



# File List

Here is a list of all documented files with brief descriptions:

<b>CCamera.h</b> <a href="#">[code]</a>	
<b>CColor.h</b> <a href="#">[code]</a>	
<b>CIntersectInfo.h</b> <a href="#">[code]</a>	
<b>CNode.h</b> <a href="#">[code]</a>	
<b>CObject.h</b> <a href="#">[code]</a>	
<b>CRay.h</b> <a href="#">[code]</a>	
<b>CScene.h</b> <a href="#">[code]</a>	
<b>CSphere.h</b> <a href="#">[code]</a>	
<b>CTargetCamera.h</b> <a href="#">[code]</a>	
<b>CVector3D.h</b> <a href="#">[code]</a>	
<b>RayTracerChap1.h</b> <a href="#">[code]</a>	
<b>UtilityLib.h</b> <a href="#">[code]</a>	



# Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

<b>CCamera</b>	<i>Classe de base Camera</i>
<b>CColor</b>	<i>Cette classe represente une couleur RGB ainsi que des operateurs sur les couleurs</i>
<b>CIntersectInfo</b>	<i>Cette classe contient des informations sur une intersection</i>
<b>CNode</b>	<i>La class node est située dessus de toutes les autres classes "objets" du raytracer. Une camera est une node, une lumière est une node, un mesh est une node, une primitive de base est une node ..</i>
<b>CObject</b>	<i>Cette classe est la classe de base pour toutes les primitives implementées dans le raytracer : spheres, plans, cônes ..</i>
<b>CRay</b>	<i>Cette classe définit l'élément de base d'un raytracer : un rayon</i>
<b>CScene</b>	<i>Cette classe contient toutes les informations sur une scène. Elle contient les éléments de la scène (caméras, lumières, objets ...)</i>
<b>CSphere</b>	<i>Cette classe définit la primitive de base "Sphere" du Raytracer</i>
<b>CTargetCamera</b>	<i>Une Target Camera (camera cible), est une camera qui possède une position et un point observé</i>
<b>CVector3D</b>	<i>Cette classe contient une representation d'un vecteur 3D (x, y, z), ainsi que des opérations sur les vecteurs</i>

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

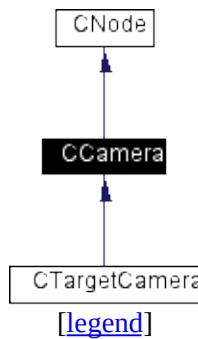
---

# CCamera Class Reference

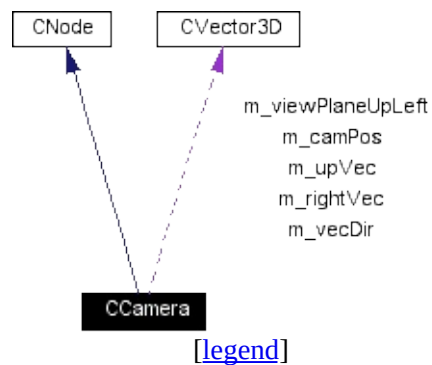
Classe de base Camera. [More...](#)

```
#include <CCamera.h>
```

Inheritance diagram for CCamera:



Collaboration diagram for CCamera:



[List of all members.](#)

## Public Methods

```
CCamera ()  
CCamera (CVector3D &vCamPos,  
CVector3D &vUpVector)  
virtual ~CCamera ()  
CVector3D CalcDirVec (float x, float y, int xRes, int  
yRes)  
__forceinline const CVector3D & GetPosition ()  
bool IntersectsNode (const CRay &ray,  
CIntersectInfo *intersectInfo=NULL)
```

## Protected Attributes

float **m\_viewplaneDist**

float **m\_viewplaneWidth**

float **m\_viewplaneHeight**

**CVector3D** **m\_camPos**

**CVector3D** **m\_vecDir**

**CVector3D** **m\_upVec**

**CVector3D** **m\_rightVec**

**CVector3D** **m\_viewPlaneUpLeft**

---



# Detailed Description

Classe de base Camera.

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**

06/02/2002

---

## Member Function Documentation

```
bool CCamera::IntersectsNode ( const CRay & ray,  
                               CIntersectInfo * intersectInfo = NULL  
                             ) [inline, virtual]
```

Une camera est une Node, elle possède donc également une classe d'intersection. Cependant on se contenter de retourner faux. Mais grâce à cette approche, il est également possible d'afficher une camera, par exemple en dessinant une sphere bleue pour les cameras ;)

Implements [CNode](#).

---

The documentation for this class was generated from the following files:

- [CCamera.h](#)
- [CCamera.cpp](#)

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CColor Class Reference

Cette classe represente une couleur RGB ainsi que des operateurs sur les couleurs. [More...](#)

```
#include <CColor.h>
```

[List of all members.](#)

## Public Methods

```
    CColor ()  
    CColor (float r, float g, float b)  
    virtual ~CColor ()  
    __forceinline void NormalizeColor ()
```

## **Public Attributes**

float **m\_r**

float **m\_g**

float **m\_b**

## Friends

\_\_forceinline friend CColor **operator** \* (const CColor &c1, const CColor &c2)

\_\_forceinline friend CColor **operator** \* (const CColor &c1, const float  
multiple)

\_\_forceinline friend CColor **operator**+ (const CColor &c1, const CColor &c2)

\_\_forceinline friend void **operator**+= (CColor &c1, const CColor &c2)

\_\_forceinline friend CColor **operator**/ (const CColor &c1, float multiple)

\_\_forceinline friend void **operator**/= (CColor &c1, float multiple)

---

## Detailed Description

Cette classe represente une couleur RGB ainsi que des operateurs sur les couleurs.

