloque especificado. Este método devuelve un objeto MInsertBlock.

Para obtener más información acerca de la inserción de bloques, véase “Inserción de bloques” en el *Manual del usuario*.

Definición e inserción de un bloque en un dibujo

Este ejemplo define un bloque y añade un círculo a su definición. Después se inserta el bloque en un dibujo, como referencia de bloque.

```
Sub Ch10_InsertingABlock()
    ' Define the block
    Dim blockObj As AcadBlock
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
    insertionPnt(2) = 0
    Set blockObj = ThisDrawing.Blocks.Add _
                     (insertionPnt, "CircleBlock")
    ' Add a circle to the block
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 0
    center(1) = 0
    center(2) = 0
    radius = 1
    Set circleObj = blockObj.AddCircle(center, radius)
    ' Insert the block
    Dim blockRefObj As AcadBlockReference
    insertionPnt(0) = 2
    insertionPnt(1) = 2
    insertionPnt(2) = 0
    Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
                (insertionPnt, "CircleBlock", 1#, 1#, 1#, 0)
    ZoomAll
    MsgBox "The circle belongs to " & blockRefObj.ObjectName
End Sub
```

Nota Una vez insertado, el SCU del archivo externo se alinea en paralelo con el plano XY del sistema de coordenadas personales (SCP) del dibujo actual. De esta forma, un bloque procedente de un archivo externo puede insertarse con cualquier orientación en el espacio previa definición de las coordenadas SCP.

<$nopage>BlockReference (objeto):

Descomposición de referencias de bloque

Utilice el método Explode para romper una referencia de bloque. Con ello podrá modificar el bloque, o añadir o borrar los objetos que lo definen.

Presentación de los resultados de la descomposición de una referencia de bloque

Este ejemplo crea un bloque y añade un círculo a su definición. Después se inserta el bloque en un dibujo, como referencia de bloque. A continuación se descompone la referencia de bloque y se muestran los objetos resultantes de este proceso junto con su tipo de objeto.

```
Sub Ch10_ExplodingABlock()
    ' Define the block
    Dim blockObj As AcadBlock
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
    insertionPnt(2) = 0
    Set blockObj = ThisDrawing.Blocks.Add _
                    (insertionPnt, "CircleBlock")
    ' Add a circle to the block
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 0
    center(1) = 0
    center(2) = 0
    radius = 1
    Set circleObj = blockObj.AddCircle(center, radius)
    ' Insert the block
    Dim blockRefObj As AcadBlockReference
    insertionPnt(0) = 2
    insertionPnt(1) = 2
```

```
    insertionPnt(2) = 0
    Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
              (insertionPnt, "CircleBlock", 1#, 1#, 1#, 0)
    ZoomAll
    MsgBox "The circle belongs to " & blockRefObj.ObjectName
    ' Explode the block reference
    Dim explodedObjects As Variant
    explodedObjects = blockRefObj.Explode
    ' Loop through the exploded objects
    Dim I As Integer
    For I = 0 To UBound(explodedObjects)
        explodedObjects(I).Color = acRed
        explodedObjects(I).Update
        MsgBox "Exploded Object " & I & ": " _
                             & explodedObjects(I).ObjectName
        explodedObjects(I).Color = acByLayer
        explodedObjects(I).Update
    Next
End Sub
```

¿Comentarios?

Redefinición de bloques

Para redefinir un bloque se puede utilizar cualquiera de los métodos y propiedades del objeto Block. Al redefinir un bloque, todas las referencias a ese bloque del dibujo se actualizarán inmediatamente para reflejar la nueva definición.

La redefinición afecta a las inserciones de bloque ya efectuadas y a las futuras. Los atributos constantes se pierden y son reemplazados por atributos constantes nuevos. Los atributos variables permanecen intactos, incluso si el nuevo bloque no tiene atributos.

Redefinición de objetos en una definición de bloque

Este ejemplo crea un bloque y añade un círculo a su definición. Después se inserta el bloque en un dibujo, como referencia de bloque. El círculo de la definición de bloque se actualiza y la referencia de bloque se actualiza automáticamente.

```
Sub Ch10_RedefiningABlock()
    ' Define the block
    Dim blockObj As AcadBlock
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
    insertionPnt(2) = 0
    Set blockObj = ThisDrawing.Blocks.Add _
                    (insertionPnt, "CircleBlock")
    ' Add a circle to the block
    Dim circleObj As AcadCircle
    Dim center(0 To 2) As Double
    Dim radius As Double
    center(0) = 0
    center(1) = 0
    center(2) = 0
```

```
    radius = 1
    Set circleObj = blockObj.AddCircle(center, radius)
    ' Insert the block
    Dim blockRefObj As AcadBlockReference
    insertionPnt(0) = 2
    insertionPnt(1) = 2
    insertionPnt(2) = 0
    Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
               (insertionPnt, "CircleBlock", 1#, 1#, 1#, 0)
    ZoomAll
    ' Redefine the circle in the block,
    ' and update the block reference
    circleObj.radius = 3
    blockRefObj.Update
End Sub
```

¿Comentarios?

Utilización de atributos

Las referencias de atributos sirven de etiqueta o identificador interactivo para enlazar texto a un bloque. Ejemplos de atributos pueden ser números de referencia de piezas, precios, comentarios o nombres de propietarios.

Es posible extraer de un dibujo información de referencia de atributos y emplearla en una hoja de cálculo o base de datos con la finalidad de generar listas de piezas o listas de elementos (BOM). Un bloque puede tener asociada más de una referencia de atributo, siempre que éstas tengan identificadores distintos. También puede definir atributos constantes. Dado que tienen el mismo valor en todas las apariciones del bloque, AutoCAD no solicita un valor cuando se inserta el bloque.

Los atributos pueden ser invisibles, lo que significa que las referencias de atributo no se muestran ni se imprimen. No obstante, la información de las referencias de atributos se almacena en el archivo del dibujo.

Para obtener más información acerca del uso de atributos, véase “Información general de atributos de bloque“ en el *Manual del usuario*.

- **Creación de definiciones y referencias de atributos**
- **Edición de definiciones de atributos**
- **Extracción de la información de atributos**

Creación de definiciones y referencias de atributos

Para crear una referencia de atributo, es preciso crear antes una definición de atributo en un bloque mediante el método AddAttribute. Este método requiere seis valores de entrada: la altura del texto de atributo, el modo de atributo, la cadena de mensaje, el punto de inserción, la cadena de identificación y el valor por defecto del atributo.

El valor del modo es opcional. Se pueden introducir cinco constantes para especificar el modo de atributo:

acAttributeModeNormal

Especifica que se mantiene el modo actual de cada atributo.

acAttributeModeInvisible

Especifica que los valores de atributos no aparezcan cuando se inserte el bloque. El comando ATTDISP anula el modo invisible.

acAttributeModeConstant

Asigna un valor fijo a los atributos para las inserciones de bloque.

acAttributeModeVerify

Solicita que se verifique el valor del atributo al insertar el bloque.

acAttributeModePreset

Establece el valor por defecto del atributo al insertar un bloque que contiene un atributo introducido. El valor no puede modificarse en este modo.

Puede introducir cualquier combinación de opciones, todas ellas o ninguna. Si desea especificar una combinación de opciones, añada las constantes juntas. Por ejemplo, podrá introducir acAttributeModeInvisible + acAttributeModeConstant.

La cadena de mensaje aparece al insertar un bloque que contiene el atributo. El valor por defecto de esta cadena es la cadena Tag. Introduzca el modo acAttributeModeConstant para desactivar el mensaje.

El identificador reconoce cada aparición del atributo. Puede utilizar cualquier carácter salvo espacios o signos de admiración. AutoCAD convierte las letras minúsculas en mayúsculas.

Una vez que la definición de atributo se ha incorporado al bloque, siempre que inserte el bloque con el método InsertBlock podrá especificar un valor distinto para la referencia de atributo.

La definición de atributo está asociada al bloque sobre el que se crea. Las definiciones de atributo que se crean en espacio modelo o espacio papel no se consideran enlazadas a ningún bloque determinado.

Creación de una definición de atributo

Este ejemplo crea un bloque y le añade un atributo. Después se inserta el bloque en el dibujo.

```
Sub Ch10_CreatingAnAttribute()
    ' Define the block
    Dim blockObj As AcadBlock
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
    insertionPnt(2) = 0
    Set blockObj = ThisDrawing.Blocks.Add _
                     (insertionPnt, "BlockWithAttribute")
    ' Add an attribute to the block
    Dim attributeObj As AcadAttribute
    Dim height As Double
    Dim mode As Long
    Dim prompt As String
    Dim insertionPoint(0 To 2) As Double
    Dim tag As String
    Dim value As String
    height = 1
    mode = acAttributeModeVerify
    prompt = "New Prompt"
    insertionPoint(0) = 5
    insertionPoint(1) = 5
    insertionPoint(2) = 0
    tag = "New Tag"
```

```
    value = "New Value"
    Set attributeObj = blockObj.AddAttribute(height, mode, _
                          prompt, insertionPoint, tag, value)
    ' Insert the block, creating a block reference
    ' and an attribute reference
    Dim blockRefObj As AcadBlockReference
    insertionPnt(0) = 2
    insertionPnt(1) = 2
    insertionPnt(2) = 0
    Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
                (insertionPnt, "BlockWithAttribute", 1#, 1#, 1#, 0)
End Sub
```

¿Comentarios?

Edición de definiciones de atributos

Puede utilizar las propiedades y métodos del objeto Attribute para editar el atributo. Algunas de las propiedades de un atributo incluyen:

Alignment

Determina la alineación horizontal y vertical del atributo.

Backward

Especifica la dirección del texto del atributo.

FieldLength

Especifica la longitud de campo del atributo.

Height

Especifica la altura del atributo

InsertionPoint

Especifica el punto de inserción del atributo.

Mode

Especifica el modo del atributo.

PromptString

Especifica la cadena de mensaje del atributo.

Rotation

Especifica la rotación del atributo

ScaleFactor

Especifica el factor de escala del atributo

TagString

Especifica la cadena del identificador del atributo.

A continuación se incluyen algunos métodos que pueden utilizarse para modificar atributos:

ArrayPolar

Crea una matriz polar

ArrayRectangular

Crea una matriz rectangular

Copy

Copia el atributo

Erase

Borra el atributo

Mirror

Refleja el atributo.

Move

Desplaza el atributo

Rotate

Rota el atributo

ScaleEntity

Asigna una escala al atributo

Para volver a crear una definición de atributo

Este ejemplo crea un bloque y le añade un atributo. Después se inserta el bloque en el dibujo. Por último, se actualiza el atributo de texto para que se muestre reflejado a la izquierda.

```
Sub Ch10_RedefiningAnAttribute()
    ' Define the block
    Dim blockObj As AcadBlock
```

```
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
    insertionPnt(2) = 0
    Set blockObj = ThisDrawing.Blocks.Add _
                    (insertionPnt, "BlockWithAttribute")
    ' Add an attribute to the block
    Dim attributeObj As AcadAttribute
    Dim height As Double
    Dim mode As Long
    Dim prompt As String
    Dim insertionPoint(0 To 2) As Double
    Dim tag As String
    Dim value As String
    height = 1
    mode = acAttributeModeVerify
    prompt = "New Prompt"
    insertionPoint(0) = 5
    insertionPoint(1) = 5
    insertionPoint(2) = 0
    tag = "New Tag"
    value = "New Value"
    Set attributeObj = blockObj.AddAttribute(height, mode, _
                          prompt, insertionPoint, tag, value)
    ' Insert the block, creating a block reference
    ' and an attribute reference
    Dim blockRefObj As AcadBlockReference
    insertionPnt(0) = 2
    insertionPnt(1) = 2
    insertionPnt(2) = 0
    Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
               (insertionPnt, "BlockWithAttribute", 1#, 1#, 1#, 0)
    ' Redefine the attribute text to display backwards.
    attributeObj.Backward = True
    attributeObj.Update
End Sub
```

¿Comentarios?

Extracción de la información de atributos

Puede extraer información de los atributos de un dibujo mediante los métodos GetAttributes y GetConstantAttributes. GetAttributes devuelve una matriz de todas las referencias a atributos enlazadas a un bloque, junto con sus valores actuales. GetConstantAttributes devuelve una matriz de los atributos constantes enlazados con el bloque o la referencia externa. Los atributos que devuelve este método son definiciones de atributos constantes, no de referencias a atributos.

Para extraer información de atributos no es preciso utilizar archivos de plantilla ni crear archivos de información de atributos. Puede examinar la información de los atributos sencillamente iterando en la matriz de referencias de atributo con las propiedades TagString y TextString de la referencia.

La propiedad TagString representa el identificador individual de la referencia de atributo. La propiedad TextString contiene el valor de la referencia de atributo.

Para obtener más información acerca de la extracción de información de atributos, véase “Extracción de datos de atributos de bloque“ en el *Manual del usuario*.

Obtención de información de referencia de atributos

Este ejemplo crea un bloque y le añade un atributo. Después se inserta el bloque en el dibujo. Se obtienen los datos de atributos y se muestran en un cuadro de mensaje. Después se actualizan para la referencia de bloque y de nuevo se obtienen y se muestran en un cuadro de mensaje.

```
Sub Ch10_GettingAttributes()
    ' Create the block
    Dim blockObj As AcadBlock
    Dim insertionPnt(0 To 2) As Double
    insertionPnt(0) = 0
    insertionPnt(1) = 0
```

```
insertionPnt(2) = 0
Set blockObj = ThisDrawing.Blocks.Add _
                     (insertionPnt, "TESTBLOCK")
' Define the attribute definition
Dim attributeObj As AcadAttribute
Dim height As Double
Dim mode As Long
Dim prompt As String
Dim insertionPoint(0 To 2) As Double
Dim tag As String
Dim value As String
height = 1#
mode = acAttributeModeVerify
prompt = "Attribute Prompt"
insertionPoint(0) = 5
insertionPoint(1) = 5
insertionPoint(2) = 0
tag = "Attribute Tag"
value = "Attribute Value"
' Create the attribute definition object on the block
Set attributeObj = blockObj.AddAttribute _
                   (height, mode, prompt, _
                   insertionPoint, tag, value)
' Insert the block
Dim blockRefObj As AcadBlockReference
insertionPnt(0) = 2
insertionPnt(1) = 2
insertionPnt(2) = 0
Set blockRefObj = ThisDrawing.ModelSpace.InsertBlock _
                  (insertionPnt, "TESTBLOCK", 1, 1, 1, 0)
ZoomAll
' Get the attributes for the block reference
Dim varAttributes As Variant
varAttributes = blockRefObj.GetAttributes
' Move the attribute tags and values into a
' string to be displayed in a Msgbox
Dim strAttributes As String
strAttributes = ""
Dim I As Integer
For I = LBound(varAttributes) To UBound(varAttributes)
    strAttributes = strAttributes + " Tag: " + _
                 varAttributes(I).TagString + vbCrLf + _
                 " Value: " + varAttributes(I).textString
Next
MsgBox "The attributes for blockReference " + _
               blockRefObj.Name & " are: " & vbCrLf _
               & strAttributes
' Change the value of the attribute
' Note: There is no SetAttributes. Once you have the
' variant array, you have the objects.
```

```
    ' Changing them changes the objects in the drawing.
    varAttributes(0).textString = "NEW VALUE!"
    ' Get the attributes again
    Dim newvarAttributes As Variant
    newvarAttributes = blockRefObj.GetAttributes
    ' Again, display the tags and values
    strAttributes = ""
    For I = LBound(varAttributes) To UBound(varAttributes)
        strAttributes = strAttributes + " Tag: " + _
                    newvarAttributes(I).TagString + vbCrLf + _
                    " Value: " + newvarAttributes(I).textString
    Next
    MsgBox "The attributes for blockReference " & _
                  blockRefObj.Name & " are: " & vbCrLf _
                  & strAttributes
End Sub
```

¿Comentarios?

<$nopage>refx.

Utilización de referencias externas

Las referencias externas (refX) vinculan otro dibujo con el actual. Cuando se inserta un dibujo como bloque, el bloque se guarda con toda la geometría asociada en la base de datos del dibujo actual. No se actualiza si se modifica el dibujo original. Sin embargo, al insertar un dibujo como una referencia externa, éste se actualiza al modificar el dibujo original. Por tanto, los dibujos que contienen referencias externas reflejan siempre los cambios más recientes efectuados en los archivos referidos desde el exterior.

Al igual que una referencia a bloque, una referencia cruzada se muestra en el dibujo actual como un único objeto. Sin embargo, apenas aumenta el tamaño del archivo del dibujo actual y no puede descomponerse. Como en los bloques, las referencias externas asociadas al dibujo pueden anidarse.

Para obtener más información acerca de las referencias externas, véase “Enlace, actualización y unión de referencias externas“ en el *Manual del usuario*.

- **Actualización de referencias externas**
- **Enlace de referencias externas**
- **Desenlace de referencias externas**
- **Recarga de referencias externas**
- **Descarga de referencias externas**
- **Unión de referencias externas**
- **Delimitación de bloques y referencias externas**
- **Solicitud de carga y rendimiento de las referencias externas**

Actualización de referencias externas

Cuando se abre o imprime un dibujo, AutoCAD carga las referencias externas de forma que el dibujo referido se refleje en el estado más reciente. Una vez efectuadas las modificaciones en el dibujo referido externamente y guardado el archivo, otros usuarios podrán acceder inmediatamente a las modificaciones con tan sólo volver a cargar la referencia externa.

