Graphics and Media Lab CSL Reference Manual

1.0.0

Generated by Doxygen 1.3.4

Tue Jan 13 21:12:01 2004

Contents

Chapter 1

Graphics and Media Lab CSL Main Page

1.1 Introduction

Graphics & Media Lab Common Source Library is a C++ library that contains a set of classes frequently used in graphics research and development.

1.2 General description

The library contains several packages, separated by particular functionality. The following sections describe them.

1.2.1 Base

This package contains the most general classes and utilities:

- definitions like ASSERT(), VERIFY() and NULL
- gml::String type definition (derived from std::string)
- smart pointer facility (gml::Ref, gml::SmartObject
- object-orienter callback facility (gml::Callback)

1.2.2 Math

Math package contains the most general math utilities: vectors, matrices, etc.

- vectors: 2D, 3D, 4D (gml::TVector2, gml::TVector3, gml::TVector4)
- matrices: 2x2, 3x3, 4x4 (gml::TMatrix2x2, gml::TMatrix3x3, gml::TMatrix4x4)
- quaternions (gml::TQuaternion)
- bounding boxes (2D, 3D) and bounding spheres (gml::BBox2, gml::BBox2, gml::BSphere3)

The library was designed independent on particular compiler, though currently we assume Win32 platform. It uses neigher MFC nor VCL or any other popular API. At the same time there is preprocessor switches that allow integration of GML data types with MFC/VCL data types.

The library uses STL and assumes STL usage by the programmer (though it is not a necessary condition)

1.2.3 Color

Color package defines gml::TColor3 and gml::TColor4 clases that represent color values.

1.2.4 Files

Files package defines some common utilities to work with filesystem:

- gml::File provides an object-oriented way to read & write to text and finary files (FILE∗ wrapper)
- gml::PathString to parse strings containing file & path names
- gml::FileFinder used to enumerate files directories
- CIniFile provides facility for loading & saving of INI-files

1.2.5 Images

This is the most complete package so far. It defines several classes to work with different kinds of images, apply filters, etc.

- gml::Image is a base class, it has several descendant, like gml::SimpleImage, gml::IntelImage, gml::GDIImage
- filters can be applied to any image independent of its type Many filters have been implemented: blurring, color conversion, resizing, etc
- Images can loaded/saved in multiple formats using either internal interface or external library FreeImage.

1.2.6 Net

Net package defines single class gml::Socket, which is a simple wrapper over standart sockets functionality.

1.2.7 Utils3D

This package have some classes, useful for developing 3D graphics application independent on different APIs like OpenGL or DirectX.

- gml::Camera and gml::Viewport classes
- gml::Browser class is a helper for navigation through a 3D scene
- gml::FrameCounter allow to calculate average frame rates (FPS)
- also several utility functions available.

1.2.8 UtilsGL

UtilsGL contains some useful utilities for OpenGL programming:

- OpenGL extentions handling (taken from NVidia SDK)
- OGL Win32 context management (gml::GLRC)

GML Common Source Library is contastantly evolving, so all the packages are updated frequently.

1.3 Installing and Usage

- Unpack the archive
- Link with necessary packages (depends on functionalities you are going to use). Note: all the packages depend on Base and on Math
- The header <CSL/base/gmlcommon.h> must be included before any other includes of GML files. It usually worth to include this file into precompiled headers list (stdafx.h for MSVC)

Chapter 2

Graphics and Media Lab CSL Module Index

2.1 Graphics and Media Lab CSL Modules

Here is a list of all modules:

Chapter 3

Graphics and Media Lab CSL Hierarchical Index

3.1 Graphics and Media Lab CSL Class Hierarchy

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

Chapter 4

Graphics and Media Lab CSL Class Index

4.1 Graphics and Media Lab CSL Class List

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


```
GLRC glrc;
// inside window creation
glrc.Create(wnd);
// before rendering into window
glrc.MakeCurrent();
// ...
// display in the screen (in case of double buffering)
glrc.SwapBuffers();
// ..
// delete gl context (may be avoided, because called in destructor anyway)
glrc.Destroy();
```

)**??**


```
Object* obj; // standard pointer, should be deleted after use
gml::Ref<Object> obj; // smart pointer frees memory when necessary
```

Attention:

Smart pointers works only with objects that support AddRef() and Release() operations. It is assumed that object keeps a reference count (a number of smapt pointers pointing to it). This reference count is incremented in AddRef() command and decremented in Release(). Also Release() performs check whether reference count is 0 and calls 'delete this'. GML provides special base class for this: gml::SmartObject. If you want to use smart pointers either derive your object from this class or provide own implementation of AddRef() and Release() methods. 1) initialization Smart pointers are initiliazed just like the standard pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
```

Smart pointers must never be defined like this: gml::Rev<Object>∗ obj; !!! 2) use of smart pointers Smart pointers are dereferenced like normal pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
obj->SomeMethod();
(*obj).SomeMethod();
```

3) smart pointers as parameters Smart pointers are implicitly casted to normal pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
Object* obj1 =  obj;
```

This operation is dangerous unless used properly. You should use it like follows. Smart pointers may be passed to functions and returned from functions like normal pointers.