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**

06/02/2002

---

The documentation for this class was generated from the following file:

- [CColor.h](#)



[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

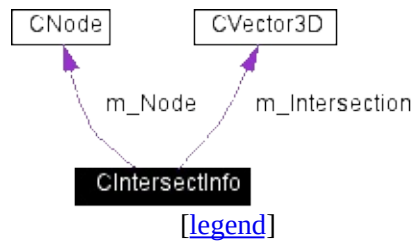
---

# CIntersectInfo Class Reference

Cette classe contient des informations sur une intersection. [More...](#)

```
#include <CIntersectInfo.h>
```

Collaboration diagram for CIntersectInfo:



[List of all members.](#)

## Public Methods

**CIntersectInfo ()**

virtual **~CIntersectInfo ()**

## Public Attributes

**CVector3D** m\_Intersection

**CNode** \* m\_Node

---

## Detailed Description

Cette classe contient des informations sur une intersection.

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**


06/02/2002

---

The documentation for this class was generated from the following file:

- [CIntersectInfo.h](#)

---

Generated on Thu Apr 4 23:14:18 2002 by  1.2.15

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

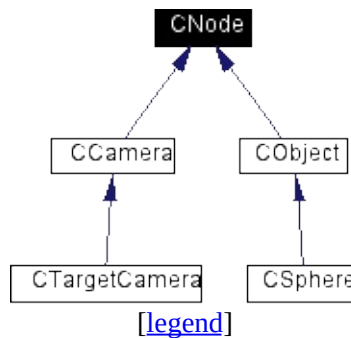
---

# CNode Class Reference

La class node est située dessus de toutes les autres classes "objets" du raytracer. Une camera est une node, une lumière est une node, un mesh est une node, une primitive de base est une node ... [More...](#)

```
#include <CNode.h>
```

Inheritance diagram for CNode:



[List of all members.](#)

## Public Methods

```
    CNode ()  
    CNode (int nodeType, char *name)  
    virtual ~CNode ()  
    virtual bool IntersectsNode (const CRay &ray, CIntersectInfo  
        *intersectInfo=NULL)=0  
    __forceinline int GetNodeType () const  
    __forceinline const char * GetName () const  
    __forceinline CNode * GetInstance () const  
    __forceinline void SetNodeType (int nodeType)  
    __forceinline void SetName (char *name)
```



## Protected Attributes

```
    int m_NodeType  
    char * m_Name
```

---

## Detailed Description

La class node est située dessus de toutes les autres classes "objets" du raytracer. Une camera est une node, une lumière est une node, un mesh est une node, une primitive de base est une node ...

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**

06/02/2002

---

The documentation for this class was generated from the following file:

- [CNode.h](#)

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

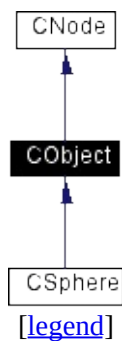
---

# CObject Class Reference

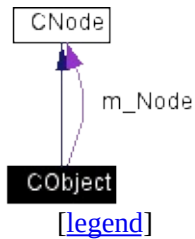
Cette classe est la classe de base pour toutes les primitives implémentées dans le raytracer : spheres, plans, cônes ... [More...](#)

```
#include <CObject.h>
```

Inheritance diagram for CObject:



Collaboration diagram for CObject:



[List of all members.](#)

## Public Methods

```
    CObject ()  
    virtual ~CObject ()  
    __forceinline SetNode (CNode *node)  
    __forceinline CNode * GetNode () const  
    virtual bool IntersectsNode (const CRay &ray, CIntersectInfo  
        *intersectInfo=NULL)=0
```

## Protected Attributes

**CNode** \* m\_Node

---

## Detailed Description

Cette classe est la classe de base pour toutes les primitives implementées dans le raytracer : spheres, plans, cônes ...

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**

06/02/2002

---

The documentation for this class was generated from the following file:

- [CObject.h](#)

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

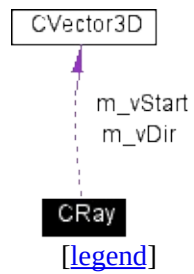


# CRay Class Reference

Cette classe définit l'élément de base d'un raytracer : un rayon. [More...](#)

```
#include <CRay.h>
```

Collaboration diagram for CRay:



[List of all members.](#)

## **Public Methods**

**CRay ()**  
virtual **~CRay ()**

## Public Attributes

**CVector3D** m\_vStart

**CVector3D** m\_vDir

float m\_t

---

## Detailed Description

Cette classe définit l'élément de base d'un raytracer : un rayon.

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**


06/02/2002

---

The documentation for this class was generated from the following file:

- [CRay.h](#)

---

Generated on Thu Apr 4 23:14:19 2002 by  1.2.15

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

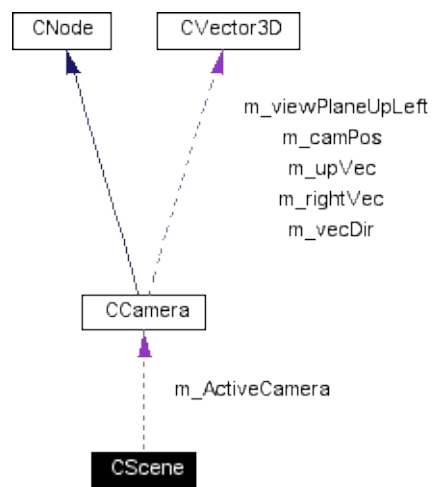
---

# CScene Class Reference

Cette classe contient toutes les informations sur une scène. Elle contient les éléments de la scène (caméras, lumières, objets ...). [More...](#)

```
#include <CScene.h>
```