Enlace de referencias externas

Al enlazar una referencia externa, se asocia un dibujo (el archivo de referencia o la referencia externa) con el dibujo actual. Cuando un dibujo hace alusión a una referencia externa, AutoCAD sólo enlaza la definición de la RefX al dibujo, a diferencia de los bloques normales, en los que se guarda la definición y el contenido del bloque con el dibujo actual. AutoCAD lee el dibujo de referencia para determinar lo que debe mostrarse en el dibujo actual. Si el archivo de referencia no existe o está dañado, sus datos no se muestran en el dibujo actual. Cada vez que se abre un dibujo, AutoCAD carga todos los objetos gráficos y no gráficos (como capas, tipos de línea y estilos de texto) desde los archivos de referencia. Si VISRETAIN está activada, AutoCAD guarda toda la información de capa dependiente de la referencia externa que se haya actualizado en el dibujo actual.

Las referencias externas pueden enlazarse tantas veces como se desee y cada una en una posición, con una escala y un ángulo de rotación diferentes. También pueden controlarse las capas dependientes y las propiedades de tipo de línea definidas en la referencia externa.

Para enlazar una referencia externa, utilice el método AttachExternalReference. Este método requiere la ruta y el nombre de archivo del dibujo al que se va a hacer referencia, el nombre de la referencia externa que se va a utilizar en el dibujo actual, el punto de inserción, la escala y la información de rotación de la referencia externa. El método AttachExternalReference devuelve el objeto ExternalReference recién creado.

Para obtener más información sobre el enlace de referencias externas, véase “Enlace de referencias externas” en el *Manual del usuario*.

Enlace de una referencia externa a un dibujo

Este ejemplo muestra todos los bloques del dibujo actual antes y después de añadir una referencia externa. Se utiliza el archivo *3D House.dwg* del directorio de ejemplos. Si no tiene esta imagen o si está situada en un directorio diferente, escriba una ruta y un nombre de archivo válidos para la variable PathName.

```
Sub Ch10_AttachingExternalReference()
    On Error GoTo ERRORHANDLER
    Dim InsertPoint(0 To 2) As Double
    Dim insertedBlock As AcadExternalReference
    Dim tempBlock As AcadBlock
    Dim msg As String, PathName As String
    ' Define external reference to be inserted
    InsertPoint(0) = 1
    InsertPoint(1) = 1
    InsertPoint(2) = 0
    PathName = "C:/Program Files/AutoCAD 2008/sample/3D House.dwg"
    ' Display current Block information for this drawing
    GoSub ListBlocks
    ' Add the external reference to the drawing
    Set insertedBlock = ThisDrawing.ModelSpace. _
            AttachExternalReference(PathName, "XREF_IMAGE", _
            InsertPoint, 1, 1, 1, 0, False)
    ZoomAll
    ' Display new Block information for this drawing
    GoSub ListBlocks
    Exit Sub
ListBlocks:
    msg = vbCrLf ' Reset message
    For Each tempBlock In ThisDrawing.Blocks
        msg = msg & tempBlock.Name & vbCrLf
    Next
    MsgBox "The current blocks in this drawing are: " & msg
    Return
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

- **Superposición de referencias externas**

¿Comentarios?

Superposición de referencias externas

Superponer es parecido a enlazar, excepto cuando se enlaza o se superpone un dibujo. Cualquier otra superposición que esté anidada se ignora y, por lo tanto, no se visualiza. En otras palabras, las superposiciones anidadas no se cargan.

Para superponer una referencia externa, asigne el valor TRUE al parámetro Overlay del método AttachExternalReference.

Desenlace de referencias externas

Puede desenlazar definiciones de referencias externas para suprimir las referencias totalmente del dibujo. También puede borrar las referencias externas por separado. Al desenlazar una definición de referencia externa se eliminan todos los símbolos dependientes asociados a ella. Si se borran todas las copias de una referencia externa en el dibujo, AutoCAD elimina la definición de la referencia externa cuando se vuelve a abrir el dibujo.

Para desenlazar una referencia externa, utilice el método Detach. Las referencias externas anidadas no se pueden desenlazar.

Desenlace de una definición de referencia externa

Este ejemplo enlaza una referencia externa y después la desenlaza. Se utiliza el archivo *3D House.dwg* del directorio de ejemplos. Si no tiene esta imagen o si está situada en un directorio diferente, escriba una ruta y un nombre de archivo válidos para la variable PathName.

```
Sub Ch10_DetachingExternalReference()
    On Error GoTo ERRORHANDLER
    ' Define external reference to be inserted
    Dim xrefHome As AcadBlock
    Dim xrefInserted As AcadExternalReference
    Dim insertionPnt(0 To 2) As Double
    Dim PathName As String
    insertionPnt(0) = 1
    insertionPnt(1) = 1
    insertionPnt(2) = 0
    PathName = "c:/AutoCAD 2008/sample/3D House.dwg"
    ' Add the external reference
    Set xrefInserted = ThisDrawing.ModelSpace. _
            AttachExternalReference(PathName, "XREF_IMAGE", _
            insertionPnt, 1, 1, 1, 0, False)
    ZoomAll
```

```
    MsgBox "The external reference is attached."
    ' Detach the external reference definition
    Dim name As String
    name = xrefInserted.name
    ThisDrawing.Blocks.Item(name).Detach
    MsgBox "The external reference is detached."
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

¿Comentarios?

Recarga de referencias externas

Si algún usuario modifica un dibujo referido externamente mientras se trabaja en el dibujo al que está enlazada la referencia externa, puede actualizar el dibujo que contiene la referencia externa con el método Reload. Al volver a cargarlo, el dibujo de referencia externa seleccionado se actualiza en el dibujo receptor. Además, si ha descargado una referencia externa, puede cargar de nuevo el dibujo referido externamente en cualquier momento.

Recarga de una definición de referencia externa

Este ejemplo enlaza una referencia externa y después la vuelve a cargar. Se utiliza el archivo *3D House.dwg* del directorio de ejemplos. Si no tiene esta imagen o si está situada en un directorio diferente, escriba una ruta y un nombre de archivo válidos para la variable PathName.

```
Sub Ch10_ReloadingExternalReference()
    On Error GoTo ERRORHANDLER
    ' Define external reference to be inserted
    Dim xrefHome As AcadBlock
    Dim xrefInserted As AcadExternalReference
    Dim insertionPnt(0 To 2) As Double
    Dim PathName As String
    insertionPnt(0) = 1
    insertionPnt(1) = 1
    insertionPnt(2) = 0
    PathName = "c:/AutoCAD 2008/sample/3D House.dwg"
    ' Add the external reference to the block
    Set xrefInserted = ThisDrawing.ModelSpace. _
            AttachExternalReference(PathName, "XREF_IMAGE", _
            insertionPnt, 1, 1, 1, 0, False)
    ZoomAll
    MsgBox "The external reference is attached."
    ' Reload the external reference definition
    ThisDrawing.Blocks.Item(xrefInserted.name).Reload
```

```
    MsgBox "The external reference is reloaded."
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

¿Comentarios?

Descarga de referencias externas

Para descargar una referencia externa, utilice el método Unload. Cuando se descarga un archivo de referencia que no utiliza el dibujo actual, el rendimiento de AutoCAD aumenta, ya que no tiene que leer ni mostrar geometría innecesaria ni información de tablas de símbolos. La geometría de las referencias externas y de las anidadas, si las hay, no se muestra en el dibujo activo mientras no se vuelvan a cargar.

Descarga de una definición de referencia externa

Este ejemplo enlaza una referencia externa y después la descarga. Se utiliza el archivo 3D House.dwg del directorio de ejemplos. Si no tiene esta imagen o si está situada en un directorio diferente, escriba una ruta y un nombre de archivo válidos para la variable PathName.

```
Sub Ch10_UnloadingExternalReference()
    On Error GoTo ERRORHANDLER
    ' Define external reference to be inserted
    Dim xrefHome As AcadBlock
    Dim xrefInserted As AcadExternalReference
    Dim insertionPnt(0 To 2) As Double
    Dim PathName As String
    insertionPnt(0) = 1
    insertionPnt(1) = 1
    insertionPnt(2) = 0
    PathName = "c:/AutoCAD 2008/sample/3D House.dwg"
    ' Add the external reference
    Set xrefInserted = ThisDrawing.ModelSpace. _
            AttachExternalReference(PathName, "XREF_IMAGE", _
            insertionPnt, 1, 1, 1, 0, False)
    ZoomAll
    MsgBox "The external reference is attached."
    ' Unload the external reference definition
    ThisDrawing.Blocks.Item(xrefInserted.name).Unload
```

```
    MsgBox "The external reference is unloaded."
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

¿Comentarios?

Unión de referencias externas

La unión de una referencia externa con un dibujo mediante el método Bind hace que la referencia forme parte permanente del dibujo y deje de considerarse un archivo de referencia externa. La información referida externamente se convierte en un bloque. Al actualizar el dibujo referido externamente, no se actualiza la referencia externa unida. Este proceso une de forma completa la base de datos del dibujo, incluidos todos sus símbolos dependientes.

Los símbolos dependientes son objetos guardados, como por ejemplo bloques, estilos de cotas, capas, tipos de línea y estilos de texto. Al unir la referencia externa se pueden utilizar los objetos con nombre en el dibujo actual.

El método Bind sólo requiere la entrada de un parámetro: bPrefixName. Si se asigna a bPrefixName el valor TRUE, los nombres de símbolo del dibujo de la referencia externa aparecen en el dibujo actual con el prefijo <nombredebloque>x, donde x es un número entero que aumenta automáticamente para evitar que se reemplacen las definiciones de bloque existentes. Si se asigna al parámetro bPrefixName el valor FALSE, los nombres de símbolo del dibujo de la referencia externa se fusionan con el dibujo actual sin el prefijo. Si hay nombres duplicados, AutoCAD utiliza los símbolos definidos con anterioridad en el dibujo. Si no está seguro de si el dibujo contiene nombres de símbolos duplicados, es recomendable que asigne al prefijo bPrefixName el valor TRUE.

Para obtener más información acerca de la asignación de referencias externas, véase “Archivo de dibujos que contienen referencias externas (unión)“ en el *Manual del usuario.*

Unión de una definición de referencia externa

Este ejemplo enlaza una referencia externa y después la une al dibujo. Se utiliza

el archivo *3D House.dwg* del directorio de ejemplos. Si no tiene esta imagen o si está situada en un directorio diferente, escriba una ruta y un nombre de archivo válidos para la variable PathName.

```
Sub Ch10_BindingExternalReference()
    On Error GoTo ERRORHANDLER
    ' Define external reference to be inserted
    Dim xrefHome As AcadBlock
    Dim xrefInserted As AcadExternalReference
    Dim insertionPnt(0 To 2) As Double
    Dim PathName As String
    insertionPnt(0) = 1
    insertionPnt(1) = 1
    insertionPnt(2) = 0
    PathName = "c:/AutoCAD 2008/sample/3D House.dwg"
    ' Add the external reference
    Set xrefInserted = ThisDrawing.ModelSpace. _
            AttachExternalReference(PathName, "XREF_IMAGE", _
            insertionPnt, 1, 1, 1, 0, False)
    ZoomAll
    MsgBox "The external reference is attached."
    ' Bind the external reference definition
    ThisDrawing.Blocks.Item(xrefInserted.name).Bind False
    MsgBox "The external reference is bound."
    Exit Sub
ERRORHANDLER:
    MsgBox Err.Description
End Sub
```

¿Comentarios?

Delimitación de bloques y referencias externas

ActiveX Automation no dispone de ningún método para delimitar contornos de bloques y referencias externas. Utilice el comando XCLIP de AutoCAD o envíe el comando XCLIP a AutoCAD utilizando el método SendCommand.

Solicitud de carga y rendimiento de las referencias externas

Mediante una combinación de solicitud de carga y uso de índices para guardar los dibujos, es posible aumentar el rendimiento de los dibujos con referencias externas. La solicitud de carga funciona junto con las variables de sistema XLOADCTL e INDEXCTL. Si se activa la solicitud de carga y se han guardado índices en los dibujos referidos, AutoCAD carga en memoria sólo los datos del dibujo referido que sea necesario para regenerar el dibujo actual. Es decir, los elementos de referencia se pueden leer “previa solicitud”.

Para poder utilizar al máximo las ventajas de la solicitud de carga, antes hay que guardar los dibujos referidos con índices espaciales y de capa. Las mejoras en el rendimiento que se producen con las solicitudes de carga son más evidentes cuando:

- Se recorta una referencia externa para mostrar una pequeña parte de la misma y se guarda un índice espacial en el dibujo referido externamente.
- Se inutilizan varias capas de la referencia externa y el dibujo referido externamente se guarda con un índice de capa.

Para activar la solicitud de carga, utilice el método XRefDemandLoad. Si decide activar la carga mediante solicitud con la opción acDemandLoadEnabledWithCopy, AutoCAD crea una copia temporal del archivo referido externamente y carga mediante solicitud el archivo temporal. De este modo, puede cargar por solicitud la referencia externa a la vez que permite que otros usuarios modifiquen el dibujo de referencia original. Cuando desactive la solicitud de carga, AutoCAD cargará el dibujo de referencia completo independientemente de la visibilidad de las capas o de la presencia de delimitaciones.

Para activar los índices espaciales y de capa, establezca la variable de sistema

INDEXCTL mediante el método SetVariable. Los parámetros siguientes se aplican a la variable de sistema INDEXCTL:

- 0 = no se han creado índices.
- 1 = se ha creado el índice de capas.
- 2 = se ha creado el índice espacial.
- 3 = se han creado el índice espacial y el de capas.

Por defecto, INDEXCTL vale 0 cuando se crea un dibujo nuevo de AutoCAD.

Para obtener más información acerca de la solicitud de carga y referencias externas, véase “Incremento del rendimiento con referencias externas de gran tamaño” en el *Manual del usuario*.

Asignación y recuperación de datos extendidos

Puede utilizar datos extendidos (datoseX) como un medio para enlazar información con los objetos de un dibujo.

Asignación de datos extendidos a un conjunto de selección

En este ejemplo, se pide al usuario que designe objetos del dibujo. Los objetos designados se introducen en un conjunto de selección y los datos extendidos especificados se enlazan con todos los objetos de dicho conjunto

```
Sub Ch10_AttachXDataToSelectionSetObjects()
    ' Create the selection set
    Dim sset As Object
    Set sset = ThisDrawing.SelectionSets.Add("SS1")
    ' Prompt the user to select objects
    sset.SelectOnScreen
    ' Define the xdata
    Dim appName As String, xdataStr As String
    appName = "MY_APP"
    xdataStr = "This is some xdata"
    Dim xdataType(0 To 1) As Integer
    Dim xdata(0 To 1) As Variant
    ' Define the values for each array
    '1001 indicates the appName
    xdataType(0) = 1001
    xdata(0) = appName
    '1000 indicates a string value
    xdataType(1) = 1000
    xdata(1) = xdataStr
    ' Loop through all entities in the selection
    ' set and assign the xdata to each entity
    Dim ent As Object
    For Each ent In sset
        ent.SetXData xdataType, xdata
    Next ent
End Sub
```

Visualización de los datos extendidos de todos los objetos de un conjunto de selección

Este ejemplo muestra los datos extendidos enlazados en el ejemplo anterior. Si enlaza datos extendidos que no son cadenas (tipo 1000), necesitará corregir este código

```
Sub Ch10_ViewXData()
    ' Find the selection created in previous example
    Dim sset As Object
    Set sset = ThisDrawing.SelectionSets.Item("SS1")
    ' Define the xdata variables to hold xdata information
    Dim xdataType As Variant
    Dim xdata As Variant
    Dim xd As Variant
    'Define index counter
    Dim xdi As Integer
    xdi = 0
    ' Loop through the objects in the selection set
    ' and retrieve the xdata for the object
    Dim msgstr As String
    Dim appName As String
    Dim ent As AcadEntity
    appName = "MY_APP"
    For Each ent In sset
        msgstr = ""
        xdi = 0
        ' Retrieve the appName xdata type and value
        ent.GetXData appName, xdataType, xdata
        ' If the xdataType variable is not initialized, there
        ' was no appName xdata to retrieve for that entity
        If VarType(xdataType) <> vbEmpty Then
            For Each xd In xdata
                msgstr = msgstr & vbCrLf & xdataType(xdi) _
                         & ": " & xd
                xdi = xdi + 1
            Next xd
        End If
        ' If the msgstr variable is NULL, there was no xdata
        If msgstr = "" Then msgstr = vbCrLf & "NONE"
        MsgBox appName & " xdata on " & ent.ObjectName & _
                                        ":" & vbCrLf & msgstr
    Next ent
End Sub
```

<$nopage>controles de cuadros de diálogo.<$nopage>controles de formularios.

Desarrollo de aplicaciones con VBA

Muchas tareas de programación implican algo más que trabajar con el modelo de objetos ActiveX de AutoCAD. Este capítulo proporciona una breve introducción a la creación de cuadros de diálogo, al manejo de errores, el control del foco de la ventana y la distribución de la aplicación a terceros.

Recuerde que la documentación de Microsoft para VBA contiene más información sobre estos temas.

- **Alguna terminología de VBA**
- **Formularios de VBA**
- **Gestión de errores**
- **Codificación de módulos de código de VBA**
- **Ejecución de macros de VBA desde una barra de herramientas o un menú**
- **Carga automática de un proyecto VBA**
- **Ejecución automática de una macro de VBA**
- **Apertura automática del IDE de VBA al cargar un proyecto**
- **Utilización de un estado de cero documentos**
- **Distribución de aplicaciones**
- **Migración de 64 bits**

Alguna terminología de VBA

En este capítulo se amplía la información sobre VBA. Los siguientes términos le ayudarán a conocer y trabajar mejor en el entorno VBA.

Proyecto

Conjunto de formularios y módulos que se agrupan en un solo archivo.

Módulo

Grupo de subrutinas y funciones (posiblemente relacionadas).

Macro

Subrutina o función pública. Las macros están a disposición del usuario como componentes ejecutables del proyecto personal.

Cuadro de diálogo

Medio utilizado para mostrar u obtener datos durante la ejecución de aplicaciones.

