Windows Dialogs

This extension package provides a number of functions to show many of the standard windows dialogs, like alert boxes, file requesters, etc., with more functionality than the functions that are default available in Game Maker.

To use this package in a game, select the item **Select Extension Packages** in the **Resources** menu. In the form that appears click on the **GM Windows Dialogs** item in the right list and press the button to move it to the left list. That is all.

Below you find a list of all functions that are available in this extension package. Note that all functions start with the prefix `wd_`. 
Message Dialogs

These dialogs are meant to give a message to the user. They can have an icon to show the type of message and multiple buttons such that the user can choose an option.

- `wd_message_simple(str)` This function shows a simple message dialog with the string as the message.
- `wd_message_set_text(str)` This function sets the text for the more extended message boxes in the next two functions.
- `wd_message_show(kind, but, but2, but3)` This function shows the message with the text provided with the previous function. `kind` indicated the kind of box (which is shown by the icon) as follows:

  - `wd_mk_none` = No icon
  - `wd_mk_warning` = Warning icon
  - `wd_mk_error` = Error icon
  - `wd_mk_information` = Information icon
  - `wd_mk_confirmation` = Confirmation icon

There can be up to three buttons. These are indicated by the arguments `but1`, `but2` and `but3`. Their values can be:

  - `wd_mb_none` = No button (used when less than three buttons are required)
  - `wd_mb_ok` = Ok button
  - `wd_mb_cancel` = Cancel button
  - `wd_mb_yes` = Yes button
  - `wd_mb_no` = No button
  - `wd_mb_abort` = Abort button
  - `wd_mb_retry` = Retry button
  - `wd_mb_ignore` = Ignore button

(The order will be determined by the system.) The function returns the button pressed.

- `wd_message_show_pos(kind, but, but2, but3, x, y)` This function works
the same as the previous function, except that the message is not place in the screen center but at position x,y on the screen.
Input Box Dialogs

These dialogs ask the user to input a string.

wd_input_box(caption, prompt, default) This function shows an input box in which the user can input a string. caption is the caption of the input box, prompt is the prompt in front of the value, and default is the default string shown. When the user presses the Cancel button the default value is returned.

wd_input_query(caption, prompt, default) This function also shows an input box in which the user can input a string. The function returns whether the user pressed the OK button. The value can then be retrieved with the next function.

wd_input_query_get_text() This function returns the text typed by the user in the last call to wd_input_query.
This dialog asks the user to pick a color.

- `wd_color_dialog_show()` This function shows the color dialog and returns whether the user pressed **Ok**.
- `wd_color_dialog_get_color()` This function returns the color selected by the user.
- `wd_color_dialog_set_color(col)` Sets the default color for the color dialog.
File Dialogs

These dialogs ask the user to select a file or a picture for opening or saving.

\[\text{wd_open_dialog_show()}\] This function shows the open file dialog and returns whether the user pressed **Open** (or **Ok**).
\[\text{wd_open_dialog_get_file()}\] This function returns the selected file.
\[\text{wd_open_dialog_set_file(fname)}\] This function sets the default file name to be used.
\[\text{wd_open_dialog_set_folder(folder)}\] This function sets the initial folder.
\[\text{wd_open_dialog_set_filter(filter)}\] This function sets the filter. The filter must have the form \text{Name1|Mask1|Name2|Mask2|...}. For example, it could be \text{Text Files|*.txt|All Files|*.*}.
\[\text{wd_open_dialog_set_extension(extension)}\] This function sets the default extension to be used when the user does not specify an extension.
\[\text{wd_open_dialog_set_caption(caption)}\] This function sets the text shown in the caption of the dialog.
\[\text{wd_open_dialog_set_option(option,value)}\] With this function different options can be set. Each option has an index and the value should be true or false. The following options exist:

\[\text{wd_fo_overwriteprompt} = \text{Warn the user when overwriting a file (only for save dialogs)}\]
\[\text{wd_fo_filemustexist} = \text{The file must exist}\]
\[\text{wd_fo_pathmustexist} = \text{The path must exist}\]
\[\text{wd_fo_createprompt} = \text{When a non-existing file is selected the user is asked whether it must be created}\]
\[\text{wd_fo_nochangedir} = \text{After the call the path is reset to the original path}\]
\[\text{wd_fo_noreadonlyreturn} = \text{No files are returned that are read-only}\]
\[\text{wd_fo_oldstyledialog} = \text{The old style dialog is shown}\]
\[\text{wd_fo_dontaddtorecent} = \text{The file is not added to the recent files list}\]
\[\text{wd_fo_forceshowhidden} = \text{Hidden files are also shown}\]