Collaboration diagram for CScene:



[\[legend\]](#)

[List of all members.](#)

## Public Methods

```
        CScene ()  
        virtual ~CScene ()  
        __forceinline int GetNumNodes () const  
        __forceinline CNode * GetNode (const unsigned int i) const  
__forceinline CCamera * GetActiveCamera () const  
        __forceinline void SetActiveCamera (const unsigned int i)  
        __forceinline void SetActiveCamera (CCamera *cam)  
        __forceinline bool AddObject (CObject *object)  
        __forceinline bool AddCamera (CCamera *camera)  
        __forceinline bool AddNode (CNode *node)
```

---

## Detailed Description

Cette classe contient toutes les informations sur une scène. Elle contient les éléments de la scène (caméras, lumières, objets ...).

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**

06/02/2002

---

The documentation for this class was generated from the following file:

- [CScene.h](#)



[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

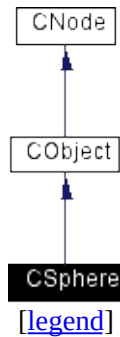
---

# CSphere Class Reference

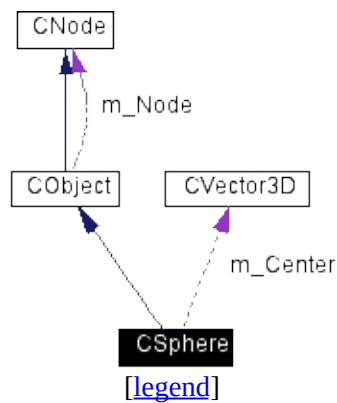
Cette classe définit la primitive de base "Sphere" du Raytracer. [More...](#)

```
#include <CSphere.h>
```

Inheritance diagram for CSphere:



Collaboration diagram for CSphere:



[List of all members.](#)

## Public Methods

**CSphere** ()

**CSphere** (**CVector3D** center, float radius)

virtual **~CSphere** ()

bool **IntersectsNode** (const **CRay** &ray, **CIntersectInfo**  
\*intersectInfo=NULL)

---

## Detailed Description

Cette classe définit la primitive de base "Sphere" du Raytracer.

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**


06/02/2002

---

The documentation for this class was generated from the following files:

- [CSphere.h](#)
- CSphere.cpp

---

Generated on Thu Apr 4 23:14:19 2002 by  1.2.15

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

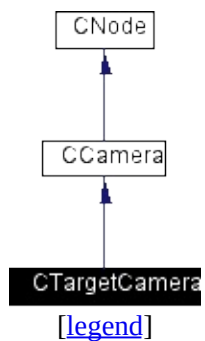
---

# CTargetCamera Class Reference

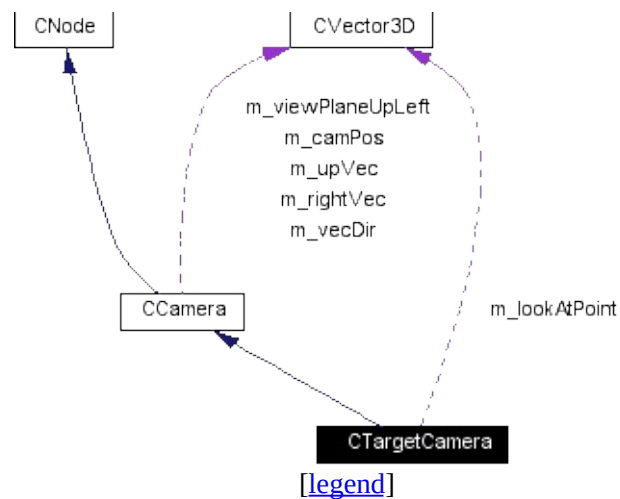
Une Target Camera (camera cible), est une camera qui possède une position et un point observé. [More...](#)

```
#include <CTargetCamera.h>
```

Inheritance diagram for CTargetCamera:



Collaboration diagram for CTargetCamera:



[List of all members.](#)

## Public Methods

**CTargetCamera** ()

**CTargetCamera** (**CVector3D** &camPos, **CVector3D** &lookAtPoint,  
**CVector3D** &upVector)

virtual **~CTargetCamera** ()

---

## Detailed Description

Une Target Camera (camera cible), est une camera qui possède une position et un point observé.

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**

06/02/2002

---

The documentation for this class was generated from the following files:

- [CTargetCamera.h](#)
- CTargetCamera.cpp



[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CVector3D Class Reference

Cette classe contient une représentation d'un vecteur 3D (x, y, z), ainsi que des opérations sur les vecteurs. [More...](#)

```
#include <CVector3D.h>
```

[List of all members.](#)

## Public Methods

```
CVector3D ()  
CVector3D (float _x, float _y, float _z)  
virtual ~CVector3D ()  
__forceinline float DotProduct (const CVector3D &v)  
__forceinline float GetMagnitude () const  
__forceinline void Normalize ()  
__forceinline float GetSquareLength () const  
__forceinline CVector3D Reflect (const CVector3D &n)
```

## **Public Attributes**

float **x**

float **y**

float **z**

## Friends

\_\_forceinline friend CVector3D **operator+** (const CVector3D &v1, const CVector3D &v2)

\_\_forceinline friend void **operator+=** (CVector3D &v1, const CVector3D &v2)

\_\_forceinline friend CVector3D **operator-** (const CVector3D &v1, const CVector3D &v2)

\_\_forceinline friend CVector3D **operator \*** (const CVector3D &v1, const CVector3D &v2)

\_\_forceinline friend CVector3D **operator \*** (const CVector3D &v1, const float multiple)

\_\_forceinline friend CVector3D **operator/** (const CVector3D &v1, const float multiple)

\_\_forceinline friend void **operator/=** (CVector3D &v1, float multiple)

\_\_forceinline friend CVector3D **operator-** (const CVector3D &v)

---

## Detailed Description

Cette classe contient une representation d'un vecteur 3D (x, y, z), ainsi que des opérations sur les vecteurs.