Forma

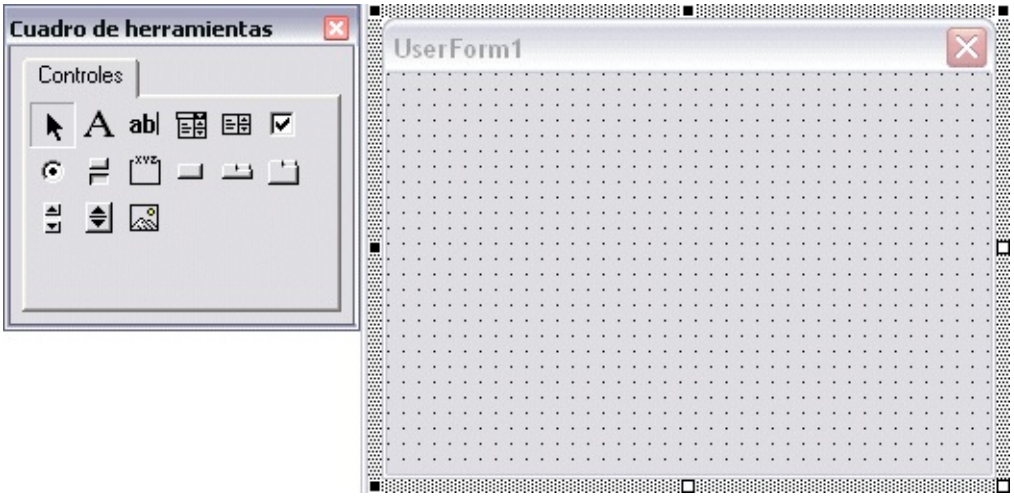
Contenedor de los controles del cuadro de diálogo.

<$nopage>controles de cuadros de diálogo.<$nopage>controles de formularios.

Formularios de VBA

Los formularios son las piezas básicas a partir de las cuales se crean los cuadros de diálogo especiales de cada aplicación. A través de los formularios personalizados se proporciona información a los usuarios, se recogen datos de los usuarios y se otorga control a éstos sobre la actividad de la aplicación.

Los formularios son como un lienzo: al principio están en blanco. Para rellenar el lienzo se utiliza una paleta. En este caso, la paleta es el cuadro de herramientas de control. El desarrollador, como el artista, coloca en el formulario los controles seleccionados en el cuadro de herramientas. Se pueden añadir tantos controles como se desee. En cualquier momento se pueden ajustar el tamaño y las propiedades de los controles y del propio formulario. Por último, se añade a los controles el código que los dota de funciones.



Aunque Visual Basic 6 admite varios tipos de formularios, VBA sólo admite los del tipo UserForm. Esto significa que algunos formularios creados y exportados en Visual Basic 6 no pueden importarse en VBA.

Los UserForms —o formularios, como se denominan en este manual— pueden ser modales o sin modo. La propiedad ShowModal de un formulario determina si son de uno u otro tipo. Los formularios modales que se presentan en la aplicación en uso deben cerrarse para que se pueda efectuar cualquier otra acción en la aplicación. Para obtener más información acerca de los formularios modales, véase Formularios modales .

Para crear un nuevo formulario en el proyecto

1. Abra la ventana Proyecto en el IDE de VBA y seleccione el proyecto al que desea añadir el formulario.
2. En el menú Insertar, elija UserForm.

   Se crea un formulario vacío y se añade al proyecto.

Para crear un formulario sin modo en el proyecto

1. Abra la ventana Proyecto en el IDE de VBA y seleccione el proyecto al que desea añadir el formulario.
2. En el menú Insertar, elija UserForm y cambie a Falso la propiedad ShowModal.
3. Añada AcFocusCtrl (*AcFocusCtrl.dll*) al cuadro de herramientas y arrastre el control al formulario.

   AcFocusCtrl mantiene el enfoque en el formulario durante la interacción con el usuario.

- **Modos de diseño y de ejecución**
- **Adición de controles a un formulario**
- **Presentación y ocultación de formularios**
- **Carga y descarga de formularios**
- **Formularios modales**

Modos de diseño y de ejecución

Los formularios se crean en el modo de diseño, en el que es posible hacer lo siguiente:

- Añadir controles
- Cambiar las propiedades
- Cambiar las propiedades de los controles
- Añadir controles al módulo

En el modo de diseño no hay interacción entre el usuario, la interfaz de usuario de AutoCAD® y el formulario.

Cuando se ejecuta la aplicación, el formulario se encuentra en modo de ejecución. En el modo de ejecución no se pueden realizar ajustes directamente en el formulario. No obstante, el formulario se muestra en la interfaz de usuario de AutoCAD y el usuario puede interactuar con él con normalidad.

<$nopage>controles de cuadros de diálogo.<$nopage>controles de formularios.

Adición de controles a un formulario

Añadir controles a un formulario es muy sencillo. Basta con seleccionar un control en el cuadro de herramientas y arrastrarlo al formulario. Al soltar el ratón, se colocará en el formulario una copia del control seleccionado. Después, se puede cambiar su posición y su tamaño. Puede añadir tantos controles como desee.

Además del método de arrastrar y soltar existen más formas de colocar controles en un formulario.

- **Cambio de tamaño y posición de los controles**
- **Utilización de controles de formato**
- **Modificación de las propiedades de un control**
- **Adición de código a un control**

Cambio de tamaño y posición de los controles

Para mover un control, basta con seleccionarlo y arrastrarlo hasta su nueva posición en el formulario.

Para cambiar el tamaño de un control, haga clic en él una vez para seleccionarlo. Cuando un control está seleccionado, su marco es visible. Para cambiar el tamaño del control, basta con seleccionar uno de los pinzamientos de tamaño (ahora visibles) del marco y arrastrarlo hasta la nueva posición. Al soltar el pinzamiento, el control cambia de tamaño de acuerdo con esa posición. Se puede seguir el mismo procedimiento para cambiar de tamaño el formulario.

Para mover o cambiar de tamaño a varios controles a la vez, selecciónelos uno a uno mientras mantiene pulsada la tecla MAYÚS. Se resaltarán todos los controles. Ahora puede desplazar o cambiar el tamaño de los controles como si se tratara de un grupo.

Para cambiar el tamaño de un control mientras se coloca

1. Seleccione el control en el cuadro de herramientas de control.
2. En el formulario, pulse, arrastre y suelte el botón del ratón. El control seleccionado quedará colocado en el formulario. El tamaño varía según se arrastra el ratón.

Para colocar varios ejemplares del mismo control

1. En el cuadro de herramientas de control, haga doble clic en el control que desee colocar.
2. En el formulario, haga clic en la posición donde desee colocar una copia del control. Desplácese a otra posición del formulario y haga clic de nuevo. Aparecerá otra copia del control. Se pueden añadir tantas copias

del control como se necesiten.

3. Cuando termine de trabajar con el control, regrese al cuadro de herramientas de control y vuelva a hacer clic en el control para que deje de estar seleccionado.

¿Comentarios?

Utilización de controles de formato

VBA proporciona varios controles de formato que ayudan a ajustar la presentación final del formulario. Estos controles pueden hallarse en el menú Formato de IDE de VBA. Permiten alinear controles entre sí, hacer que dos o más controles tengan el mismo tamaño, cambiar la distancia entre ellos y centrarlos en el formulario.

Cuando utilice los controles de formato, recuerde que es posible seleccionar varios a la vez por medio de MAYÚS.

Modificación de las propiedades de un control

Las propiedades determinan diversas características de los controles, como el tamaño, la forma, el color, la etiqueta y los valores por defecto. Se puede modificar las propiedades de un control en el modo de diseño utilizando la ventana Propiedades.

Para cambiar las propiedades de un control

1. En el formulario, seleccione un control.
2. Pulse F4 para abrir la ventana Propiedades, si aún no está abierta.
3. En la ventana Propiedades, localice la propiedad que desee modificar y seleccione el valor actual de la misma.
4. Cambie el valor por el que estime conveniente para la propiedad.

También puede cambiar la propiedad de un control durante la ejecución si escribe código para acceder a la misma. Para obtener información acerca del cambio de propiedades de un control en tiempo de ejecución, véase la documentación de Microsoft.

Adición de código a un control

Ahora que ya tiene un formulario con el aspecto adecuado, puede añadir código a los controles. Para abrir la ventana de códigos de un control, haga doble clic en él en la ventana UserForm. Se abre la ventana Código, con una subrutina creada para ese control y su procedimiento de evento predeterminado.

Puede añadir código al procedimiento de evento predeterminado o elegir un procedimiento distinto de la lista desplegable de procedimientos, que se encuentra en la esquina superior derecha de la ventana Código.

Presentación y ocultación de formularios

En este momento ya tiene un formulario con el aspecto adecuado y con código detrás de todos sus controles. El último paso es conseguir que el formulario se muestre al usuario durante la ejecución. La presentación en pantalla de los formularios se realiza a través del método Show de VBA. Puede llamar al método Show desde cualquier módulo de código de la aplicación.

El formulario creado es modal por defecto, lo que impide al usuario interactuar directamente con AutoCAD mientras el formulario esté en pantalla. Por ejemplo, el usuario no podrá seleccionar un punto u objeto del dibujo mientras esté abierto el formulario. Para que el usuario pueda acceder al dibujo de AutoCAD, deberá utilizar el método Hide de VBA. El método Hide oculta el formulario y concede al usuario un acceso limitado a AutoCAD. Cuando se utiliza el método Hide, es importante recordar que el formulario no se descarga de la memoria; conservará todos los valores actuales mientras esté oculto.

Las llamadas al método Hide se realizan igual que las del método Show.

Presentación de un formulario

En este ejemplo se muestra el formulario llamado “UserForm1”:

```
Public Sub MyApplication()
   UserForm1.Show
End Sub
```

Ahora se puede llamar a la subrutina (que hace que se presente el formulario) como macro, desde el comando VBARUN o desde un menú de AutoCAD.

Ocultación de un formulario

En este ejemplo se oculta el formulario llamado “UserForm1”:

```
Public Sub MyAppHide()
   UserForm1.Hide
End Sub
```

¿Comentarios?

Carga y descarga de formularios

Es posible que en algunas ocasiones desee cargar un formulario en memoria durante la ejecución, pero no mostrarlo. Esto puede ser útil para conseguir un control mejor cuando tiene lugar la carga en la aplicación, o cuándo el programa necesita acceder al formulario pero no hace falta mostrárselo al usuario.

Para cargar un formulario sin mostrarlo en pantalla, utilice el método Load de VBA. Posteriormente podrá utilizar el método Show para hacerlo visible en el momento adecuado durante la ejecución de la aplicación. Recuerde que el usuario no podrá interactuar con el formulario mientras no esté visible.

Si se llama al método Show y aún no se ha cargado el formulario, se cargará automáticamente.

En algunos casos es posible que desee descargar un formulario concreto. Cuando se descarga un formulario, éste desaparece de la memoria, y el sistema recupera toda la memoria asociada al mismo. Hasta que el formulario se vuelve a cargar, mediante los métodos Load o Show, ni el usuario ni el programa pueden acceder al mismo. Podría optar por descargar un formulario si sabe que no se utilizará más en la aplicación y necesita dejar libre la memoria.

El método Hide no realiza ninguna descarga. Si finaliza la ejecución de la aplicación y el formulario no se ha descargado, se descargará automáticamente. En la tabla siguiente se comparan los métodos Show, Hide, Load y Unload de VBA:

Métodos Show, Hide, Load y Unload de VBA:	
Método	**Descripción**
Show	Presenta un formulario. Si aún no se ha cargado, el

	formulario se carga automáticamente.
Hide	Oculta un formulario. El formulario no se descarga de la memoria.
Cargar	Carga un formulario en memoria, pero no lo muestra en pantalla.
Unload	Descarga un formulario de la memoria. Puede efectuarse de forma explícita desde el método Unload o de forma automática al cerrar la aplicación.

Formularios modales

Cuando se define un cuadro de diálogo como 'modal' en VBA de AutoCAD, el usuario debe responder al mismo antes de poder continuar utilizando la aplicación. El siguiente código no se ejecutará mientras no se cierre el cuadro de diálogo modal con uno de los métodos Hide o Unload. Por tanto, el programador de la aplicación debe estudiar cuidadosamente dónde y cuándo es conveniente utilizar cuadros de diálogo.

Por ejemplo, puede presentarse un cuadro de diálogo que solicite al usuario la selección de un objeto en el dibujo de AutoCAD. Para que el usuario pueda designar el objeto en la ventana de la aplicación de AutoCAD, es preciso ocultar primero el formulario mediante el método Hide. Una vez que se ha seleccionado el objeto, se puede utilizar el método Show para volver a presentar el formulario, con todos sus datos, y continuar con la aplicación.

Nota Aunque otros formularios de la aplicación se desactiven cuando se abre un cuadro de diálogo modal, no ocurre lo mismo con las otras aplicaciones.

Gestión de errores

La mayoría de los entornos de desarrollo proporcionan un sistema de tratamiento de errores por defecto. En VB y VBA, la reacción por defecto ante un error consiste en mostrar un mensaje de error y cerrar la aplicación. Si bien este comportamiento es el adecuado en las fases de desarrollo, no es el más conveniente para el usuario final. Puede haber errores que se deseen ignorar, y otros que deban producir una respuesta especial. En algunas ocasiones puede ser aconsejable suprimir la presentación del mensaje de error, o simplemente, controlar el mensaje que se presenta al usuario. Además, la terminación automática de las aplicaciones no tendrá nunca una buena acogida por parte del usuario final.

Por lo general, las rutinas de control de errores son necesarias cuando se requiere que el usuario introduzca algún dato y cuando se trabaja con entrada y salida de archivos. Recuerde que, aunque esté seguro de que el archivo necesario existe y está disponible para su proceso, pueden darse ciertas condiciones imprevistas que provoquen errores.

Nota La mayoría de ejemplos de código que se encuentran en la documentación de AutoCAD utilizan identificación de errores. De esta forma el ejemplo es menos complejo y más preciso. Pero como ocurre con todos los lenguajes de programación, es imprescindible una correcta identificación y solución de los errores para que las aplicaciones sean estables.

- **Definición de tipos de error de las aplicaciones**
- **Identificación de errores de ejecución**
- **Respuesta a errores identificados**
- **Respuesta a errores de entrada de datos del usuario de AutoCAD**

Definición de tipos de error de las aplicaciones

Existen tres tipos diferentes de errores que podrá encontrar en sus aplicaciones: errores de compilación, errores de ejecución y errores lógicos.

- Los errores de compilación se producen al crear la aplicación. En su mayoría son errores de sintaxis, problemas relativos al alcance de las variables y fallos de escritura. En VBA, estos tipos de error se detectan en el entorno de desarrollo. Cuando se introduce una línea de código incorrecta, ésta se resalta y aparece un mensaje de error que describe el problema. Los errores que se producen durante la compilación deben corregirse para que la aplicación se pueda ejecutar.
- Los errores de ejecución son un poco más difíciles de detectar y corregir. Se producen durante la ejecución del código, y a menudo incluyen la entrada de datos por parte del usuario. Por ejemplo, si la aplicación solicita al usuario que introduzca el nombre de un dibujo y el nombre que escribe no existe, se produce un error de tiempo de ejecución. Para gestionar con eficacia los errores de ejecución es necesario prever los problemas posibles y escribir el código necesario para resolver estas situaciones.
- Los errores lógicos son los que presentan mayor dificultad de detección y de resolución. Una situación en la que no existen errores de compilación ni de ejecución y el resultado del programa siga siendo incorrecto puede ser síntoma de un error lógico. Esto es lo que los programadores conocen como "bug", que puede ser muy fácil o muy difícil de identificar.

Para obtener información acerca de la localización y corrección de estos tres tipos de error, véase la documentación del entorno de desarrollo propio. Los errores específicos de AutoCAD se consideran errores de ejecución, y por lo tanto es el tipo de error que se trata con más detenimiento en esta publicación.

¿Comentarios?

Identificación de errores de ejecución

En VB y VBA, los errores de ejecución se identifican por medio de la instrucción `On Error`. Esta instrucción pone literalmente una trampa al sistema y, cuando se produce un error, el proceso se desvía hacia el gestor de errores concreto que se haya especificado. En este caso se omite la gestión de errores establecida por defecto.

La instrucción `On Error`tiene tres modalidades:

- `On Error Resume Next`
- `On Error GoTo Label`
- `On Error GoTo 0`

La instrucción `On Error Resume Next` se utiliza cuando se desea ignorar los errores. Esta instrucción identifica el error, y en vez de presentar un mensaje de error y cerrar el programa, pasa a la siguiente línea de código y continúa con la ejecución. Por ejemplo, suponga que desea crear una subrutina para iterar en el espacio modelo y cambiar el color de cada entidad, pero sabe que AutoCAD arrojará un error si se intenta asignar color a una entidad que se encuentra en una capa bloqueada. En lugar de cerrar el programa, basta con que se salte la entidad de la capa bloqueada y continúe procesando las demás entidades. Esto es, precisamente, lo que la instrucción `On Error Resume Next` permite hacer.

La instrucción `On Error GoTo Label` se utiliza para escribir rutinas específicas de control de errores. Esta instrucción identifica el error, y en vez de presentar un mensaje de error y cerrar el programa, salta a un punto concreto del código. A continuación, el código responde al error de la forma más idónea para la aplicación. Por ejemplo, el código anterior podría ampliarse para que se muestre un mensaje con el identificador de todas las entidades de la capa

bloqueada.