Multiple options can be set with multiple calls.
wd_save_dialog_show()
wd_save_dialog_get_file()
wd_save_dialog_set_file(fname)
wd_save_dialog_set_folder(folder)
wd_save_dialog_set_filter(filter)
wd_save_dialog_set_extension(extension)
wd_save_dialog_set_caption(caption)
wd_save_dialog_set_option(option,value)
These are the same functions as above but this time showing the save file dialog.

wd_open_picture_show()
wd_open_picture_get_file()
wd_open_picture_set_file(fname)
wd_open_picture_set_folder(folder)
wd_open_picture_set_filter(filter)
wd_open_picture_set_extension(extension)
wd_open_picture_set_caption(caption)
wd_open_picture_set_option(option,value)
These are the same functions as above but this time showing the open picture dialog.

wd_save_picture_show()
wd_save_picture_get_file()
wd_save_picture_set_file(fname)
wd_save_picture_set_folder(folder)
wd_save_picture_set_filter(filter)
wd_save_picture_set_extension(extension)
wd_save_picture_set_caption(caption)
wd_save_picture_set_option(option,value)
These are the same functions as above but this time showing the save picture dialog.
Find and Replace Dialogs

These dialogs ask the user for a string to find and, when using replace dialogs, for a replace string. Of course the game itself must take care of the appropriate action. These dialogs stay visible until the user closes them or they are closed using the hide functions. To find out whether the user presses a find or replace key, you must regularly, e.g. in a step event, check whether a button was pressed using the functions `wd_find_dialog_find()` and similar functions for the replace dialog (see below for more information).

- `wd_find_dialog_show()` This function shows the find dialog.
- `wd_find_dialog_show_pos(x,y)` This function shows the find dialog at the indicated position.
- `wd_find_dialog_hide()` This function hides the find dialog.
- `wd_find_dialog_find()` This function returns whether, since the last call, the user pressed the Find Next button. Best check this in a step event. Note that when the user presses the button only the next call of this function returns true.
- `wd_find_dialog_get_tex()` This function returns the find text.
- `wd_find_dialog_set_tex()` This function sets the find text.
- `wd_find_dialog_set_option(option,value)` With this function different options can be set. Each option has an index and the value should be true or false. The following options exist:
  - `wd_fr_hidematchcase` = hide the match case check box
  - `wd_fr_hidewholeword` = hides the whole word check box
  - `wd_fr_hideupdown` = hides the Up or Down selection box
  - `wd_fr_matchcase` = whether match case is selected
  - `wd_fr_wholeword` = whether whole word is selected
  - `wd_fr_down` = whether the Down direction is selected (rather than Up)

- `wd_find_dialog_get_option(option)` With this function you can check the value of the different options. You can use the same options as indicated above. So for example you can use this to check whether only a whole word search must be performed.
wd_replace_dialog_show()  This function shows the replace dialog.
wd_replace_dialog_show_pos(x,y)  This function shows the replace dialog at the indicated position.
wd_replace_dialog_hide()  This function hides the replace dialog.
wd_replace_dialog_find()  This function returns whether, since the last call, the user pressed the Find Next button. Best check this in a step event. Note that when the user presses the button only the next call of this function returns true.
wd_replace_dialog_replace()  This function returns whether, since the last call, the user pressed the Replace or Replace All button. Best check this in a step event.
wd_replace_dialog_get_tex()  This function returns the find text.
wd_replace_dialog_set_tex()  This function sets the find text.
wd_replace_dialog_get_replace()  This function returns the replace text.
wd_replace_dialog_set_replace()  This function sets the replace text.
wd_replace_dialog_set_option(option,value)  With this function different options can be set. Each option has an index and the value should be true or false. The possible options are the same as for the find dialog. wd_find_dialog_get_option(option)  With this function you can check the value of the different options. You can use the same options as indicated above except for the following additional ones.

    wd_fr_replace = the user selected to replace only one occurrence
    wd_fr_replaceall = the user selected to replace all occurrences

When the user clicked on a replace button you should check these options to see what replace action he wants.
Font Dialogs

This dialog asks the user to choose a font.

wd_font_dialog_show() This function shows the font dialog and returns whether the user pressed Ok.
wd_font_dialog_get_color() This function returns the font color selected by the user.
wd_font_dialog_set_color(col) Sets the default color for the font.
wd_font_dialog_get_size() This function returns the font size selected by the user.
wd_font_dialog_set_size(size) Sets the default size for the font.
wd_font_dialog_get_name() This function returns the font name selected by the user.
wd_font_dialog_set_name(name) Sets the default name for the font.
wd_font_dialog_set_style(style) With this function the font style can be set. The following styles exist:

wd_fs_normal = normal style
wd_fs_bold = bold style
wd_fs_italic = italic style
wd_fs_underline = underline style
wd_fs_strikeout = strikeout style

If you want a combination of these style, simply add up the values.
wd_font_dialog_get_style() This function returns the current font style.