**Author:**

Benoît Lemaire (aka DaRkWoLf)

**Date:**

06/02/2002

---

The documentation for this class was generated from the following file:

- [CVector3D.h](#)



# Class Hierarchy

[Go to the graphical class hierarchy](#)

This inheritance list is sorted roughly, but not completely, alphabetically:

- **CColor**
- **CIntersectInfo**
- **CNode**
  - **CCamera**
    - **CTargetCamera**
  - **CObject**
    - **CSphere**
- **CRay**
- **CScene**
- **CVector3D**



[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# Compound Members

Here is a list of all documented class members with links to the classes they belong to:

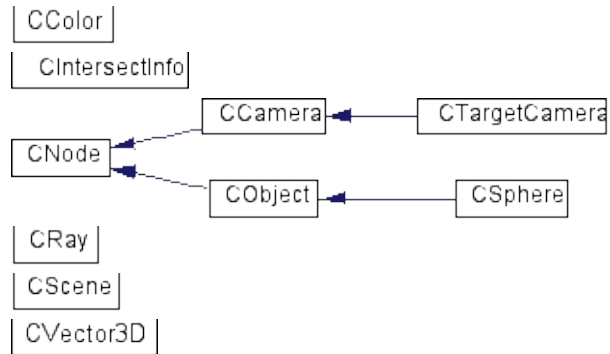
- [IntersectsNode\(\)](#) : [CCamera](#)

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# Graphical Class Hierarchy

[Go to the textual class hierarchy](#)



[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CCamera.h

```
00001
00007 #ifndef __CCAMERA_H__
00008 #define __CCAMERA_H__
00009
00010 #include "CNode.h"
00011 #include "UtilityLib.h"
00012
00013 class CCamera : public CNode
00014 {
00015     protected:
00016         float m_viewplaneDist;
00017         float m_viewplaneWidth, m_viewplaneHeight;
00018         CVector3D m_camPos, m_vecDir, m_upVec, m_rightVec, m_v
00019
00020     public:
00021         CCamera() {}
00022         CCamera(CVector3D& vCamPos, CVector3D& vUpVect
00023     virtual ~CCamera() {}
00024
00025         CVector3D CalcDirVec(float x, float y, int xRe
00026
00027         // get
00028         __forceinline const CVector3D& GetPosition() {
00029
00030         // node raytrace related
00031         bool IntersectsNode(const CRay& ray, CIntersec
00032     };
00033
00034
00035 };
00036
00037
00038 #endif /* #ifndef __CCAMERA_H__ */
```

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CColor.h

```
00001
00007 #ifndef __CCOLOR_H__
00008 #define __CCOLOR_H__
00009
00010 class CColor
00011 {
00012     public:
00013         float m_r, m_g, m_b;    // Les trois composant
00014
00015         // constructeurs et destructeur
00016         CColor() : m_r(0), m_g(0), m_b(0) {};
00017         CColor(float r, float g, float b) : m_r(r), m_
00018         virtual ~CColor() {};
00019
00020         // operateurs
00021         __forceinline friend CColor operator* (const C
00022         { return CColor(c1.m_r*c2.m_r, c1.m_g*c2.m_g,
00023
00024         __forceinline friend CColor operator* (const C
00025         { return CColor(c1.m_r*multiple, c1.m_g*multip
00026
00027         __forceinline friend CColor operator+ (const C
00028         { return CColor(c1.m_r+c2.m_r, c1.m_g+c2.m_g,
00029
00030         __forceinline friend void operator+= (CColor&
00031         { c1.m_r += c2.m_r; c1.m_g += c2.m_g; c1.m_b +
00032
00033         __forceinline friend CColor operator/ (const C
00034         { return CColor(c1.m_r/multiple, c1.m_g/multip
00035
00036         __forceinline friend void operator/= (CColor&
00037         { c1.m_r /= multiple; c1.m_g /= multiple; c1.m
00038
00039         // main methods
00040         __forceinline void NormalizeColor()    { if (
00041     };
00042
00043 #endif /* #ifndef __CCOLOR_H__ */
```

---





[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CIntersectInfo.h

```
00001
00007 #ifndef __CINTERSECTINFO_H__
00008 #define __CINTERSECTINFO_H__
00009
00010 #include "UtilityLib.h"
00011 #include "stdlib.h"
00012
00013 // forward declarations
00014 class CScene;
00015 class CNode;
00016
00017 // information sur l'intersection
00018 class CIntersectInfo
00019 {
00020     public:
00021         CVector3D          m_Intersection; // Pos
00022         CNode              *m_Node;
00023
00024         // constructeurs et destructeur
00025         CIntersectInfo() : m_Node(NULL) {}
00026         virtual ~CIntersectInfo() {}
00027 };
00028
00029 #endif /* #ifndef __CINTERSECTINFO_H__ */
```