Gestión de errores con la secuencia On Error Resume Next

La siguiente subrutina itera en el espacio modelo y cambia a rojo el color de las entidades. Intente ejecutar esta subrutina en un dibujo que tenga algunas de sus entidades en una capa bloqueada. A continuación, amplíe con explicaciones la instrucción `On Error Resume Next` y vuelva a ejecutar la subrutina. Observará que la subrutina se interrumpe en la primera entidad de la capa bloqueada.

```
Sub Ch11_ColorEntities()
    Dim entry As Object
    On Error Resume Next
    For Each entry In ThisDrawing.ModelSpace
        entry.Color = acRed
    Next entry
End Sub
```

Gestión de errores con la instrucción On Error GoTo

La siguiente subrutina itera en el espacio modelo y cambia a rojo el color de las entidades. El gestor de errores muestra, por cada entidad de la capa bloqueada, un mensaje de error junto con el identificador de la entidad. Intente ejecutar esta subrutina en un dibujo que tenga algunas de sus entidades en una capa bloqueada. A continuación, amplíe con explicaciones la instrucción `On Error Resume GoTo MyErrorHandling` y vuelva a ejecutar la subrutina. Observará que la subrutina se interrumpe en la primera entidad de la capa bloqueada.

```
Sub Ch11_ColorEntities2()
    Dim entry As Object
    On Error GoTo MyErrorHandler
    For Each entry In ThisDrawing.ModelSpace
        entry.Color = acRed
    Next entry
    ' Important! Exit the subroutine before the error handler
    Exit Sub
MyErrorHandler:
    Msgbox entry.EntityName + " is on a locked layer." + _
                  " The handle is: " + entry.Handle
    Resume Next
```

```
End Sub
```

La instrucción `On Error GoTo 0` cancela el identificador del error actual. Las instrucciones `On Error Resume Next` y `On Error GoTo Label` siguen activas hasta que finaliza la subrutina, hasta que se declara otra rutina de control de errores o hasta que se cancela la rutina de control de errores con la instrucción `On Error GoTo 0`.

¿Comentarios?

Respuesta a errores identificados

Una vez identificado el error, ¿qué hacer con él? La respuesta depende de la naturaleza de la aplicación y del error mismo.

VB y VBA proporcionan información sobre el tipo de error identificado, por medio del objeto Err. Este objeto presenta varias propiedades: Number, Description, Source, Helpfile, HelpContext y LastDLLError. Las propiedades del objeto Err se rellenan con la información del último error que se produce. Las propiedades de número y descripción son las más importantes. La propiedad Number contiene el código de error único que tiene asociado el error, mientras que la propiedad Description incluye el mensaje de error que se presentaría.

Se puede utilizar el identificador del error para comparar la propiedad Number del error con un valor supuestamente esperado, y conocer la naturaleza del error producido. Una vez que se conozca el tipo de error, se pueden emprender las acciones necesarias para resolverlo.

Respuesta a errores de entrada de datos del usuario de AutoCAD

Los métodos que requieren la entrada de determinados datos por parte del usuario incorporan cierta cantidad de identificación de errores. Si el usuario intenta introducir datos distintos de los especificados, AutoCAD rechaza la entrada y vuelve a preguntar. El uso del método InitializeUserInput proporciona un control adicional de los datos introducidos por el usuario, pero también puede incluir condiciones adicionales que se deben verificar por medio de la identificación de errores. Si desea ver un ejemplo de identificación de errores requerida con ciertos tipos de datos introducidos por el usuario, consulte Solicitud de datos de usuario

Codificación de módulos de código de VBA

Aunque VBA no admite la creación de ejecutables, sí ofrece protección por contraseña para la visibilidad de los formularios de proyectos, las clases y los módulos en el nivel de proyectos. Puede encontrar esta función de protección en el menú del IDE de VBA. Seleccione Herramientas ➤ Propiedades de proyecto ➤ Protección.

Ejecución de macros de VBA desde una barra de herramientas o un menú

Para ejecutar una macro de VBA desde una barra de herramientas o un menú de AutoCAD, basta con modificar la propiedad Macro de dicho menú o barra de herramientas. La propiedad Macro debe tener definido Igual a.

```
-VBARUN nombrearchivo.dvb!nombremódulo.nombremacro
```

donde *nombrearchivo* es el nombre del archivo de proyecto, *nombremódulo* el nombre del módulo donde se halla la macro que se debe ejecutar y *nombremacro* el nombre de la macro. El nombre del archivo sólo se requiere cuando el archivo no se carga en la sesión actual de AutoCAD. Si se proporciona el nombre del archivo, éste se carga.

Para obtener más información acerca de la modificación de menús y barras de herramientas, véase Personalización de barras de herramientas y menús.

Carga automática de un proyecto VBA

Existen dos maneras de cargar automáticamente un proyecto de VBA:

- Cuando está cargado, VBA busca en el directorio de AutoCAD un proyecto llamado *acad.dvb*. Este archivo se carga automáticamente como proyecto por defecto.
- Excepto el predeterminado, *acad.dvb*, todos los demás proyectos se pueden utilizar cargándolos expresamente en el inicio por medio del comando VBALOAD. La siguiente muestra de código utiliza el archivo de inicio de AutoLISP para cargar VBA y un proyecto de VBA llamado *myproj.dvb* cuando se inicia AutoCAD. Abra el Bloc de notas de Windows (*notepad.exe*) y cree el archivo *acad.lsp*, o ábralo si ya existe, y escriba o añada al final las siguientes líneas:

```
(defun S::STARTUP()
(command "_VBALOAD" "myproj.dvb")
)
```

Ejecución automática de una macro de VBA

Es posible ejecutar automáticamente cualquier macro del archivo *acad.dvb*, si se le llama con la versión de línea de comando de VBARUN desde una función de inicio de AutoCAD como *acad.lsp*. Por ejemplo, para ejecutar de forma automática una macro llamada `drawline`, en primer lugar guarde la macro `drawline` en el archivo *acad.dvb*. A continuación, ejecute *notepad.exe* y cree (o añada) las siguientes líneas de *acad.lsp*:

```
(defun S::STARTUP()
     (command "_-vbarun" "drawline")
)
```

Se puede conseguir que se ejecute una macro al cargar VBA si se le asigna el nombre de macro `AcadStartup`. Una macro del archivo *acad.dvb* que se llame `AcadStartup` se ejecuta de forma automática al cargarse VBA.

Apertura automática del IDE de VBA al cargar un proyecto

En el cuadro de diálogo Abrir proyecto VBA existe una opción que permite abrir automáticamente el entorno de desarrollo interactivo. Active la casilla de selección Abrir el Editor de Visual Basic, situada en la esquina inferior izquierda del cuadro de diálogo, y el IDE de VBA se abrirá automáticamente al cargar un proyecto VBA. Esta opción permanecerá activa hasta que la desactive.

Nota Para acceder al cuadro de diálogo Abrir proyecto VBA, escriba VBALOAD en la línea de comando. Se abrirá el cuadro de diálogo y podrá elegir el proyecto que desee cargar. Si no ve el cuadro de diálogo Abrir proyecto VBA, es probable que esto se deba a que la variable de sistema FILEDIA esté desactivada. La variable de sistema activa y desactiva la presentación de los cuadros de diálogo. Para volver a activar FILEDIA asígnele el valor 1.

Utilización de un estado de cero documentos

Un estado de cero documentos en AutoCAD es aquél en el que no está abierto ningún dibujo. Deben tenerse en cuenta varias consideraciones cuando se trabaje en VBA en un estado de cero documentos:

- El objeto `ThisDrawing` no está definido cuando AutoCAD se encuentra en un estado de cero documentos. Si se intenta utilizar `ThisDrawing`, se produce un error.
- Los objetos que dependen de los documentos tampoco están definidos en un estado de cero documentos. Estos objetos son aquellos que están debajo del objeto Document en el modelo de objetos de AutoCAD. Se puede trabajar con objetos que no dependen de documentos, como Application o MenuBar.
- AutoCAD no tiene línea de comando en el estado de cero documentos. Todo intento de acceder a la línea de comando de AutoCAD en un estado de cero documentos produce un error.

Distribución de aplicaciones

Las aplicaciones VBA pueden distribuirse de dos formas:

- Incrustadas en un archivo de dibujo de AutoCAD
- Almacenadas en un archivo de proyecto VBA

Debe elegirse la opción de distribución más apropiada para la aplicación concreta. Las aplicaciones específicas del dibujo actual y que no necesitan acceder a otros dibujos suelen incrustarse en el dibujo. La incrustación de la aplicación en el dibujo garantiza que ésta se cargue al abrir el dibujo y que el usuario la tenga disponible mientras el dibujo se mantenga abierto.

Las aplicaciones que se comparten entre distintos usuarios, que se actualizan con frecuencia, que requieren abrir y cerrar otros dibujos o que no se utilizan a menudo, suele ser más conveniente guardarlas en un archivo de proyecto VBA. De esta forma, la aplicación tiene una ubicación centralizada y se asegura que todos utilizan su última versión.

Si desea más información sobre los proyectos incrustados y los archivos de proyecto de VBA, consulte Descripción de los proyectos VBA globales e incrustados.

- **Distribución de aplicaciones de Visual Basic 6**

Distribución de aplicaciones de Visual Basic 6

Las aplicaciones de Visual Basic 6, así como otras aplicaciones externas al proceso, no pueden almacenarse en los dibujos de AutoCAD. Estas aplicaciones se compilan en ejecutables independientes (EXE).

Migración de 64 bits

Microsoft no proporciona una versión de 64 bits de VBA para desarrolladores; por tanto será necesario cambiar las aplicaciones de 32 bits que se desarrollen para las versiones de AutoCAD de 64 bits. En 64 bits, VBA se ejecuta como componente fuera de proceso, al que se accede a través de una capa de "conversión" de 32 a 64 bits.

No se garantiza que el comportamiento de VBA en la versión de 64 bits de AutoCAD sea idéntico al de VBA con 32 bits de AutoCAD. Por ejemplo, si el IDE de VBA está activo o si se muestra una ventana de modos, puede que haya un pequeño retraso cuando se vuelva a pintar una ventana de AutoCAD.

Para obtener más información, consulte el *AutoCAD Migration Guide*

- **Creación de instancias de objetos de AutoCAD en VBA**
- **Gestión de ID de objetos**
- **Apéndice de métodos de 32 bits**

Creación de instancias de objetos de AutoCAD en VBA

Los sistemas operativos de 64 bits pueden ejecutar aplicaciones de 32 y 64 bits, pero no pueden mezclar estos tipos en un proceso. Por ejemplo, no es posible cargar archivos DLL de 32 bits en un proceso de 64 bits ni viceversa. Todos los componentes ejecutables (archivos EXE y DLL) que se cargan en un proceso deben coincidir en el tipo binario del proceso. Los componentes en proceso de las aplicaciones de 64 bits deben cambiarse a procesos de 64 bits siempre que sea posible.

Puede producirse un error al intentar crear un objeto nuevo. El teclado `New` de VB intentará cargar los archivos DLL COM de AutoCAD de 64 bits. Puesto que VBA es una aplicación de 32 bits, no puede cargar archivos DLL de 64 bits. Por ejemplo, un código del tipo

```
Dim color As AcadAcCmColor
Set color = New AcadAcCmColor
```

or

```
Dim color As New AcadAcCmColor
color.SomeMethod()
```

necesita modificarse o cambiarse a

```
Dim color As AcadAcCmColor
Set color = AcadApplication.GetInterfaceObject(“Autocad.AcCmColor.17
```

El problema anterior debe solucionarse mediante `AcadApplication.GetInterfaceObject(“ProgIdOfAcAnyObjec` para cualquier objeto derivado de `IDispatch`. No se espera que las clases derivadas de `IUnknown` (por ejemplo, `AcSmSheetSet, AcSmSheetMgr`, etc.) migren a VBA de 64 bits. Es recomendable cambiar estos procesos a VB .Net.

¿Comentarios?

Gestión de ID de objetos

Al iniciar AutoCAD 2008 de 64 bits, los ID de objetos se representan mediante un tipo de datos de enteros de 64 bits (`_int64`). Si se accede a estos valores en VBA de 32 bits se producirá un error de compilación. Como solución, se utiliza un nuevo conjunto de nombres de métodos con el sufijo "32" correspondientes a los métodos anteriores (por ejemplo `ObjectID32()`, `OwnerID32()`). Estos métodos utilizan el tipo de datos `LONG` que se asigna de manera interna al tipo de datos de enteros de 64 bits.

Para ser más precisos, un ID de objeto de 32 bits se crea de forma interna para cada ID de objeto necesario en VBA. Este ID se asigna a su ID real de 64 bits de manera que si se devuelve el ID de 32 bits a AutoCAD a partir de un código VBA, el ID de objeto de 64 bits se devolverá y se utilizará de forma interna para todos los fines.

En el siguiente ejemplo se muestra un código de ID de objeto de 32 bits cambiado.

Código original:

```
Dim splineObj As AcadSpline
Dim objectID As Long
objectID = splineObj.objectID
Dim tempObj As AcadObject
Set tempObj = ThisDrawing.ObjectIdToObject(objectID)
```

Código cambiado para que sea compatible con 64 bits:

```
Dim splineObj As AcadSpline
Dim objectID As Long
objectID = splineObj.objectID32
Dim tempObj As AcadObject
Set tempObj = ThisDrawing.ObjectIdToObject32(objectID)
```

Las aplicaciones de VBA también pueden utilizar un identificador de objetos en lugar de su ID de objeto. En el siguiente ejemplo se muestra cómo utilizar el identificador en lugar del ID de objeto:

Código original:

```
Dim splineObj As AcadSpline
Dim objectID As Long
objectID = splineObj.objectID
Dim tempObj As AcadObject
Set tempObj = ThisDrawing.ObjectIdToObject(objectID)
```

Código cambiado para que sea compatible con 64 bits:

```
Dim splineObj As AcadSpline
Dim objectHandle As String
objectHandle = splineObj.Handle
Dim tempObj As AcadObject
Set tempObj = ThisDrawing.HandleToObject(objectHandle)
```

Nota : Se ignorará el soporte para VBA en futuras versiones de AutoCAD. Los desarrolladores de VBA deben prepararse para cambiar su código VBA a VB.Net.

¿Comentarios?

Apéndice de métodos de 32 bits

En la siguiente tabla se enumeran los nuevos códigos añadidos en AutoCAD 2008 para las sustituciones de 32 bits:

Tabla de métodos de VBA para sistemas de 64 bits	
Método	**Descripción**
GetBlockAttributeValue32	Devuelve el valor de atributo desde una celda de bloque especificada para el objeto de definición de atributo contenido en el bloque mediante el correspondiente ID de objeto de 32 bits.
GetBlockTableRecordId32	Obtiene el ID de objeto de 32 bits del registro de la tabla de bloques asociado a la celda tipo del bloque y a `nContent`.
GetDataLink32	Devuelve el ID de objeto de 32 bits del objeto de vínculo de datos.
GetFieldId32	Devuelve el ID de objeto de 32 bits del objeto de campo asociado a la celda especificada.
GetGridLinetype32	Devuelve el objeto de ID de 32 bits del objeto de tipo de línea de

	rejilla.
Key32	Especifica el ID de objeto del objeto de origen en la operación CopyObjects de un sistema de 64 bits.
ObjectID32	Obtiene el ID de objeto de un sistema de 64 bits.
ObjectIDtoObject32	Obtiene el objeto que corresponde a un determinado ID de objeto de un sistema de 64 bits.
OwnerID32	Obtiene el ID de objeto del objeto de propietario (primario) de un sistema de 64 bits.
SetBlockAttributeValue32	Establece el valor de atributo desde una celda de bloque especificada para el objeto de definición de atributo contenido en el bloque y en `nContent` mediante el correspondiente ID de objeto de 32 bits.
SetBlockTableRecordId32	Establece el registro de tabla de bloques del ID de objeto de 32 bits asociado a la celda tipo del bloque y a `nContent`.
SetDataLink32	Establece el ID de objeto de 32 bits del objeto de vínculo de datos.
SetFieldId32	Establece el ID de objeto de 32 bits del objeto de campo asociado a la celda especificada y a `nContent`.

SetGridLinetype32	Establece el ID de objeto de 32 bits del objeto de tipo de rejilla.
Value32	Especifica el valor actual de la propiedad o del ID de objeto del objeto clonado que se acaba de crear para un sistema de 64 bits.

Interacción con otras aplicaciones y con las API de Windows

La tecnología ActiveX permite intercambiar información fácilmente con otras aplicaciones de AutoCAD y con aplicaciones compatibles con ActiveX, como Microsoft Excel o Microsoft Word. En este capítulo se exponen algunos procedimientos básicos para la interacción con otras aplicaciones.

Interacción con aplicaciones de Visual LISP

Las aplicaciones de Visual LISP® tienen acceso a todo el rango de objetos ActiveX®. Pueden llamar a todos los métodos de ActiveX, y definir y recuperar sus propiedades. Además, las aplicaciones de Visual LISP pueden ejecutar macros de VBA por medio del comando VBARUN

Las aplicaciones de ActiveX y VBA pueden ejecutar aplicaciones de Visual LISP por medio del método SendCommand. Este método permite a ActiveX y a las aplicaciones VBA enviar comandos a la línea de comando de AutoCAD.

Para obtener más información acerca del acceso a objetos ActiveX a través de Visual LISP, véase *AutoLISP Developer's Guide*.

Interacción con otras aplicaciones de Windows

La tecnología ActiveX de AutoCAD®permite intercambiar información fácilmente con otras aplicaciones compatibles con ActiveX, como Microsoft Excel y Microsoft Word. Esta capacidad permite al usuario recopilar, almacenar y presentar información de AutoCAD con formatos distintos a los dibujos de AutoCAD. También facilita la carga de información en AutoCAD desde estas aplicaciones para dirigir los procesos de creación o manipulación de objetos de AutoCAD. Un ejemplo práctico de esta tecnología es la creación de una lista de elementos, a partir de los objetos de un dibujo de AutoCAD, con el formato de hoja de cálculo de Microsoft Excel.

Ya se ha visto la forma de escribir código por medio del Modelo de objetos de ActiveX de AutoCAD. Para intercambiar información con otras aplicaciones compatibles con ActiveX basta con hacer una referencia al modelo de objetos de ActiveX de la otra aplicación y escribir el código necesario para utilizar sus objetos.