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CNode.h

```
00001
00008 #ifndef __CNODE_H__
00009 #define __CNODE_H__
00010
00011 #include "UtilityLib.h"
00012 #include "CIntersectInfo.h"
00013 #include "CRay.h"
00014
00015
00016 class CNode
00017 {
00018     protected:
00019         int m_NodeType;
00020         char* m_Name;
00021
00022     public:
00023         // constructeurs et destructeur
00024         CNode() { m_Name = NULL; };
00025         CNode(int nodeType, char *name) { m_NodeType=n
00026         virtual ~CNode() { SAFERELEASE(m_Name); }
00027
00028         // raytrace related
00029         virtual bool IntersectsNode(const CRay& ray, C
00030
00031         // get
00032         __forceinline int GetNode() const
00033         __forceinline const char* GetName() const
00034         __forceinline CNode* GetNodeInstance() const
00035
00036         // set
00037         __forceinline void SetNodeType(int nodeTy
00038         __forceinline void SetName(char* name)
00039 };
00040
00041 #endif /* #ifndef __CNODE_H__ */
```

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CObject.h

```
00001
00008 #ifndef __COBJECT_H__
00009 #define __COBJECT_H__
00010
00011 #include "CNode.h"
00012
00013 class CObject : public CNode
00014 {
00015     protected:
00016         CNode          *m_Node;
00017
00018     public:
00019         CObject() { m_Node = GetNodeInstance(); }
00020         virtual ~CObject() { }
00021
00022         // set
00023         __forceinline SetNode(CNode* node) { m_Node =
00024
00025         // get
00026         __forceinline CNode* GetNode() const
00027
00028         // [hérité de CNode]
00029         virtual bool IntersectsNode(const CRay& ray, C
00030 };
00031
00032 #endif /* #ifndef __COBJECT_H__ */
```

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---



# CRay.h

```
00001
00007 #ifndef __CRAY_H__
00008 #define __CRAY_H__
00009
00010 #include "UtilityLib.h"
00011
00012 class CRay
00013 {
00014     public:
00015         CVector3D      m_vStart, m_vDir;
00016         float          m_t;
00017
00018         // constructeurs et destructeur
00019         CRay() {}
00020         virtual ~CRay() {}
00021 };
00022
00023 #endif /* #ifndef __CRAY_H__ */
```

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CScene.h

```
00001
00008 #ifndef __CSCENE_H__
00009 #define __CSCENE_H__
00010
00011 #include "CCamera.h"
00012 #include "CObject.h"
00013 #include "CNode.h"
00014 #include "assert.h"
00015 #include <vector>
00016
00017 using namespace std;
00018
00019 class CScene
00020 {
00021     private:
00022         int
00023         vector<CNode*>                m_Nodes;
00024         vector<CCamera*>              m_Cameras;
00025         CCamera                       *m_Act
00026
00027     public:
00028         // constructeurs et destructeurs
00029         CScene() : m_NumNodes(0), m_ActiveCamera(NULL)
00030         virtual ~CScene()
00031         { }
00032
00033         // get
00034         __forceinline int                GetNum
00035         __forceinline CNode*            GetNode(const
00036         __forceinline CCamera*         GetActiveCamer
00037
00038         // set
00039         __forceinline void              SetActiveCamer
00040         __forceinline void              SetActiveCamer
00041
00042         // methodes
00043         __forceinline bool              AddObject(CObject* obj
00044         __forceinline bool              AddCamera(CCamera* cam
00045         __forceinline bool              AddNode(CNode* node) {
00046     };
00047
00048 #endif /* #ifndef __CSCENE_H__ */
```



[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CSphere.h

```
00001
00007 #ifndef __CSPHERE_H__
00008 #define __CSPHERE_H__
00009
00010 #include "CObject.h"
00011
00012 class CSphere : public CObject
00013 {
00014     private:
00015         CVector3D      m_Center; // Le centre de la s
00016         float          m_Radius; // Le rayon de la sp
00017
00018     public:
00019         // constructeurs et destructeur
00020         CSphere() {}
00021         CSphere(CVector3D center, float radius) : m_Ce
00022         virtual ~CSphere() {}
00023
00024         // [hérité de CObject]
00025         bool IntersectsNode(const CRay& ray, CIntersec
00026 };
00027
00028 #endif /* #ifndef __CSPHERE_H__ */
```

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CTargetCamera.h

```
00001
00007 #ifndef __CTARGETCAM_H__
00008 #define __CTARGETCAM_H__
00009
00010 #include "CCamera.h"
00011
00012 class CTargetCamera : public CCamera
00013 {
00014     private:
00015         CVector3D      m_lookAtPoint; // Le point ob
00016
00017     public:
00018         // constructeurs et destructeur
00019         CTargetCamera() {}
00020         CTargetCamera(CVector3D& camPos, CVector3D& lo
00021     virtual ~CTargetCamera() {}
00022 };
00023
00024 #endif /* #ifndef __CTARGETCAM_H__ */
```



[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)