Nota En este capítulo sólo se proporciona una introducción breve de las funciones de programación entre aplicaciones. Puesto que el tema no es específico de AutoCAD, se trata también en la documentación de Microsoft y en otras guías de programación independientes.

Para intercambiar información entre modelos de objetos de ActiveX

1. Incluya referencias al modelo de objetos de ActiveX de las otras aplicaciones.

   De esta forma, el código considera los nombres y las relaciones de los objetos del otro modelo de objetos.

2. Cree una instancia de la otra aplicación.

   Se crean objetos válidos para iniciar instancias de los objetos básicos en

el otro modelo de objetos.

3. Escriba el código utilizando tanto el modelo de objetos de AutoCAD como el modelo de objetos de las otras aplicaciones.

   Aquí se realiza el intercambio de datos.

- **Referencias a la biblioteca de objetos ActiveX de otras aplicaciones**
- **Creación de una instancia de la otra aplicación**
- **Programación con objetos de otras aplicaciones**

Referencias a la biblioteca de objetos ActiveX de otras aplicaciones

Para escribir código que consulte otra aplicación, debe indicar a VBA que le permita disponer de los objetos de la otra aplicación. Para ello, se establece una referencia en la biblioteca de objetos de la otra aplicación. Se trata de un archivo situado en su ordenador en el que están definidos todos los objetos, métodos, propiedades, constantes y eventos de la aplicación.

Las referencias a una biblioteca de objetos se realizan desde el entorno IDE de VBA. En el IDE de VBA, en el menú Herramientas, existe una opción llamada Referencias. Esta opción de menú abre un cuadro de diálogo con la lista de todas las bibliotecas de objetos VBA que se encuentran en el sistema. Para establecer una referencia a una biblioteca, basta con seleccionar la biblioteca en la lista. Las bibliotecas que tienen casillas de selección marcadas cuentan ya con referencias en el proyecto actual. Por ejemplo, para añadir la biblioteca de objetos de Microsoft Excel, seleccione la entrada correspondiente de la lista.

Cuando haya creado una referencia a la biblioteca de objetos de la otra aplicación, podrá utilizar el Examinador de objetos de VBA para ver una lista de los objetos de la aplicación.

Nota Debe establecer la referencia de cada proyecto VBA que vaya a utilizar este modelo de objetos. Cuando se establece la referencia para un proyecto, ésta no se establece automáticamente para ningún otro proyecto, por motivos de eficacia.

Para establecer una referencia a la biblioteca de objetos de otra aplicación

1. En el IDE de VBA, abra el menú Herramientas y seleccione la opción de menú Referencias.

2. Busque y seleccione el elemento correspondiente a la aplicación a la que desea acceder en la lista Referencias.
3. Seleccione Aceptar para cerrar el cuadro de diálogo con las modificaciones realizadas.

Creación de una instancia de la otra aplicación

Una vez establecida una referencia a la biblioteca de objetos de una aplicación, es necesario crear una instancia de la aplicación. Esto significa que se debe iniciar la otra aplicación desde el programa, para que el código tenga acceso a los objetos con los que debe trabajar.

Para ello, primero debe declarar una variable que represente la otra aplicación. Utilice la instrucción `Dim` para seguir el mismo procedimiento que con los objetos internos. Debería incluir una cualificación del tipo de aplicación en la instrucción `Dim`. Por ejemplo, esta instrucción `Dim` declara una variable de objeto del tipo `Excel.Application`:

```
Dim ExcelAppObj as Excel.Application
```

Tras declarar la variable, utilice la instrucción `Set` con la palabra clave New para que la variable equivalga a una copia de la aplicación que se esté ejecutando. Por ejemplo, la siguiente instrucción `Set` establece la variable declarada más arriba como la aplicación Excel. La palabra clave New inicia una nueva sesión de Excel.

```
Set ExcelAppObj = New Excel.Application
```

Nota Algunas aplicaciones sólo permiten una instancia en ejecución de la aplicación por vez. Si se utiliza la nueva palabra clave New con una de estas aplicaciones, se establece una referencia a la instancia existente y no se ejecuta una nueva sesión de la aplicación.

Programación con objetos de otras aplicaciones

Después de añadir una referencia a la biblioteca de objetos y crear una instancia de la aplicación puede crear y manipular objetos en ella. Puede utilizar todos los objetos, métodos y propiedades definidos por el modelo de objetos. Por ejemplo, si se utilizan las declaraciones de variable de la sección anterior, la siguiente línea de código presenta la sesión de Excel al usuario.

```
ExcelAppObj.Visible = TRUE
```

Debería estudiar en profundidad el modelo de objetos de la aplicación para la que esté escribiendo el código. Puede utilizar el Examinador de objetos de VBA o el archivo de ayuda de la aplicación para obtener más información sobre cualquier modelo de objetos al que se haga referencia.

- **Salida de la otra aplicación**

Salida de la otra aplicación

Cuando se inicia una aplicación mediante programación, se utiliza memoria del ordenador. Salga de la aplicación en cuanto termine de utilizarla para que queden libres los recursos del sistema.

Aunque cada Modelo de objetos es distinto, casi todos tienen un método Quit para el objeto Application, que se puede utilizar para cerrar la aplicación correctamente. Por ejemplo, si se utilizan las declaraciones de variable de la sección anterior, la siguiente línea de código ejecuta la salida de Excel:

```
ExcelAppObj.Application.Quit
```

Nota Ni destruir ni superar el alcance de la variable del objeto originan obligatoriamente el fin de la aplicación. La total limpieza de la memoria sólo se asegura utilizando el método apropiado para salir de la aplicación.

Enumeración de atributos de AutoCAD en una hoja de cálculo de Excel

Esta subrutina localiza todas las referencias de bloque del dibujo actual. A continuación, encuentra los atributos enlazados a dichas referencias de bloque y los presenta en una hoja de cálculo de Excel. Para ejecutar este ejemplo, siga estos pasos:

1. Abra un dibujo que contenga referencias a bloque con atributos (el dibujo de ejemplo *sample/activeX/attrib.dwg* contiene referencias a bloque de este tipo).
2. Abra el IDE de VBA utilizando el comando VBAIDE de AutoCAD.
3. En el IDE de VBA, elija Herramientas ▸ Referencias y seleccione Microsoft Excel 8.0 Object Model.

4. Copie esta subrutina en una ventana de código de VBA y ejecútela.

```
Sub Ch12_Extract()
    Dim Excel As Excel.Application
    Dim ExcelSheet As Object
    Dim ExcelWorkbook As Object
    Dim RowNum As Integer
    Dim Header As Boolean
    Dim elem As AcadEntity
    Dim Array1 As Variant
    Dim Count As Integer
    ' Launch Excel.
    Set Excel = New Excel.Application
    ' Create a new workbook and find the active sheet.
    Set ExcelWorkbook = Excel.Workbooks.Add
    Set ExcelSheet = Excel.ActiveSheet
    ExcelWorkbook.SaveAs "Attribute.xls"
    RowNum = 1
    Header = False
    ' Iterate through model space finding
    ' all block references.
    For Each elem In ThisDrawing.ModelSpace
        With elem
            ' When a block reference has been found,
            ' check it for attributes
            If StrComp(.EntityName, "AcDbBlockReference", 1) _
                = 0 Then
                If .HasAttributes Then
                    ' Get the attributes
                    Array1 = .GetAttributes
                    ' Copy the Tagstrings for the
                    ' Attributes into Excel
                    For Count = LBound(Array1) To UBound(Array1)
                        If Header = False Then
                            If StrComp(Array1(Count).EntityName, _
                                  "AcDbAttribute", 1) = 0 Then
                                ExcelSheet.Cells(RowNum, _
                                    Count + 1).value = _
                                    Array1(Count).TagString
                            End If
                        End If
                    Next Count
                    RowNum = RowNum + 1
                    For Count = LBound(Array1) To UBound(Array1)
                        ExcelSheet.Cells(RowNum, Count + 1).value _
                                    = Array1(Count).textString
                    Next Count
                    Header = True
                End If
```

```
            End If
        End With
    Next elem
    Excel.Application.Quit
End Sub
```

¿Comentarios?

Acceso a las API de Windows desde VBA

Es posible acceder a los procedimientos API de Windows® desde la mayoría de las aplicaciones de Windows. Estos procedimientos permiten ampliar la capacidad funcional de la aplicación.

Las API de Windows permiten obtener información sobre el sistema actual, como los otros programas instalados o en ejecución, el lugar donde se encuentra la información y la configuración de control actual. También permiten acceder a los controles de sonido, multimedia y palanca de mandos. Estas tareas sólo representan alguna de las muchas funciones que proporcionan las API de Windows.

Para utilizar una API de Windows, primero se debe declarar en la aplicación. Esto se lleva a cabo con la instrucción `Declare`. La instrucción `Declare` requiere cierta información:

- El nombre de la biblioteca de vínculos dinámicos (DLL) que contenga el procedimiento que desee utilizar.
- El nombre del procedimiento conforme aparezca en la DLL.
- El nombre del procedimiento conforme desee utilizarlo en la aplicación.
- Los parámetros que el procedimiento espere recibir.
- El tipo de datos del valor devuelto (si el procedimiento llamado es una función).

La instrucción `Declare` puede colocarse en cualquiera de los módulos de VBA. Si se sitúa en un módulo estándar, el procedimiento estará disponible para todos los módulos de la aplicación, a menos que se limite su alcance con la palabra clave Private. Si se ubica la instrucción `Declare` en un módulo de formulario o clase, el procedimiento sólo estará disponible en dicho módulo.

Después de declarar un procedimiento es posible llamarlo, como a cualquier otro procedimiento de la aplicación.

No es fácil interpretar bien las instrucciones `Declare` correctas. Sin embargo, es muy fácil interpretar `Declare` de modo incorrecto, y las consecuencias pueden ser catastróficas. No olvide guardar toda la información de las aplicaciones activas antes de probar una instrucción `Declare` nueva.

Para facilitar el uso de las instrucciones `Declare`, Microsoft proporciona un archivo que presenta en una lista muchas de las instrucciones de uso más frecuente. El archivo se llama *Win32api.txt* y se suministra con Visual Basic 6 y con Office. Puede buscar en éste el procedimiento que necesita y copiar la declaración `Declare` en el código.

La documentación de Microsoft VBA contiene más información sobre la instrucción `Declare` y un ejemplo de su uso. La referencia de las API de Microsoft Windows forma parte de los CD que se entregan a los programadores suscritos al Centro de Recursos para Desarrolladores de Microsoft y proporciona una referencia de todos los procedimientos disponibles en las API de Windows. El libro *Visual Basic Programmer's Guide to the Win32 API*, de Dan Appleman, constituye un excelente recurso para los programadores de Visual Basic 6.

<$nopage>aprendizaje:

Aprendizaje de ActiveX/VBA: Diseño del camino de jardín

Este aprendizaje muestra cómo utilizar ActiveX y Visual Basic para Aplicaciones (VBA) y cómo añadir una macro a AutoCAD. Se orienta hacia la arquitectura paisajística, pero los conceptos que contiene se pueden aplicar a cualquier especialidad.

Este Aprendizaje está destinado al usuario avanzado de AutoCAD que a su vez es principiante en programación VBA.

<$nopage>aprendizaje:

Inspeccionar el entorno

Para el aprendizaje, necesitará el entorno de desarrollo integrado de VBA (VBA IDE) de AutoCAD®. VBA IDE se instala automáticamente con la opción de instalación Completa del programa de instalación de AutoCAD. Si seleccionó la opción de instalación Personalizada en el momento de instalar AutoCAD, VBA IDE puede no haberse instalado. Es posible que tenga que instalarlo ejecutando de nuevo el programa de instalación de AutoCAD.

Para comprobar si VBA IDE está instalado

1. Inicie AutoCAD.
2. En la línea de comando, escriba **vbaide** y pulse INTRO.

   Si se abre VBA IDE, esto significa que está instalado. Si aparece el mensaje “AutoCAD VBA no se encuentra instalado”, VBA IDE no está instalado.

Definir el objetivo

El objetivo de este aprendizaje es desarrollar una nueva macro para AutoCAD que dibuje el camino de un jardín y lo rellene con losetas circulares de cemento. La nueva macro tendrá la siguiente secuencia de solicitudes:

Command: **gardenpath**

Punto inicial del camino: *El usuario especificará el punto inicial*

Punto final del camino: *El usuario especificará el punto final*

Mitad de la anchura del camino: *El usuario especificará un número*

Radio de las losetas: *El usuario especificará un número*

Espacio entre las losetas: *El usuario especificará un número*

En primer lugar, la macro solicitará al usuario que especifique los puntos inicial y final que determinarán la línea de centro del camino. Luego, solicitará al usuario que especifique la mitad de la anchura del camino y el radio de las losetas circulares. Finalmente, el usuario especificará el espacio entre las losetas. Usará la mitad de la anchura del camino en vez de la anchura completa puesto que es más fácil visualizar la mitad de la anchura desde la línea de centro del camino.

La primera función

La macro Gardenpath se desarrolla utilizando una serie de funciones y subrutinas. Muchas subrutinas requieren la manipulación de ángulos. Puesto que ActiveX especifica ángulos en radianes, pero la mayoría de los usuarios utiliza grados para medir ángulos, comenzaremos por crear una función que convierta grados a radianes.

Para convertir grados a radianes

1. En la línea de comando, escriba **vbaide** y pulse INTRO.
2. En VBA IDE, en el menú Ver, pulse Código para abrir la ventana Código.
3. Escriba el siguiente código en la ventana Código:

```
Const pi = 3.14159
' Conversión de un ángulo en grados a radianes
Function dtr(a As Double) As Double
    dtr = (a / 180) * pi
End Function
```

Observe que tan pronto como se pulsa INTRO después de escribir la línea `Function dtr(a As Double) As Double`, `End Function` se añade automáticamente. Esto garantiza que todas las subrutinas y funciones tienen una instrucción `End` asociada.

Ahora revise el código. Primero, la constante pi se define con el valor de 3.14159. Esto permite que se utilice la palabra *pi* en lugar de tener que teclear 3.14159 cada vez que vaya a usar el valor.

A continuación, define una función llamada `dtr` (abreviación de grados a radianes). La función `dtr` toma un argumento, *a*, que es el ángulo en grados. El resultado se obtiene dividiendo el ángulo en grados por 180 y,

a continuación, multiplicando su valor por pi. La línea que comienza por una comilla simple es un comentario. VBA ignora todo el texto que haya en una línea después de una comilla simple.

Ahora esta función puede utilizarse en otras subrutinas de todo el proyecto

4. Guarde su trabajo. Pulse Archivo ➤ Guardar Global1. Escriba *gardenpath.dvb* como nombre del proyecto.

A continuación, añadirá una función para calcular la distancia entre puntos.

Para calcular la distancia entre dos puntos

1. Escriba el siguiente código después de la función `dtr`:

```
' Cálculo de la distancia entre dos puntos
Function distance(sp As Variant, ep As Variant) _
 As Double
    Dim x As Double
    Dim y As Double
    Dim z As Double
    x = sp(0) - ep(0)
    y = sp(1) - ep(1)
    z = sp(2) - ep(2)
    distance = Sqr((Sqr((x ^ 2) + (y ^ 2)) ^ 2) + (z ^ 2))
End Function
```

2. Guarde su trabajo.

Obtención de datos

La macro Gardenpath pregunta al usuario dónde debe dibujarse el camino, qué anchura debe tener, de qué tamaño son las losetas de cemento y cuál es el espacio entre éstas. Ahora definirá una subrutina que solicita al usuario todos estos datos y que calcula diversos números que se utilizarán en el resto de la macro.

En esta subrutina, utilizará los métodos de entrada de datos de usuario del objeto Utility.

Declaración de variables

La siguiente subrutina utiliza diversas variables. Todas las variables deben declararse previamente para que la subrutina pueda acceder a ellas.

En VBA IDE, escriba el siguiente código en la ventana Código, inmediatamente después de la línea `Const pi = 3.14159`:

```
Private sp(0 To 2) As Double
Private ep(0 To 2) As Double
Private hwidth As Double
Private trad As Double
Private tspac As Double
Private pangle As Double
Private plength As Double
Private totalwidth As Double
Private angp90 As Double
Private angm90 As Double
```

Ahora observe las dos listas desplegables de la parte superior de la ventana Código. Estas listas se denominan cuadro Objeto y cuadro Procedimiento/Evento y actualmente muestran respectivamente los términos (General) y (Declaraciones). Estas listas muestran la sección del código en la que está trabajando en este momento, y le permiten desplazarse rápidamente a otra sección simplemente seleccionándola en la lista. La sección (Declaraciones) es el lugar apropiado para declarar variables que va a utilizar en más de una subrutina.

Escritura de la subrutina gpuser

La subrutina `gpuser` solicita al usuario la información necesaria para dibujar un camino de jardín. Escriba lo siguiente después de la función `distance`:

```
' Adquisición de información para el camino del jardín
Private Sub gpuser()
    Dim varRet As Variant
    varRet = ThisDrawing.Utility.GetPoint( _
     , "Punto inicial del camino: ")
    sp(0) = varRet(0)
    sp(1) = varRet(1)
    sp(2) = varRet(2)
    varRet = ThisDrawing.Utility.GetPoint( _
     , "Punto final del camino: ")
    ep(0) = varRet(0)
    ep(1) = varRet(1)
    ep(2) = varRet(2)
    hwidth = ThisDrawing.Utility. _
     GetDistance(sp, "Mitad de anchura del camino: ")
    trad = ThisDrawing.Utility. _
     GetDistance(sp, "Radio de las losetas: ")
    tspac = ThisDrawing.Utility. _
     GetDistance(sp, "Espacio entre losetas: ")
    pangle = ThisDrawing.Utility.AngleFromXAxis( _
     sp, ep)
    totalwidth = 2 * hwidth
    plength = distance(sp, ep)
    angp90 = pangle + dtr(90)
    angm90 = pangle - dtr(90)
End Sub
```

En la subrutina `gpuser`, la línea `Dim varRet As Variant` declara la variable `varRet`. Puesto que esta variable se utiliza solamente en esta subrutina, puede declararse aquí localmente, en vez de hacerlo en la sección (Declaraciones).

La siguiente línea, `varRet = ThisDrawing.Utility.GetPoint( , "Punto inicial del camino: ")`, llama al método GetPoint. El carácter de subrayado sirve para que una línea larga sea más fácil de leer, ya que indica a VBA que debe leer esa línea y la siguiente como si formaran una sola línea. El carácter de subrayado puede eliminarse colocando todo el código en una única línea.

Para acceder al método GetPoint, antes debe ir al objeto `ThisDrawing` que representa el dibujo actual. Después de escribir `ThisDrawing` se escribe un punto (`.`), lo que significa que va a acceder a algo que hay dentro de ese objeto. Después del punto, se escribe `Utility` y otro punto. Una vez más, va a acceder a algo que hay dentro del objeto Utility. Finalmente, escriba `GetPoint`, que es el nombre del método que se está invocando.

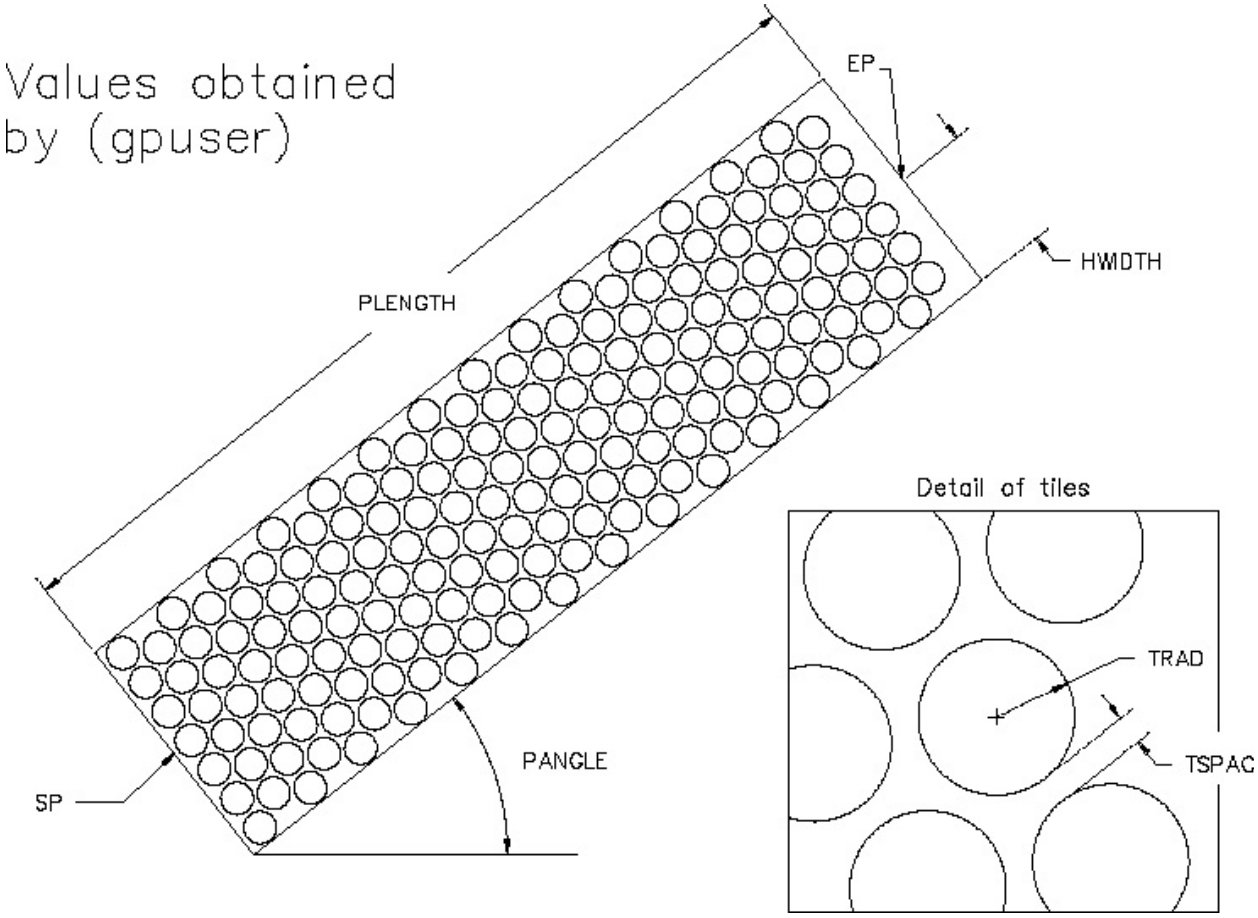
El método GetPoint toma dos parámetros. El primer parámetro es opcional y no se utilizará. Deje el parámetro en blanco y escriba únicamente una coma para marcar su ubicación. El segundo parámetro es la solicitud, que también es opcional. Para este parámetro, ha escrito una cadena que solicita al usuario que especifique el punto inicial. El punto especificado por el usuario se coloca en la variable `varRet`. Las tres líneas siguientes de la subrutina copian el punto devuelto por el usuario en la matriz `sp`.

El punto final se obtiene de la misma forma.

El método GetDistance se utiliza para obtener la mitad de la anchura del camino (`hwidth`), el radio de las losetas (`trad`), y el espacio entre éstas (`tspac`). El método GetDistance utiliza dos parámetros. El primer parámetro es un punto base. Para este valor, usted determina el punto inicial. El segundo parámetro es la solicitud, para la que proporciona una cadena que solicita al usuario el dato correspondiente. Lo interesante acerca del método GetDistance es que puede devolver tanto un valor escrito en la línea de comando como la distancia entre el punto inicial y un punto seleccionado en AutoCAD.

La subrutina continua calculando diversas variables utilizadas más tarde en la macro. La variable `pangle` se define con el ángulo entre los puntos inicial y final y se halla utilizando el método AngleFromXAxis. La anchura del camino se halla multiplicando la mitad de la anchura por dos. La variable `plength` se define como la longitud del camino y se halla utilizando la función distancia escrita anteriormente. Finalmente, se calcula y se guarda el ángulo del camino más y menos 90 grados en `angp90` y `angm90`, respectivamente.

La siguiente ilustración muestra la forma en la que las variables obtenidas por `gpuser` especifican las dimensiones del camino.



Guarde su trabajo.

¿Comentarios?

Dibujo del contorno del camino

Ahora que ha obtenido la ubicación y la anchura del camino, puede dibujar su contorno. Añada el siguiente código bajo la subrutina `gpuser`:

```
' Dibujo del contorno del camino
Private Sub drawout()
    Dim points(0 To 9) As Double
    Dim pline As AcadLWPolyline
    Dim varRet As Variant
    varRet = ThisDrawing.Utility.PolarPoint( _
        sp, angm90, hwidth)
    points(0) = varRet(0)
    points(1) = varRet(1)
    points(8) = varRet(0)
    points(9) = varRet(1)
    varRet = ThisDrawing.Utility.PolarPoint( _
        varRet, pangle, plength)
    points(2) = varRet(0)
    points(3) = varRet(1)
    varRet = ThisDrawing.Utility.PolarPoint( _
        varRet, angp90, totalwidth)
    points(4) = varRet(0)
    points(5) = varRet(1)
    varRet = ThisDrawing.Utility.PolarPoint( _
        varRet, pangle + dtr(180), plength)
    points(6) = varRet(0)
    points(7) = varRet(1)
    Set pline = ThisDrawing.ModelSpace. _
     AddLightWeightPolyline(points)
End Sub
```

Esta subrutina dibuja el contorno del camino utilizando el método AddLightweightPolyline. Este método requiere un parámetro: una matriz de puntos que genere la polilínea. Debe hallar todos los puntos que forman el objeto de polilínea y colocarlos en una matriz en el orden en que deben dibujarse. Para esta polilínea, los puntos necesarios son los vértices del camino.

Para hallar los vértices del camino, utilice el método PolarPoint. Este método encuentra un punto que está a un ángulo y una distancia determinados desde un punto base. Comience por el punto inicial (`sp`) y encuentre el primer vértice del camino trabajando en dirección contraria a las agujas del reloj. Este vértice estará a una distancia equivalente a la mitad de la anchura del camino (`hwidth`) y a -90 grados del ángulo del camino. Puesto que desea dibujar un rectángulo cerrado para el camino, ese punto se convierte en el primer y último punto de la matriz. Por lo tanto, las coordenadas *X* e *Y* obtenidas con el método PolarPoint se desplazan a la primera y a la última posición de la matriz de puntos.

Los restantes vértices del camino se hallan de la misma forma utilizando la longitud y la anchura del camino (`plength` y `width`), y el ángulo del camino. Cada vez que se invoca el método PolarPoint, las coordenadas obtenidas (`varRet`) se copian en la matriz de puntos.

Una vez identificados los vértices en la matriz de puntos, se invoca el método AddLightweightPolyline. Observe que este método es invocado desde el objeto ModelSpace. Si ejecutara esta macro, vería que la polilínea todavía no es visible en AutoCAD. La polilínea no será visible hasta que actualice la visualización, cosa que hará más tarde.

Dibujo de las losetas

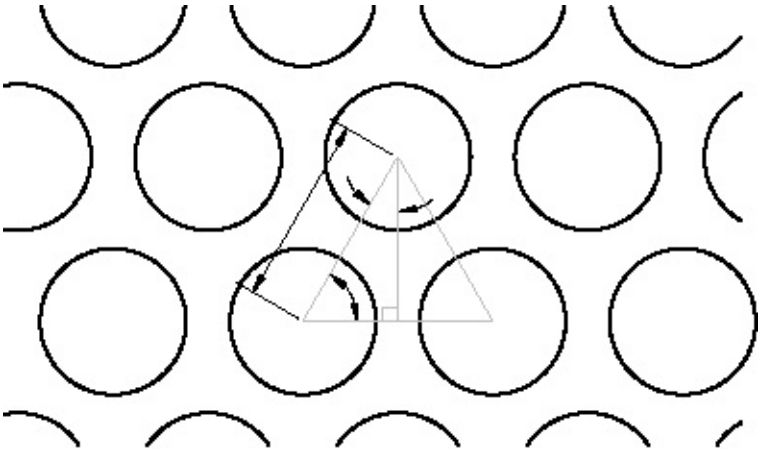
Ahora que se ha desarrollado la subrutina de entrada de datos de usuario y la subrutina para dibujar el contorno, ya se puede rellenar el camino con losetas circulares. Esta tarea requiere algo de geometría.

En VBA IDE, escriba el siguiente código en la ventana Código, después de la rutina `drawout`:

```
' Colocación de una hilera de losetas a lo largo de la distancia dad
' y posiblemente desfase de ésta
Private Sub drow(pd As Double, offset As Double)
    Dim pfirst(0 To 2) As Double
    Dim pctile(0 To 2) As Double
    Dim pltile(0 To 2) As Double
    Dim cir As AcadCircle
    Dim varRet As Variant
    varRet = ThisDrawing.Utility.PolarPoint( _
     sp, pangle, pd)
    pfirst(0) = varRet(0)
    pfirst(1) = varRet(1)
    pfirst(2) = varRet(2)
    varRet = ThisDrawing.Utility.PolarPoint( _
     pfirst, angp90, offset)
    pctile(0) = varRet(0)
    pctile(1) = varRet(1)
    pctile(2) = varRet(2)
    pltile(0) = pctile(0)
    pltile(1) = pctile(1)
    pltile(2) = pctile(2)
    Do While distance(pfirst, pltile) < (hwidth - trad)
        Set cir = ThisDrawing.ModelSpace.AddCircle( _
         pltile, trad)
        varRet = ThisDrawing.Utility.PolarPoint( _
         pltile, angp90, (tspac + trad + trad))
        pltile(0) = varRet(0)
        pltile(1) = varRet(1)
        pltile(2) = varRet(2)
```

```
    Loop
    varRet = ThisDrawing.Utility.PolarPoint( _
     pctile, angm90, tspac + trad + trad)
    pltile(0) = varRet(0)
    pltile(1) = varRet(1)
    pltile(2) = varRet(2)
    Do While distance(pfirst, pltile) < (hwidth - trad)
        Set cir = ThisDrawing.ModelSpace.AddCircle( _
         pltile, trad)
        varRet = ThisDrawing.Utility.PolarPoint( _
         pltile, angm90, (tspac + trad + trad))
        pltile(0) = varRet(0)
        pltile(1) = varRet(1)
        pltile(2) = varRet(2)
    Loop
End Sub
' Dibujo de las hileras de losetas
Private Sub drawtiles()
    Dim pdist As Double
    Dim offset As Double
    pdist = trad + tspac
    offset = 0
    Do While pdist <= (plength - trad)
        drow pdist, offset
        pdist = pdist + ((tspac + trad + trad) * Sin(dtr(60)))
        If offset = 0 Then
            offset = (tspac + trad + trad) * Cos(dtr(60))
        Else
            offset = 0
        End If
    Loop
End Sub
```

Para comprender cómo funcionan estas subrutinas, consulte la siguiente ilustración. La subrutina `drow` dibuja una hilera de losetas a una distancia dada a lo largo del camino especificada por su primer argumento, y desfasa la hilera perpendicularmente al camino con una distancia especificada por el segundo argumento. Se desea desfasar las losetas en hileras alternas para que cubran más espacio y se distribuyan de forma más estética.



La subrutina `drow` halla la ubicación de la primera hilera mediante el método PolarPoint para desplazarla a lo largo del camino con la distancia especificada por el primer argumento. La subrutina vuelve a utilizar entonces el método PolarPoint para desplazarse perpendicularmente al camino para efectuar el desfase. La subrutina utiliza la instrucción `While` para continuar dibujando círculos hasta que se encuentra el final del camino. El método PolarPoint de la primera instrucción `While` se desplaza a la siguiente posición de loseta creando un espacio equivalente a dos radios de loseta (`trad`) más un espacio entre losetas (`tspac`). El segundo bucle `while` dibuja entonces las losetas de la hilera en la otra dirección hasta que se encuentra el otro borde.

La subrutina `drawtiles` invoca `drow` repetidamente hasta que se dibujan todas las hileras de losetas. La subrutina `While loop` recorre paso a paso el camino, invocando `drow` para cada hilera. Las losetas de las hileras adyacentes forman triángulos equiláteros, tal como se muestra en la ilustración anterior. Las aristas de estos triángulos equivalen al doble del radio de la loseta más el espacio entre losetas. Por lo tanto, por la trigonometría, la distancia a lo largo del camino entre hileras es el seno de 60 grados multiplicado por esta cantidad, y el desfase de las hileras impares es el coseno sesenta grados multiplicado por esta cantidad.

La instrucción `If` se utiliza en `drawtiles` para desfasar hileras alternas. Si el desfase es igual a 0, defínalo como el espacio entre los centros de las hileras multiplicadas por el coseno de 60 grados, tal como se explicó anteriormente. Si el desfase no es igual a 0, establézcalo en 0. Esto alterna el desfase de las hileras de la forma deseada.

Guarde su trabajo.

¿Comentarios?

Integración de los elementos

Ahora ya es posible combinar las subrutinas en la macro Gardenpath. En VBA IDE escriba el siguiente código en la ventana Código, después de la subrutina`drawtiles`:

```
' Ejecución del comando, invocando las funciones constituyentes
Sub gardenpath()
    Dim sblip As Variant
    Dim scmde As Variant
    gpuser
    sblip = ThisDrawing.GetVariable("blipmode")
    scmde = ThisDrawing.GetVariable("cmdecho")
    ThisDrawing.SetVariable "blipmode", 0
    ThisDrawing.SetVariable "cmdecho", 0
    drawout
    drawtiles
    ThisDrawing.SetVariable "blipmode", sblip
    ThisDrawing.SetVariable "cmdecho", scmde
End Sub
```

La subrutina`path` invoca`gpuser` para obtener la entrada de los datos necesarios. El método GetVariable se utiliza entonces para obtener los valores actuales de las variables de sistema BLIPMODE y CMDECHO y guarda estos valores como `sblip` y `scmde`. La subrutina utiliza entonces el método SetVariable para establecer ambas variables de sistema en 0, desactivando marcas auxiliares y eco de comandos. A continuación, se dibuja el camino usando las subrutinas `drawout` y `drawtiles`. Finalmente, se utiliza el método SetVariable para restablecer el valor original de las variables de sistema.

Como puede verse, ésta es la única subrutina, entre las que ha escrito, que no comienza con la palabra clave Private, que garantiza que la subrutina sólo puede invocarse desde el módulo actual. Puesto que la subrutina `gardenpath` debe estar disponible para el usuario, debe omitirse la palabra clave Private.

Guarde su trabajo.

¿Comentarios?

Ejecución del código paso a paso

Ahora ejecute la macro, recorriendo el código paso a paso a medida que se ejecuta.

En el menú Herr. de AutoCAD, pulse Macro ➤ Macros. En el cuadro de diálogo Macros, seleccione *ThisDrawing.gardenpath* y pulse Entrar.

VBA IDE aparecerá en primer plano en la pantalla, y la primera línea de la macro `gardenpath` aparecerá resaltada. La línea resaltada es la línea de código que está apunto de ejecutarse. Para ejecutar la línea, pulse F8. La siguiente línea de código que debe ejecutarse es la subrutina `gpuser`. Para ejecutar paso a paso la subrutina `gpuser` vuelva a pulsar F8.

Ahora está al principio de la rutina `gpuser`. Pulse F8 una vez más para resaltar el primer método GetPoint. Antes de ejecutar esta línea abra la ventana Locales pulsando Ver ➤ Ventana Locales. Esta ventana se muestra en la parte inferior de VBA IDE. Todas las variables locales y sus valores se muestran en la ventana Locales mientras se ejecuta la macro.