---

# CVector3D.h

```
00001
00007 #ifndef __CVECTOR3D_H__
00008 #define __CVECTOR3D_H__
00009
00010 #include "math.h"
00011 #include "stdio.h"
00012
00013 class CVector3D
00014 {
00015     public:
00016         float x, y, z;
00017
00018         // constructeurs et destructeur
00019         CVector3D() : x(0.0f), y(0.0f), z(0.0f) {}
00020         CVector3D(float _x, float _y, float _z) : x(_x
00021         virtual ~CVector3D() {}
00022
00023         // operateurs
00024         __forceinline friend CVector3D operator+ (const
00025         { return CVector3D(v1.x+v2.x, v1.y+v2.y, v1.z+
00026
00027         __forceinline friend void operator+= (CVector3
00028         { v1.x += v2.x; v1.y += v2.y; v1.z += v2.z; }
00029
00030         __forceinline friend CVector3D operator- (const
00031         { return CVector3D(v1.x-v2.x, v1.y-v2.y, v1.z-
00032
00033         __forceinline friend CVector3D operator* (const
00034         { return CVector3D(      ( v1.y * v2.z ) - ( v1
00035
00036         __forceinline friend CVector3D operator* (const
00037         { return CVector3D(v1.x*multiple, v1.y*multipl
00038
00039         __forceinline friend CVector3D operator/ (const
00040         { return CVector3D(v1.x/multiple, v1.y/multipl
00041
00042         __forceinline friend void operator/= (CVector3
00043         { v1.x /= multiple; v1.y /= multiple; v1.z /=
00044
00045         __forceinline friend CVector3D operator- (const
00046         { return CVector3D(-v.x, -v.y, -v.z); }
00047
```

```
00048         // methodes
00049         __forceinline float DotProduct(const CVector3D
00050         __forceinline float GetMagnitude() const
00051         __forceinline void Normalize() { float mag = GetMagnit
00052         __forceinline float GetSquareLength() const {
00053         __forceinline CVector3D Reflect(const CVector3
00054 };
00055
00056
00057 #endif /* #ifndef __CVECTOR3D_H__ */
```

---

Generated on Thu Apr 4 23:14:17 2002 by  1.2.15

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# RayTracerChap1.h

```
00001
00009 #include "CTargetCamera.h"
00010 #include "UtilityLib.h"
00011 #include "CSphere.h"
00012 #include "CScene.h"
```

---

Generated on Thu Apr 4 23:14:17 2002 by  1.2.15


[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# UtilityLib.h

```
00001 #ifndef __UTILITYLIB_H__
00002 #define __UTILITYLIB_H__
00003
00004 #define SAFERELEASE(x) { if (x) delete(x); x=NULL; }
00005
00006 #include "CColor.h"
00007 #include "CVector3D.h"
00008
00009 #endif /* #ifndef __UTILITYLIB_H__ */
```

---

Generated on Thu Apr 4 23:14:17 2002 by  1.2.15

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---



# Graph Legend

This page explains how to interpret the graphs that are generated by doxygen.

Consider the following example:

```
/*! Invisible class because of truncation */
class Invisible { };

/*! Truncated class, inheritance relation is hidden */
class Truncated : public Invisible { };

/* Class not documented with doxygen comments */
class Undocumented { };

/*! Class that is inherited using public inheritance */
class PublicBase : public Truncated { };

/*! A template class */
template<class T> class Templ { };

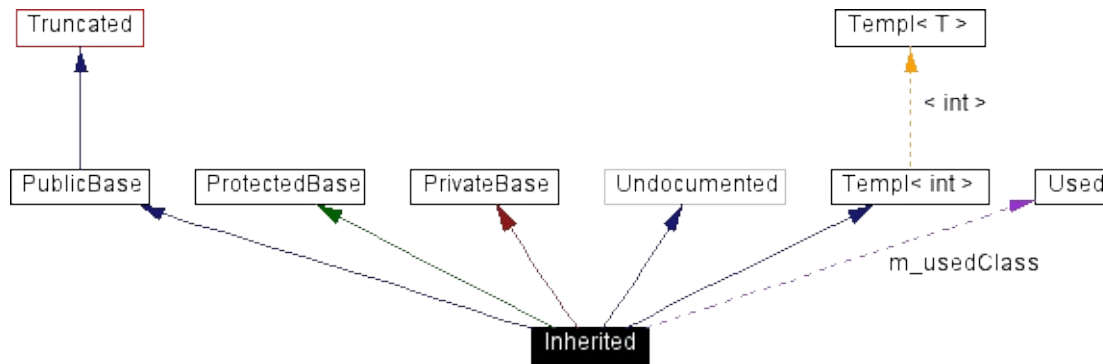
/*! Class that is inherited using protected inheritance */
class ProtectedBase { };

/*! Class that is inherited using private inheritance */
class PrivateBase { };

/*! Class that is used by the Inherited class */
class Used { };

/*! Super class that inherits a number of other classes */
class Inherited : public PublicBase,
                 protected ProtectedBase,
                 private PrivateBase,
                 public Undocumented
                 public Templ<int>
{
    private:
        Used *m_usedClass;
};
```

If the `MAX_DOT_GRAPH_HEIGHT` tag in the configuration file is set to 240 this will result in the following graph:



The boxes in the above graph have the following meaning:

- A filled black box represents the struct or class for which the graph is generated.
- A box with a black border denotes a documented struct or class.
- A box with a grey border denotes an undocumented struct or class.
- A box with a red border denotes a documented struct or class for which not all inheritance/containment relations are shown. A graph is truncated if it does not fit within the specified boundaries.

The arrows have the following meaning:

- A dark blue arrow is used to visualize a public inheritance relation between two classes.
- A dark green arrow is used for protected inheritance.
- A dark red arrow is used for private inheritance.
- A purple dashed arrow is used if a class is contained or used by another class. The arrow is labeled with the variable(s) through which the pointed class or struct is accessible.
- A yellow dashed arrow denotes a relation between a template instance and the template class it was instantiated from. The arrow is labeled with the template parameters of the instance.



# CCamera Member List

This is the complete list of members for [CCamera](#), including all inherited members.

<b>CalcDirVec</b> (float x, float y, int xRes, int yRes) (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	
<b>CCamera</b> () (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[inline]
<b>CCamera</b> (CVector3D &vCamPos, CVector3D &vUpVector) (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	
<b>CNode</b> () (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>CNode</b> (int nodeType, char *name) (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>GetName</b> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>GetInstance</b> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>GetType</b> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>GetPosition</b> () (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[inline]
<b>IntersectsNode</b> (const CRay &ray, CIntersectInfo *intersectInfo=NULL)	<a href="#">CCamera</a>	[inline, virtual]
<b>m_camPos</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_Name</b> (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[protected]
<b>m_NodeType</b> (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[protected]
<b>m_rightVec</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_upVec</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_vecDir</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_viewplaneDist</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_viewplaneHeight</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_viewPlaneUpLeft</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_viewplaneWidth</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>SetName</b> (char *name) (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>SetNodeType</b> (int nodeType) (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>~CCamera</b> () (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[inline, virtual]
<b>~CNode</b> () (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline, virtual]



[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CColor Member List

This is the complete list of members for **CColor**, including all inherited members.

<b>CColor()</b> (defined in <b>CColor</b> )	<b>CColor</b> [inline]
<b>CColor</b> (float r, float g, float b) (defined in <b>CColor</b> )	<b>CColor</b> [inline]
<b>m_b</b> (defined in <b>CColor</b> )	<b>CColor</b>
<b>m_g</b> (defined in <b>CColor</b> )	<b>CColor</b>
<b>m_r</b> (defined in <b>CColor</b> )	<b>CColor</b>
<b>NormalizeColor()</b> (defined in <b>CColor</b> )	<b>CColor</b> [inline]
<b>operator *</b> (defined in <b>CColor</b> )	<b>CColor</b> [friend]
<b>operator *</b> (defined in <b>CColor</b> )	<b>CColor</b> [friend]
<b>operator+</b> (defined in <b>CColor</b> )	<b>CColor</b> [friend]
<b>operator+=</b> (defined in <b>CColor</b> )	<b>CColor</b> [friend]
<b>operator/</b> (defined in <b>CColor</b> )	<b>CColor</b> [friend]
<b>operator/=</b> (defined in <b>CColor</b> )	<b>CColor</b> [friend]
<b>~CColor()</b> (defined in <b>CColor</b> )	<b>CColor</b> [inline, virtual]

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---



# CIntersectInfo Member List

This is the complete list of members for [CIntersectInfo](#), including all inherited members.

<b>CIntersectInfo()</b> (defined in <a href="#">CIntersectInfo</a> )	<a href="#">CIntersectInfo</a>	[inline]
<b>m_Intersection</b> (defined in <a href="#">CIntersectInfo</a> )	<a href="#">CIntersectInfo</a>	
<b>m_Node</b> (defined in <a href="#">CIntersectInfo</a> )	<a href="#">CIntersectInfo</a>	
<b>~CIntersectInfo()</b> (defined in <a href="#">CIntersectInfo</a> )	<a href="#">CIntersectInfo</a>	[inline, virtual]

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CNode Member List

This is the complete list of members for **CNode**, including all inherited members.

<b>CNode()</b> (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>CNode</b> (int nodeType, char *name) (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>GetName()</b> const (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>GetInstance()</b> const (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>GetType()</b> const (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>IntersectsNode</b> (const CRay &ray, CIntersectInfo *intersectInfo=NULL)=0 (defined in <b>CNode</b> )	<b>CNode</b>	[pure virtual]
<b>m_Name</b> (defined in <b>CNode</b> )	<b>CNode</b>	[protected]
<b>m_NodeType</b> (defined in <b>CNode</b> )	<b>CNode</b>	[protected]
<b>SetName</b> (char *name) (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>SetNodeType</b> (int nodeType) (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>~CNode()</b> (defined in <b>CNode</b> )	<b>CNode</b>	[inline, virtual]

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CObject Member List

This is the complete list of members for **CObject**, including all inherited members.

<b>CNode()</b> (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>CNode</b> (int nodeType, char *name) (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>CObject()</b> (defined in <b>CObject</b> )	<b>CObject</b>	[inline]
<b>GetName()</b> const (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>GetNode()</b> const (defined in <b>CObject</b> )	<b>CObject</b>	[inline]
<b>GetInstance()</b> const (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>GetNodeType()</b> const (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>IntersectsNode</b> (const CRay &ray, CIntersectInfo *intersectInfo=NULL)=0 (defined in <b>CObject</b> )	<b>CObject</b>	[pure virtual]
<b>m_Name</b> (defined in <b>CNode</b> )	<b>CNode</b>	[protected]
<b>m_Node</b> (defined in <b>CObject</b> )	<b>CObject</b>	[protected]
<b>m_NodeType</b> (defined in <b>CNode</b> )	<b>CNode</b>	[protected]
<b>SetName</b> (char *name) (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>SetNode</b> (CNode *node) (defined in <b>CObject</b> )	<b>CObject</b>	[inline]
<b>SetNodeType</b> (int nodeType) (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>~CNode()</b> (defined in <b>CNode</b> )	<b>CNode</b>	[inline, virtual]
<b>~CObject()</b> (defined in <b>CObject</b> )	<b>CObject</b>	[inline, virtual]

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CRay Member List

This is the complete list of members for **CRay**, including all inherited members.

<b>CRay()</b> (defined in <b>CRay</b> )	<b>CRay</b>	[inline]
<b>m_t</b> (defined in <b>CRay</b> )	<b>CRay</b>	
<b>m_vDir</b> (defined in <b>CRay</b> )	<b>CRay</b>	
<b>m_vStart</b> (defined in <b>CRay</b> )	<b>CRay</b>	
<b>~CRay()</b> (defined in <b>CRay</b> )	<b>CRay</b>	[inline, virtual]

---

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---



# CScene Member List

This is the complete list of members for [CScene](#), including all inherited members.