Ahora pulse F8 para ejecutar el método GetPoint. Observe que el resaltado desaparece y no se presenta nuevo código. Esto es porque el método GetPoint está esperando a que el usuario especifique un punto en AutoCAD. Vuelva a la ventana de AutoCAD. Verá la solicitud que ha especificado en la llamada GetPoint de la línea de comandos. Especifique un punto.

El control vuelve ahora a la macro. La línea que sigue a la llamada al método GetPoint queda resaltada. Continúe la ejecución paso a paso del código pulsando F8. Recuerde volver a la ventana de AutoCAD cuando tenga que introducir datos.

Ejecución de la macro

No es necesario recorrer paso a paso el código cada vez que se ejecuta la macro. Se puede ejecutar la macro desde el menú Herr. pulsando Macro ➤ Macros, seleccionando una macro y pulsando Ejecutar. Esto le permite ver el flujo de ejecución de la misma forma en que lo haría el usuario. Ejecute la macro desde AutoCAD, especificando los siguientes valores:

Punto inicial del camino: **2, 2**

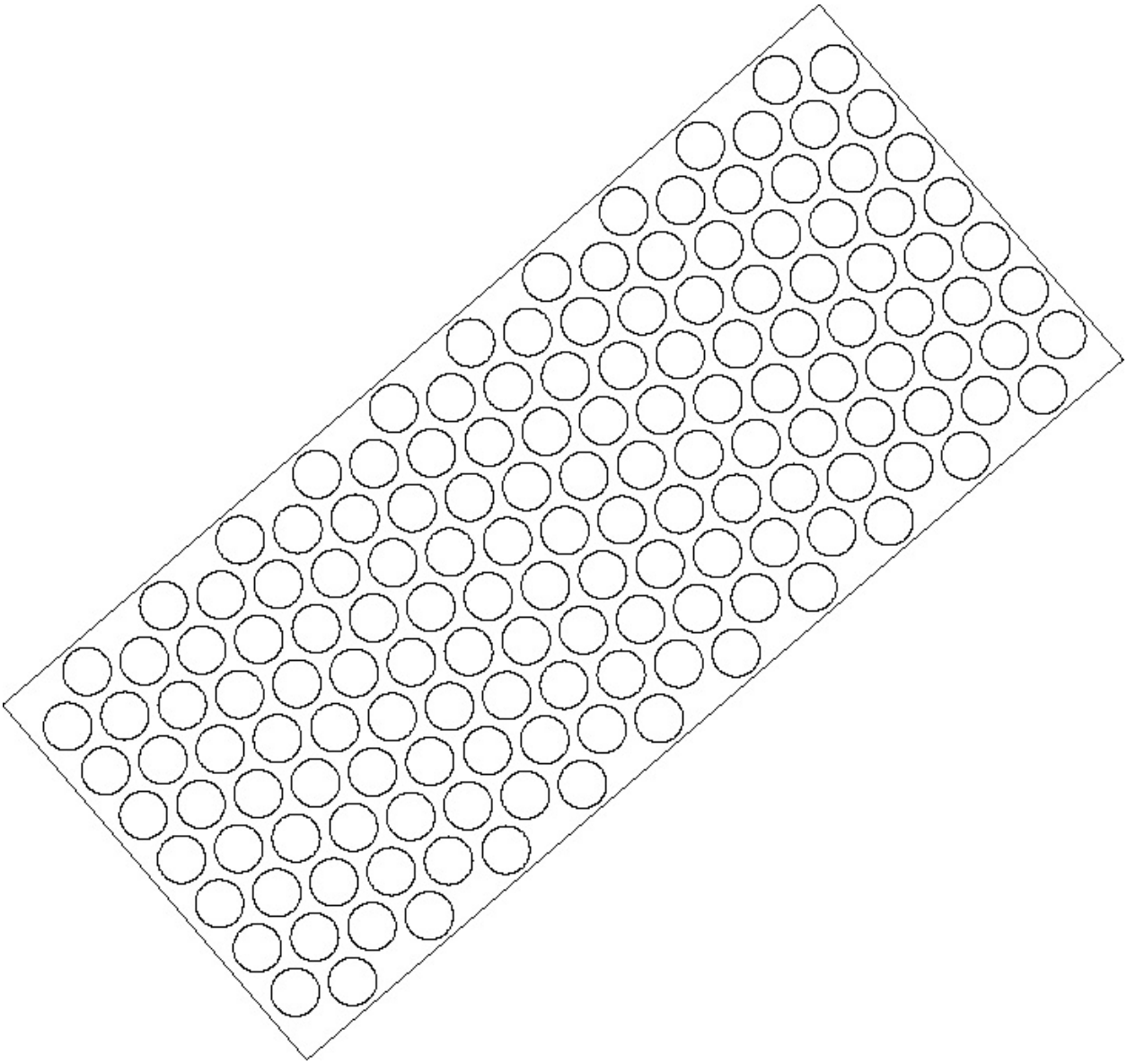
Punto final del camino: **9, 8**

Mitad de anchura del camino: **2**

Radio de las losetas: **0,2**

Espacio entre losetas: **0,1**

Este ejemplo debería dibujar un camino de jardín como el que se muestra en la siguiente figura:



Puede experimentar con la macro Gardenpath especificando con datos diferentes.

¿Comentarios?

Adición de interfaz de cuadro de diálogo

La macro Gardenpath se ha escrito para aceptar la introducción de datos en la línea de comando. Para añadir cuadros de diálogo, utilice los formularios de VBA IDE.

Primero, copie la versión terminada de *gardenpath.dvb* en otro archivo, *gpdialog.dvb*. Luego arrastre *gpdialog.dvb* a AutoCAD.

- **Creación del cuadro de diálogo**
- **Utilización de la ventana Proyecto para navegar por el proyecto**
- **Actualización del código existente**
- **Adición de código al cuadro de diálogo**

Creación del cuadro de diálogo

El cuadro de diálogo que va a crear contiene dos botones de opción (si se selecciona uno, el otro se deselecciona) para escoger la forma de la loseta: circular o poligonal. El cuadro de diálogo incluye también tres cajas de texto para introducir los siguientes valores numéricos: el radio de las losetas, el espaciado entre las mismas y el número de lados de la loseta (que está sólo disponible si se ha seleccionado la opción Polígono).

Para crear un cuadro de diálogo en VBA IDE

1. En el menú Insertar, pulse UserForm para abrir un nuevo formulario. Se muestran dos ventanas, un cuadro de herramientas y un formulario en blanco de usuario.
2. Seleccione y arrastre uno por uno los siguientes controles desde el cuadro de herramientas y sitúelos en el formulario de usuario. Tendrá que colocar dos botones de opción (

   ), tres etiquetas (

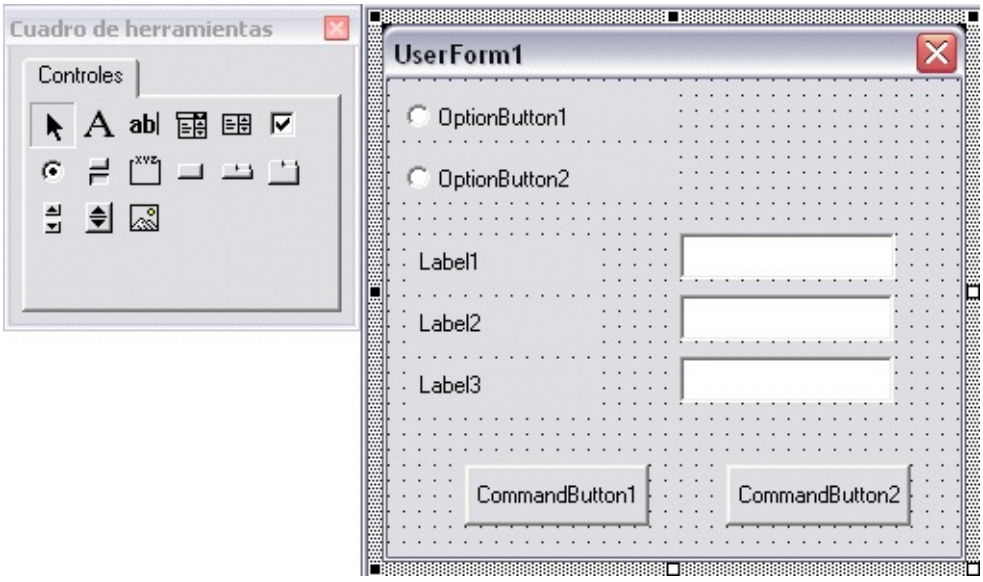
   ), tres cuadros de texto (

) y dos botones de comando (

   ), tal como se aprecia en el siguiente formulario:



3. Cierre el cuadro de herramientas.

Para establecer las propiedades de los controles de botón de opción

1. En el formulario de usuario, seleccione el control OptionButton1. En el menú Ver, pulse Ventana Propiedades y cambie las siguientes propiedades para OptionButton1:

   (Name) = gp_poly

   Caption = Polígono

   ControlTipText = Loseta en forma de polígono

   Accelerator = P

2. En el formulario de usuario, seleccione el control OptionButton2. En la ventana Propiedades, cambie las siguientes propiedades para OptionButton2:

   (Name) = gp_circ

   Caption = Círculo

   ControlTipText = Loseta en forma de círculo

   Accelerator = I

Para definir las propiedades de los controles de etiqueta

1. En el formulario de usuario, seleccione el control Label1. En la ventana

Propiedades, cambie las siguientes propiedades para Label1:

(Name) = label_trad

Caption = Radio de las losetas

TabStop = True

2. En el formulario de usuario, seleccione el control Label2. En la ventana Propiedades, cambie las siguientes propiedades para Label2:

   (Name) = label_tspac

   Caption = Espacio entre losetas

   TabStop = True

3. En el formulario de usuario, seleccione el control Label3. En la ventana Propiedades, cambie las siguientes propiedades para Label3:

   (Name) = label_tsides

   Caption = Número de caras

   TabStop = True

Para definir las propiedades de los controles del cuadro de texto

1. En el formulario de usuario, seleccione el control TextBox1. En la ventana Propiedades, cambie las siguientes propiedades para TextBox1:

   (Name) = gp_trad

2. En el formulario de usuario, seleccione el control TextBox2. En la ventana Propiedades, cambie las siguientes propiedades para TextBox2:

   (Name) = gp_tspac

3. En el formulario de usuario, seleccione el control TextBox3. En la ventana Propiedades, cambie las siguientes propiedades para TextBox3:

   (Name) = gp_tsides

Para establecer las propiedades de los controles de botón de comando y la ventana de formulario

1. En el formulario de usuario, seleccione el control CommandButton1. En la ventana Propiedades, cambie las siguientes propiedades para

CommandButton1:

(Name) = accept

Caption = Aceptar

ControlTipText = Acepta las opciones

Accelerator = O

Default = True

2. En el formulario de usuario, seleccione el control CommandButton2. En la ventana Propiedades, cambie las siguientes propiedades para CommandButton2:

   (Name) = cancel

   Caption = Cancelar

   ControlTipText = Cancela la operación

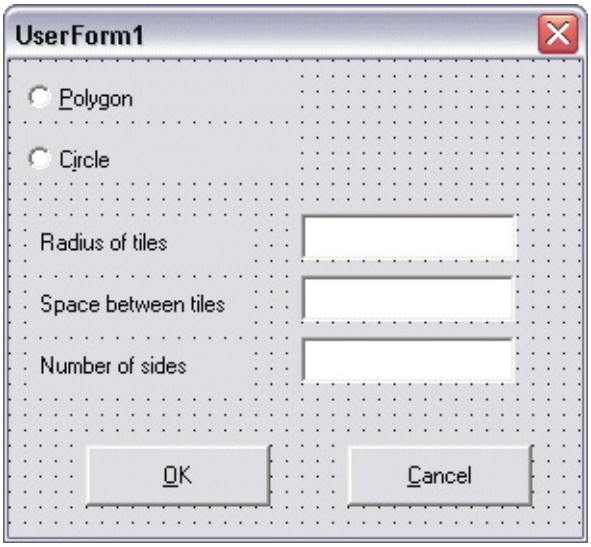
   Accelerator = C

3. Seleccione todo el formulario haciendo clic en el fondo del formulario, lejos de cualquier control. En la ventana Propiedades, cambie las siguientes propiedades para el formulario:

   (Name) = gpDialog

   Caption = Camino de jardín

   El formulario deberá ahora tener este aspecto:



4. Guarde su trabajo.

¿Comentarios?

Utilización de la ventana Proyecto para navegar por el proyecto

En VBA IDE, la ventana Proyecto contiene el nombre y la ubicación del proyecto, una carpeta llamada *AutoCAD Objetos* y una carpeta llamada *Formularios*. (Puede que tenga que pulsar Alternar carpetas para ver las carpetas.) Cuando se abre la carpeta *AutoCAD Objetos* (puede que ya esté abierta), puede verse un icono de dibujo y el nombre `ThisDrawing`. Al abrir la carpeta *Formularios* (puede que ya esté abierta), puede verse un icono de formulario y el nombre `gpDialog`, el formulario que acaba de crear.

Puede utilizar la ventana Proyecto para navegar por el código y para que le ayude a saber dónde está trabajando. Por ejemplo, para ver el código asociado con el formulario que ha creado, resalte `gpDialog` en la ventana Proyecto y pulse Ver código.

Se abre la ventana Código correspondiente al formulario.

Resalte `ThisDrawing` en la ventana Proyecto. Puede ver el código haciendo clic en Ver código. Todo el código que ha escrito está en esta ventana.

Actualización del código existente

Ahora que ha creado un cuadro de diálogo, puede añadir o modificar código.

Para modificar el código existente

1. Abra el código correspondiente a `ThisDrawing`, si todavía no está abierto.
2. Actualice las siguientes líneas de la sección Declaraciones:

```
Public trad As Double ' Actualizado
Public tspac As Double ' Actualizado
Public tsides As Integer ' Adición
Public tshape As String ' Adición
```

   Puesto que el código del formulario accede a `trad` y `tspac`, ha actualizado sus definiciones para hacerlas públicas. Las variables privadas sólo están disponibles en el módulo en el que se han definido, por lo que las variables deben convertirse en públicas. Además, ha añadido `tsides` para el número de lados de las losetas poligonales y `tshape` para que el usuario seleccione la forma de las losetas, que puede ser un círculo o un polígono.

3. Vaya a la subrutina `gpuser`. Elimine las dos líneas que obtienen el radio de las losetas y el espacio entre ellas, puesto que esta información se obtiene ahora a través del formulario. En concreto, elimine lo siguiente:

```
trad = ThisDrawing.Utility. _
     GetDistance(sp, "Radio de las losetas: ")
tspac = ThisDrawing.Utility. _
     GetDistance(sp, "Espacio entre losetas: ")
```

4. Añada las líneas que cargan y muestran el formulario. Añada las siguientes líneas en el lugar de las líneas eliminadas en el paso 3:

```
Load gpDialog
gpDialog.Show
```

5. Añada una subrutina al final del archivo de código que dibuja tanto las losetas circulares como las losetas poligonales:

```
'Dibuja la loseta con la forma seleccionada
Sub DrawShape(pltile)
    Dim angleSegment As Double
    Dim currentAngle As Double
    Dim angleInRadians As Double
    Dim currentSide As Integer
    Dim varRet As Variant
    Dim aCircle As AcadCircle
    Dim aPolygon As AcadLWPolyline
    ReDim points(1 To tsides * 2) As Double
    'Rama basada en el tipo de forma a dibujar
    Select Case tshape
    Case "Círculo"
        Set aCircle = ThisDrawing.ModelSpace. _
                      AddCircle(pltile, trad)
    Case "Polígono"
        angleSegment = 360 / tsides
        currentAngle = 0
        For currentSide = 0 To (tsides - 1)
            angleInRadians = dtr(currentAngle)
            varRet = ThisDrawing.Utility.PolarPoint(pltile, _
                     angleInRadians, trad)
            points((currentSide * 2) + 1) = varRet(0)
            points((currentSide * 2) + 2) = varRet(1)
            currentAngle = currentAngle + angleSegment
        Next currentSide
        Set aPolygon = ThisDrawing.ModelSpace. _
                       AddLightWeightPolyline(points)
        aPolygon.Closed = True
    End Select
End Sub
```

   Esta subrutina utiliza la instrucción `Select Case` para ramificar el control del programa según el tipo de forma que se deba dibujar. La variable `tshape` se utiliza para determinar el tipo de forma.

6. A continuación, vaya a la subrutina `drow`. Encuentre los dos casos en los que aparece la siguiente línea:

```
Set cir = ThisDrawing.ModelSpace.AddCircle(pltile, trad)
```

Cambie estas líneas para que dibujen la forma correspondiente de losetas, como se muestra a continuación:

```
DrawShape (pltile) ' Actualizado
```

¿Comentarios?

Adición de código al cuadro de diálogo

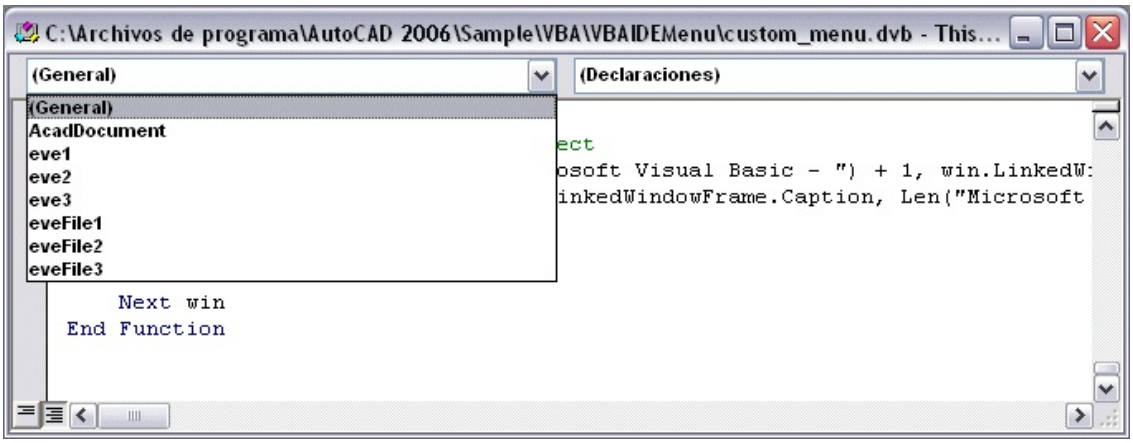
Ahora puede eliminar el código para la creación de losetas circulares e invocar la subrutina `DrawShape` para que dibuje la forma apropiada.

Para añadir gestores de eventos para el cuadro de diálogo

1. Abrir el código para `gpDialog`.
2. Escriba el siguiente código en la parte superior de la ventana:

```
Private Sub gp_poly_Click()
    gp_tsides.Enabled = True
    ThisDrawing.tshape = "Polígono"
End Sub
Private Sub gp_circ_Click()
    gp_tsides.Enabled = False
    ThisDrawing.tshape = "Círculo"
End Sub
```

Observe que las subrutinas `gp_poly_Click()` y `gp_circ_Click()` tienen el mismo nombre que los dos controles de opción añadidos anteriormente, con la adición de `_Click`. Estas subrutinas se ejecutan automáticamente cuando el usuario pulsa en el control respectivo. Observe también que el cuadro Objeto muestra los controles del formulario, ordenados alfabéticamente por la propiedad "Name" (nombre).



3. Sitúe el cursor sobre la línea `Private Sub gp_poly_Click()` y abra el cuadro Procedimiento/Evento.

   Podrá ver una lista de todos los eventos a los que puede responder para el control de opción `gp_poly`. Las dos subrutinas que ha escrito gestionan el evento `Click`. También puede añadir código para responder al evento `DblClick`, que se ejecutará automáticamente cuando el usuario haga doble clic en el control. Puede añadir código para cualquiera de los eventos de la lista. Estos tipos de subrutinas se denominan gestores de eventos.

   Observe el código que ha escrito para estos dos gestores de eventos. El primer gestor de eventos responde al evento `Click` que corresponde al control de opción `gp_poly`. La primera línea de código activa el cuadro de texto para el número de lados. Este cuadro de texto sólo está disponible para polígonos, por lo que no está activado a no ser que seleccione el control Polígono. La siguiente línea de código establece la variable `tshape` como `Polígono`.

   El segundo gestor de eventos responde al evento `Click` para el control de opción `gp_circ` . Este gestor desactiva el cuadro de texto para número de lados y establece la variable `tshape` en `Círculo`.

4. Añada el siguiente gestor de eventos para el botón Aceptar:

```
Private Sub accept_Click()
    If ThisDrawing.tshape = "Polígono" Then
        ThisDrawing.tsides = CInt(gp_tsides.text)
        If (ThisDrawing.tsides < 3#) Or _
                        (ThisDrawing.tsides > 1024#) Then
            MsgBox "Escriba un valor entre 3 y " & _
                   "1024 para el número de lados."
```

```
            Exit Sub
        End If
    End If
    ThisDrawing.trad = CDbl(gp_trad.text)
    ThisDrawing.tspac = CDbl(gp_tspac.text)
    If ThisDrawing.trad < 0# Then
        MsgBox "Escriba un valor positivo para el radio."
        Exit Sub
    End If
    If (ThisDrawing.tspac < 0#) Then
        MsgBox "Escriba un valor positivo para el espaciado."
        Exit Sub
    End If
    GPDialog.Hide
End Sub
```

Este código comprueba si la elección final de la forma ha sido la de polígono. Si es así, el código obtiene el número de lados del control `gp_tsides`. El valor que introduce el usuario se almacena en la propiedad Text. Puesto que se almacena como cadena de texto, la cadena debe convertirse al entero equivalente utilizando la función `CInt`. Una vez obtenido, el gestor de eventos comprueba el rango del valor para asegurar que se encuentra entre 3 y 1024. Si no es así, se muestra un mensaje y se sale del gestor de eventos sin que tenga lugar ningún otro proceso. El resultado es que aparece un mensaje y que el usuario tiene otra oportunidad para cambiar el valor. Después de pulsar de nuevo el botón Aceptar, este gestor de eventos se ejecuta y vuelve a comprobar el valor.

La macro obtiene valores de radio y de espacio, pero estos valores son dobles, no enteros, y se obtienen utilizando la función `CDbl`. Estos valores también se verifican para comprobar que son positivos.

Una vez obtenidos y verificados los valores, la instrucción `gpDialog.Hide` oculta el formulario, devolviendo el control a la subrutina que invocó el formulario por primera vez.

5. Añada el siguiente gestor de eventos para el botón Cancelar:

```
Private Sub cancel_Click()
    Unload Me
    Final
End Sub
```

Este sencillo gestor de eventos descarga el formulario y completa la

macro.

Lo único que todavía no ha hecho es añadir los valores iniciales para el formulario. Hay un evento llamado Initialize que se aplica al formulario. Se ejecuta cuando se carga el formulario por primera vez.

6. Añada el siguiente gestor de eventos para la inicialización de formularios:

```
Private Sub UserForm_Initialize()
    gp_circ.Value = True
    gp_trad.Text = ".2"
    gp_tspac.Text = ".1"
    gp_tsides.Text = "5"
    gp_tsides.Enabled = False
    ThisDrawing.tsides = 5
End Sub
```

Este código establece los valores iniciales del formulario y para la variable `tsides`. La `tsides` debe establecerse en un número positivo mayor que 3, aunque el usuario seleccione un círculo. Para comprender esto, fíjese en la subrutina `DrawShape` que ha escrito anteriormente. Hay una variable llamada `points` que se define utilizando el número de lados del polígono. Tanto si se solicita una forma de polígono como si no, se asigna memoria a la variable. Por este motivo, `tsides` debe estar dentro de un rango válido. El usuario puede cambiar este valor durante la ejecución de la macro.

Ahora puede guardar la macro y ejecutarla desde AutoCAD.

Comparación entre Visual LISP y ActiveX/VBA

La mayoría de las funciones disponibles en la interfaz de Visual LISP se encuentran también en la interfaz de ActiveX y VBA. Este apéndice pretende servir de referencia a los desarrolladores que ya conocen Visual LISP, para que conozcan la correspondencia de las funciones en ActiveX y VBA.

Comparación entre Visual LISP y ActiveX/VBA

En la siguiente tabla se comparan las funciones de Visual LISP con los operadores y las funciones similares de ActiveX®, VBA y Visual Basic 6. Los equivalentes de ActiveX Automation están indicados por "AutoCAD.Application." y los equivalentes de Visual Basic 6 se enumeran como función u operador.

Comparación entre Visual LISP y ActiveX/VBA

Función de AutoLISP	Equivalente de ActiveX, VBA o Visual Basic 6
+ (suma)	+ (operador de suma)
– (resta)	- (operador de resta)
* (multiplicación)	* (operador de multiplicación)
/ (división)	/ (operador de división)
= (es igual que)	= (igual que, operador de comparación)
!= (distinto de)	<> (distinto de, operador de comparación)
< (menor que)	< (menor que, operador de comparación)
<= (menor o igual que)	<= (menor o igual que, operador de comparación)

!= (distinto de)	<> (distinto de, operador de comparación)
> (mayor que)	> (mayor que, operador de comparación)
> (mayor que es igual que)	>= (mayor o igual que, operador de comparación)
~ (No binario)	Operador No
1+ (incremento)	Utilizar + (operador de suma)
1- (decremento)	Utilizar – (operador de resta)
abs	Función Abs
acad_colordlg	*No existe*
acad_helpdlg	Buscar HELP en el índice de Ayuda en pantalla
acad_strlsort	Buscar SORT en el índice de Ayuda en pantalla
action_tile	Utilizar el editor de diálogo
add_list	Utilizar el editor de diálogo
ads	Método AutoCAD.Application.ListADS
alert	Función MsgBox
and	Operador And
angle	Método AutoCAD.Application.ActiveDocument.Utility. AngleFromXAxis
angtof	Método AutoCAD.Application.ActiveDocument.Utility.
angtos	Método AutoCAD.Application.ActiveDocument.Utility.

	Método AngleToString
append	Utilizar las funciones de gestión de matrices
apply	*No existe*
arx	Método AutoCAD.Application.ListARX
arxload	Método AutoCAD.Application.LoadARX
arxunload	Método AutoCAD.Application.UnloadARX
ascii	Función Asc
assoc	*No existe*
atan	Función Atn
atof	Función CDbl
atoi	Función Cint
atom	Buscar IS en el índice de Ayuda en pantalla
atoms-family	*No existe*
autoarxload	*No existe*
autoload	*No existe*
Boole	Utilizar operadores lógicos
boundp	Buscar IS en el índice de Ayuda en pantalla
car/cdr	Utilizar las funciones de gestión de matrices
chr	Función Chr

client_data_tile	Utilizar el editor de diálogo
close	Método AutoCAD.Application.Documents.Close
comando	Método AutoCAD.ActiveDocument.SendCommand
cond	Instrucción Select Case
cons	Usar las funciones de gestión de matrices o el método AutoCAD.Application.collection.Add<entityname>
cos	Función Cos
cvunit	Utilizar las funciones de conversión
defun	Palabras clave Function y End Function
dictadd	Método AutoCAD.Application.ActiveDocument.Dictionaries.Add
dictnext	Método AutoCAD.Application.ActiveDocument.Dictionaries.Item
dictremove	AutoCAD.Application.ActiveDocument.Dictionaries. Método Dictionary.Delete
dictrename	AutoCAD.Application.ActiveDocument.Dictionaries. Método Dictionary.Rename
dictsearch	AutoCAD.Application.ActiveDocument.Dictionaries. Métodos Dictionary.GetName y GetObject
dimx_tile y dimy_tile	Utilizar el editor de diálogo
distance	Método AutoCAD.Application.Utility.GetDistance interactivo.

distof	*No existe*
done_dialog	Utilizar el editor de diálogo
end_image	Utilizar el editor de diálogo
end_list	Utilizar el editor de diálogo
entdel	AutoCAD.Application.ActiveDocument.collection_object. Método Delete
entget	AutoCAD.Application.ActiveDocument.collection_object. property properties
entlast	AutoCAD.Application.ActiveDocument.Modelspace. Item(count-1)
entmake	AutoCAD.Application.ActiveDocument.Modelspace. Método Add<entityname>
entmakex	AutoCAD.Application.ActiveDocument.Modelspace. Método Add<entityname>
entmod	Utilizar cualquiera de las propiedades de lectura y escritura del objeto
entnext	Método AutoCAD.Application.ActiveDocument.collection.Item
entsel	Objeto/métodos/propiedades AutoCAD.Application.ActiveDocument.SelectionSets
entupd	AutoCAD.Application.ActiveDocument.Modelspace.object. Método Update
eq	*No existe*

equal	Operador Eqv
error	Objeto/método/propiedades de error
eval	*No existe*
exit	Método AutoCAD.Application.Quit
exp	Función Exp
expand	*No existe*
expt	^ (operador exponencial)
fill_image	Utilizar el editor de diálogo
findfile	Función Dir
fix	Funciones Fix, Int, Cint
float	Función CDbl
foreach	Instrucción For Each...Next
gc	AutoCAD.Application.ActiveDocument.PurgeAll
gcd	*No existe*
get_attr	Utilizar el editor de diálogo
get_tile	Utilizar el editor de diálogo
getangle	Método AutoCAD.Application.ActiveDocument.Utility.GetAngle
getcfg	Propiedad AutoCAD.Application.Preferences.property
getcname	*No existe*

getcorner	Método AutoCAD.Application.ActiveDocument.Utility.GetCorner
getdist	Método AutoCAD.Application.ActiveDocument.Utility.GetDistance
getenv	Propiedad AutoCAD.Application.Preferences.property
getfiled	Utilizar el cuadro de diálogo de archivo
getint	Método AutoCAD.Application.ActiveDocument.Utility.GetInteger
getkword	Método AutoCAD.Application.ActiveDocument.Utility.GetKeyword
getorient	Método AutoCAD.Application.ActiveDocument.Utility. Método GetOrientation
getpoint	Método AutoCAD.Application.ActiveDocument.Utility.GetPoint
getreal	Método AutoCAD.Application.ActiveDocument.Utility.GetReal
getstring	Método AutoCAD.Application.ActiveDocument.Utility.GetString
getvar	Método AutoCAD.Application.GetVariable
graphscr	AppActivate AutoCAD.Application.Caption
grclear	*Función anticuada*
grdraw	*No existe*
grread	*No existe*

grtext	AutoCAD.Application.ActiveDocument.Utility.Prompt
grvecs	*No existe*
handent	AutoCAD.Application.ActiveDocument.ModelSpace.object. Propiedad Handle
help	Buscar HELP en el índice de Ayuda en pantalla
if	Instrucción If… Then… Else
initget	Método AutoCAD.Application.ActiveDocument.Utility. InitializeUserInput
inters	AutoCAD.Application.ActiveDocument.Modelspace.object. IntersectWith
itoa	Función Str
lambda	*No existe*
last	arrayname(UBound(arrayname))
length	Función UBound
(lista)	Instrucción ReDim
listp	Función IsArray
load_dialog	Utilizar el editor de diálogo
load	AutoLISP no puede utilizarse con Automation
log	Función Log
logand	Función And

logior	Función Or
lsh	Función Imp
mapcar	*No existe*
max	Función Max
mem	*No existe*
member	Utilizar colección
menucmd	Objeto AutoCAD.Application.MenuBar
menugroup	Objeto AutoCAD.Application.MenuGroup
min	Función Min
minusp	Usar sintaxis < 0
mode_tile	Utilizar el editor de diálogo
namedobjdict	Colección AutoCAD.Application.ActiveDocument.
nentsel	AutoCAD.Application.ActiveDocument.SelectionSets. Método SelectionSet.SelectAtPoint
nentselp	AutoCAD.Application.ActiveDocument.SelectionSets. Método SelectionSet.SelectAtPoint
new_dialog	Utilizar el editor de diálogo
not	Utilizar los operadores lógicos
nth	Utilizar sintaxis de objeto(n)
null	Función IsNull

numberp	TypeName (función)
open	Función Open
or	Utilizar los operadores lógicos
osnap	*No existe* (Se puede utilizar el método SetVariable para controlar la variable de sistema OSMODE.)
polar	Método AutoCAD.Application.ActiveDocument.Utility.PolarPoint
prin1	AutoCAD.Application.ActiveDocument.Utility.Prompt
princ	AutoCAD.Application.ActiveDocument.Utility.Prompt
print	AutoCAD.Application.ActiveDocument.Utility.Prompt
progn	*No existe*
prompt	AutoCAD.Application.ActiveDocument.Utility.Prompt
quit	Método AutoCAD.Application.Quit
quote	*No existe*
read	*No existe*
read-char	Función Input
read-line	Función Line Input
redraw	AutoCAD.Application.ActiveDocument.Modelspace.object. Método Update
regapp	AutoCAD.Application.ActiveDocument. Método RegisteredApplications.Add

rem	Función Mod
repeat	For… Each, While,
reverse	*No existe*
rtos	Método AutoCAD.Application.ActiveDocument.Utility.RealToString
set	Función Set
set_tile	Utilizar el editor de diálogo
setcfg	Propiedad AutoCAD.Application.Preferences.property
setfunhelp	*No existe*
setq	Función Set
setvar	Método AutoCAD.Application.SetVariable
sin	Función Sin
setview	AutoCAD.Application.ActiveDocument.Viewports.Viewport Método SetView
slide_image	Utilizar el editor de diálogo
snvalid	*No existe*
sqrt	Función Sqr
ssadd	Método AutoCAD.Application.ActiveDocument.SelectionSets.Add
ssdel	AutoCAD.Application.ActiveDocument.SelectionSets. Método SelectionSet.Delete

ssget	AutoCAD.Application.ActiveDocument.SelectionSets. Método SelectionSet.SelectOnScreen
ssgetfirst	*No existe*
sslength	AutoCAD.Application.ActiveDocument.SelectionSets. Método SelectionSet.Count
ssmemb	Comparar ID de objeto con los miembros de SelectionSet
ssname	AutoCAD.Application.ActiveDocument.SelectionSets. Propiedad SelectionSet.Name
ssnamex	*No existe*
sssetfirst	AutoCAD.Application.ActiveDocument.PickfirstSelectionSe
startapp	Función Shell
start_dialog	Utilizar el editor de diálogo
start_image	Utilizar el editor de diálogo
start_list	Utilizar el editor de diálogo
strcase	Función StrConv
strcat	Operador &
strlen	Función Len
subst	*No existe*
substr	Función Mid
tablero	*No existe*

tblnext	AutoCAD.Application.ActiveDocument.collection_object. Método Item
tblobjname	AutoCAD.Application.ActiveDocument.collection_object. Método Name
tblsearch	AutoCAD.Application.ActiveDocument.collection_object. Método Name
term_dialog	Utilizar el editor de diálogo
terpri	*No existe*
textbox	AutoCAD.Application.ActiveDocument.space.object. Método GetBoundingBox
textpage	*No existe*
textscr	*No existe*
trace	*No existe*
trans	Método AutoCAD.Application.ActiveDocument.Utility. Método TranslateCoordinates
type	TypeName (función)
unload_dialog	Utilizar el editor de diálogo
untrace	*No existe*
vector_image	Utilizar el editor de diálogo
ver	Propiedad AutoCAD.Application.Version
ventanas	Colección AutoCAD.Application.ActiveDocument.

wcmatch	Operador Like
while	While… Wend
write-char	Función Print
write-line	Función Print
xdroom	*No existe*
xdsize	*No existe*
zerop	Usar = 0 sintaxis

¿Comentarios?