<b>AddCamera</b> (CCamera *camera) (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>AddNode</b> (CNode *node) (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>AddObject</b> (CObject *object) (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>CScene</b> () (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>GetActiveCamera</b> () const (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>GetNode</b> (const unsigned int i) const (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>GetNumNodes</b> () const (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>SetActiveCamera</b> (const unsigned int i) (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>SetActiveCamera</b> (CCamera *cam) (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline]
<b>~CScene</b> () (defined in <a href="#">CScene</a> )	<a href="#">CScene</a>	[inline, virtual]

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CSphere Member List

This is the complete list of members for [CSphere](#), including all inherited members.

<a href="#">CNode</a> () (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<a href="#">CNode</a> (int nodeType, char *name) (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<a href="#">CObject</a> () (defined in <a href="#">CObject</a> )	<a href="#">CObject</a>	[inline]
<a href="#">CSphere</a> () (defined in <a href="#">CSphere</a> )	<a href="#">CSphere</a>	[inline]
<a href="#">CSphere</a> (CVector3D center, float radius) (defined in <a href="#">CSphere</a> )	<a href="#">CSphere</a>	[inline]
<a href="#">GetName</a> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<a href="#">GetNode</a> () const (defined in <a href="#">CObject</a> )	<a href="#">CObject</a>	[inline]
<a href="#">GetInstance</a> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<a href="#">GetNodeType</a> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<a href="#">IntersectsNode</a> (const CRay &ray, CIntersectInfo *intersectInfo=NULL) (defined in <a href="#">CSphere</a> )	<a href="#">CSphere</a>	[virtual]
<a href="#">m_Name</a> (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[protected]
<a href="#">m_Node</a> (defined in <a href="#">CObject</a> )	<a href="#">CObject</a>	[protected]
<a href="#">m_NodeType</a> (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[protected]
<a href="#">SetName</a> (char *name) (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<a href="#">SetNode</a> (CNode *node) (defined in <a href="#">CObject</a> )	<a href="#">CObject</a>	[inline]
<a href="#">SetNodeType</a> (int nodeType) (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<a href="#">~CNode</a> () (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline, virtual]
<a href="#">~CObject</a> () (defined in <a href="#">CObject</a> )	<a href="#">CObject</a>	[inline, virtual]
<a href="#">~CSphere</a> () (defined in <a href="#">CSphere</a> )	<a href="#">CSphere</a>	[inline, virtual]

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CTargetCamera Member List

This is the complete list of members for [CTargetCamera](#), including all inherited members.

<b>CalcDirVec</b> (float x, float y, int xRes, int yRes) (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	
<b>CCamera</b> () (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[inline]
<b>CCamera</b> (CVector3D &vCamPos, CVector3D &vUpVector) (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	
<b>CNode</b> () (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>CNode</b> (int nodeType, char *name) (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>CTargetCamera</b> () (defined in <a href="#">CTargetCamera</a> )	<a href="#">CTargetCamera</a>	[inline]
<b>CTargetCamera</b> (CVector3D &camPos, CVector3D &lookAtPoint, CVector3D &upVector) (defined in <a href="#">CTargetCamera</a> )	<a href="#">CTargetCamera</a>	
<b>GetName</b> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>GetInstance</b> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>GetType</b> () const (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[inline]
<b>GetPosition</b> () (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[inline]
<b>IntersectsNode</b> (const CRay &ray, CIntersectInfo *intersectInfo=NULL)	<a href="#">CCamera</a>	[inline, virtual]
<b>m_camPos</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_Name</b> (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[protected]
<b>m_NodeType</b> (defined in <a href="#">CNode</a> )	<a href="#">CNode</a>	[protected]
<b>m_rightVec</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_upVec</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_vecDir</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_viewplaneDist</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_viewplaneHeight</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
<b>m_viewPlaneUpLeft</b> (defined in <a href="#">CCamera</a> )	<a href="#">CCamera</a>	[protected]
		[protected]

<b>m_viewplaneWidth</b> (defined in <b>CCamera</b> )	<b>CCamera</b>	
<b>SetName</b> (char *name) (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>SetNodeType</b> (int nodeType) (defined in <b>CNode</b> )	<b>CNode</b>	[inline]
<b>~CCamera</b> () (defined in <b>CCamera</b> )	<b>CCamera</b>	[inline, virtual]
<b>~CNode</b> () (defined in <b>CNode</b> )	<b>CNode</b>	[inline, virtual]
<b>~CTargetCamera</b> () (defined in <b>CTargetCamera</b> )	<b>CTargetCamera</b>	[inline, virtual]

[Main Page](#) [Class Hierarchy](#) [Compound List](#) [File List](#) [Compound Members](#)

---

# CVector3D Member List

This is the complete list of members for [CVector3D](#), including all inherited members.

<b>CVector3D()</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [inline]
<b>CVector3D</b> (float _x, float _y, float _z) (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [inline]
<b>DotProduct</b> (const CVector3D &v) (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [inline]
<b>GetMagnitude</b> () const (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [inline]
<b>GetSquareLength</b> () const (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [inline]
<b>Normalize</b> () (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [inline]
<b>operator *</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [friend]
<b>operator *</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [friend]
<b>operator+</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [friend]
<b>operator+=</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [friend]
<b>operator-</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [friend]
<b>operator-</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [friend]
<b>operator/</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [friend]
<b>operator/=</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [friend]
<b>Reflect</b> (const CVector3D &n) (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [inline]
<b>x</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a>
<b>y</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a>
<b>z</b> (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a>
<b>~CVector3D</b> () (defined in <a href="#">CVector3D</a> )	<a href="#">CVector3D</a> [inline, virtual]