```
...
Object* SomeMethod(Object* obj);
...
gml::Ref<Object> obj =  new Object(1,2,3);
gml::Ref<Object> obj1 = SomeMethod(obj);
```

The general rule is this: **Use ONLY smart pointers when storing objects. Use normals pointers when passing smart pointers as parameters or return values** So the following is forbidden:

```
class SomeClass
  {
  private:
    Object* obj;
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      } // at this point obj will be deleted, and m_obj will be undefined!!
  };
```

Correct use:

```
class SomeClass
  {
  private:
    gml::Ref<Object> obj;
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      }
  };
```

)**??**

Chapter 5

Graphics and Media Lab CSL File Index

5.1 Graphics and Media Lab CSL File List

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

Chapter 6

Graphics and Media Lab CSL Page Index

6.1 Graphics and Media Lab CSL Related Pages

Here is a list of all related documentation pages:

Chapter 7

Graphics and Media Lab CSL Module Documentation

7.1 DScene

Classes

- class **Mesh**

7.2 Base

Classes

- class Callback

  Base class for callback with no arguments and no return value.

- class **CallbackR**
- class **Callback1**
- class **CallbackR1**
- class **Callback2**
- class **CallbackR2**
- class **Callback3**
- class **CallbackR3**
- class **FunctionCallback**
- class **FunctionCallbackR**
- class **FunctionCallback1**
- class **FunctionCallbackR1**
- class **FunctionCallback2**
- class **FunctionCallbackR2**
- class **FunctionCallback3**
- class **FunctionCallbackR3**
- class **MethodCallback**
- class **MethodCallbackR**
- class **MethodCallback1**
- class **MethodCallbackR1**
- class **MethodCallback2**
- class **MethodCallbackR2**
- class **MethodCallback3**
- class **MethodCallbackR3**
- class Ref

  Smart pointer class Smart pointers provide automatic memory management, so that programmer never cares on freeing memory. Smart pointers mechanism frees an object when there are no any references to it. Smart pointers should be used instead of standard C pointers:

```
Object* obj; // standard pointer, should be deleted after use
gml::Ref<Object> obj; // smart pointer frees memory when necessary
```

  Attention:

  Smart pointers works only with objects that support AddRef() and Release() operations. It is assumed that object keeps a reference count (a number of smapt pointers pointing to it). This reference count is incremented in AddRef() command and decremented in Release(). Also Release() performs check whether reference count is 0 and calls 'delete this'. GML provides special base class for this: gml::SmartObject. If you want to use smart pointers either derive your object from this class or provide own implementation of AddRef() and Release() methods. 1) initialization Smart pointers are initiliazed just like the standard pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
```

  Smart pointers must never be defined like this: gml::Rev<Object> obj; !!! 2) use of smart pointers Smart pointers are dereferenced like normal pointers:*

```
gml::Ref<Object> obj =  new Object(1,2,3);
obj->SomeMethod();
(*obj).SomeMethod();
```

  3) smart pointers as parameters Smart pointers are implicitly casted to normal pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
Object* obj1 =  obj;
```

This operation is dangerous unless used properly. You should use it like follows. Smart pointers may be passed to functions and returned from functions like normal pointers.

```
...
Object* SomeMethod(Object* obj);
...
gml::Ref<Object> obj =  new Object(1,2,3);
gml::Ref<Object> obj1 = SomeMethod(obj);
```

The general rule is this: **Use ONLY smart pointers when storing objects. Use normals pointers when passing smart pointers as parameters or return values** *So the following is forbidden:*

```
class SomeClass
  {
  private:
    Object* obj;
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      } // at this point obj will be deleted, and m_obj will be undefined!!
  };
```

Correct use:

```
class SomeClass
  {
  private:
    gml::Ref<Object> obj;
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      }
  };
```

.

- class SmartObject

  Base class for objects with support for reference counting.

- class String

  Standard string with some extensions.

- class Ref

  Smart pointer class Smart pointers provide automatic memory management, so that programmer never cares on freeing memory. Smart pointers mechanism frees an object when there are no any references to it. Smart pointers should be used instead of standard C pointers:

  ```
  Object* obj; // standard pointer, should be deleted after use
  gml::Ref<Object> obj; // smart pointer frees memory when necessary
  ```

  Attention:

  Smart pointers works only with objects that support AddRef() and Release() operations. It is assumed that object keeps a reference count (a number of smapt pointers pointing to it). This reference count is incremented in AddRef() command and decremented in Release(). Also Release() performs check whether reference count is 0 and calls 'delete this'. GML provides special base class for this: gml::SmartObject. If you want to use smart pointers either derive your object from this class or provide own implementation of AddRef() and Release() methods. 1) initialization Smart pointers are initiliazed just like the standard pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
```

Smart pointers must never be defined like this: gml::Rev<Object> obj; !!! 2) use of smart pointers Smart pointers are dereferenced like normal pointers:*

```
gml::Ref<Object> obj =  new Object(1,2,3);
obj->SomeMethod();
(*obj).SomeMethod();
```

3) smart pointers as parameters Smart pointers are implicitly casted to normal pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
Object* obj1 =  obj;
```

This operation is dangerous unless used properly. You should use it like follows. Smart pointers may be passed to functions and returned from functions like normal pointers.

```
...
Object* SomeMethod(Object* obj);
...
gml::Ref<Object> obj =  new Object(1,2,3);
gml::Ref<Object> obj1 = SomeMethod(obj);
```

The general rule is this: **Use ONLY smart pointers when storing objects. Use normals pointers when passing smart pointers as parameters or return values** *So the following is forbidden:*

```
class SomeClass
  {
  private:
    Object* obj;
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      } // at this point obj will be deleted, and m_obj will be undefined!!
  };
```

Correct use:

```
class SomeClass
  {
  private:
    gml::Ref<Object> obj;
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      }
  };
```

.

Functions

- Ref< newT > & gml::Ref::ConvRef (const Ref< oldT > &old_ref)

  Casting & convertion of smart references.

7.3 Color

Classes

- class TColor3

  3 Color repsentation

- class TColor4

  4 Color repsentation

Typedefs

- typedef TColor3< BYTE > **Color3ub**
- typedef TColor3< WORD > **Color3w**
- typedef TColor3< short > **Color3s**
- typedef TColor3< float > **Color3f**
- typedef TColor4< BYTE > **Color4ub**
- typedef TColor4< float > **Color4f**
- typedef TColor3< BYTE > gml::ColorRGBub
- typedef TColor3< WORD > gml::ColorRGBw
- typedef TColor3< float > gml::ColorRGBf
- typedef TColor4< BYTE > gml::ColorRGBAub
- typedef TColor4< float > gml::ColorRGBAf

Functions

- TColor3< T > gml::TColor3::operator * (const double d, const TColor3< T > &u)

  Multiplication of scalar d by TColor3 u.

- TColor4< T > gml::TColor4::operator * (const double d, const TColor4< T > &u)

  Multiplication of scalar d by TColor4 u.

7.3.1 Typedef Documentation

7.3.1.1 typedef TColor3<BYTE> ColorRGBub

Deprecated
use Color3 instead

7.3.1.2 typedef TColor3<WORD> ColorRGBw

Deprecated
use Color3 instead

7.3.1.3 typedef TColor3<float> ColorRGBf

Deprecated
use Color3 instead

7.3.1.4 typedef TColor4<BYTE> ColorRGBAub

Deprecated
use Color3 instead

7.3.1.5 typedef TColor4<float> ColorRGBAf

Deprecated
use Color3 instead

7.4 Files

Classes

- class File

  A class for working with files (a wrapped to FILE ∗).

- class FileFinder
- class CIniFile

  Handling of INI-files.

- class PathString

  Processing of the path to file.

7.5 ImageFilters

Classes

- class FilterGrayscale

  Creates a grayscale version of a color bitmap.

- class FilterLum2RGB

  Creates a Lum2RGB version of a color bitmap.

- class FilterMedianBlur

  Creates a MedianBlur version of a color bitmap.

- class FilterRepres

  Transferres between different representations.

- class FilterResize

  Creates a MedianBlur version of a color bitmap.

- class FilterRGB2BGR

  Creates a Lum2RGB version of a color bitmap.

- class FilterRGB2RGBA

  Creates a grayscale version of a color bitmap.

- class ReflectFilter

  Creates a grayscale version of a color bitmap.

- class SetRectFilter

  Creates a grayscale version of a color bitmap.

- class ImageComposer

  Produce single image from several reference images.

- class ImageDecomposer

  Decompose an image into several destination images.

- class ImageFilter

  Interface for generic image modifier.

7.6 Images

Modules

- ImageFilters

Classes

- class GDIImage

  Manipulates uncompressed device- and platform-independent bitmaps.

- class GDISurface

  Generic image class.

- class Image

  Generic image class.

- class ImageLoader

  Loading and saving bitmaps to files (base class).

- class IntelImage

  Manipulates uncompressed device- and platform-independent bitmaps.

- class IntelSurface

  Generic image class.

- class SimpleImage

  Manipulates uncompressed device- and platform-independent bitmaps.

- class FRImageLoader

  Image loader based on FreeImage third-party library.

- class DIntelImage

  Intel libraries IplImage wrap class.

- class SimpleImageLoader

  Image loader based on FreeImage third-party library.

Functions

- gml::GDIImage::GDIImage (const GDIImage &Orig)

  Copy constructor.

- gml::GDIImage::GDIImage (const Image &Orig)

  Copy constructor.

- GDIImage & gml::GDIImage::operator= (const Image &Orig)

  Assignment operator.

- GDIImage & **operator=** (const GDIImage &Orig)
- Image & gml::Image::operator= (Image const &Orig)
- int **GetWidth** () const
- int **GetHeight** () const
- FORMAT **GetFormat** () const
- REPRES **GetRepres** () const
- ORIENT **GetOrient** () const
- int **GetElemSize** () const
- BYTE ** gml::Image::GetLineArray () const

  Returns pointer to an array containing the starting addresses of the bitmap lines.

- bool **IsLocked** () const
- gml::IntelImage::IntelImage (const IntelImage &Orig)

  Copy constructor.

- gml::IntelImage::IntelImage (const Image &Orig)

  Copy constructor.

- IntelImage & gml::IntelImage::operator= (const Image &Orig)

  Assignment operator.

- IntelImage & **operator=** (const IntelImage &Orig)
- gml::SimpleImage::SimpleImage (const SimpleImage &Orig)

  Copy constructor.

- gml::SimpleImage::SimpleImage (const Image &Orig)

  Copy constructor.

- SimpleImage & gml::SimpleImage::operator= (const Image &Orig)

  Assignment operator.

- SimpleImage & **operator=** (const SimpleImage &Orig)

Variables

- const **PRESERVE_JPEG_EXIF** = 1

7.6.1 Function Documentation

7.6.1.1 Image & operator= (Image const & *Orig*) [inline, inherited]

Assignment operator. Note that assignment between different derived classes is possible and results in a format conversion.

Reimplemented in GDIImage, IntelImage, and SimpleImage.

7.6.1.2 BYTE ** GetLineArray () const [inline, inherited]

Returns pointer to an array containing the starting addresses of the bitmap lines.

This array should be used whenever the bitmap bits need to be manipulated directly.

7.7 Math2

Classes

- class BBox2

  A template class for 2D boundary box.

- class TVector2

  Template class for 2D vectors.

- class Math2

  Comparison of 2D vectors with a tolerance.

Typedefs

- typedef BBox2< double > **BBox2d**
- typedef BBox2< float > **BBox2f**
- typedef BBox2< int > **BBox2i**
- typedef TVector2< short > **Vector2s**
- typedef TVector2< int > **Vector2i**
- typedef TVector2< float > **Vector2f**
- typedef TVector2< double > **Vector2d**
- typedef Math2< float > **Math2f**
- typedef Math2< double > **Math2d**

Functions

- TVector2< T > gml::TVector2::operator ∗ (const double d, const TVector2< T > &u)

  Multiplication of scalar d by TVector u.

- double gml::TVector2::SqrLength (const TVector2< T > &u)

  Squared length of given vector.

- double gml::TVector2::Length (const TVector2< T > &u)

  Length of the vector.

- double gml::TVector2::DotProd (const TVector2< T > &v1, const TVector2< T > &v2)

  Dot product.

- double gml::TVector2::DotProduct (const TVector2< T > &v1, const TVector2< T > &v2)

  Dot product.

- double gml::TVector2::CrossProd (const TVector2< T > &v1, const TVector2< T > &v2)

  Cross product.

- double gml::TVector2::CrossProduct (const TVector2< T > &v1, const TVector2< T > &v2)

  Cross product.

- double gml::TVector2::Cos (const TVector2< T > &a, const TVector2< T > &b)
  cos between two vectors
- double gml::TVector2::Sin (const TVector2< T > &a, const TVector2< T > &b)
  sin between two vectors
- TVector2< T_TO > gml::TVector2::Conv (const TVector2< T_FROM > &u)
  Convert TVector2<T_FROM> to TVector2<T_TO>.
- TVector2< float > gml::TVector2::ConvF (const TVector2< T > &u)
  Convert TVector2<T> to TVector2<float>.
- TVector2< double > gml::TVector2::ConvD (const TVector2< T > &u)
  Convert TVector2<T> to TVector2<double>.
- TVector2< int > gml::TVector2::ConvI (const TVector2< T > &u)
  Convert TVector2<T> to TVector2<int>.
- TVector2< short > gml::TVector2::ConvS (const TVector2< T > &u)
  Convert TVector2<T> to TVector2<int>.

7.8 Math3

Classes

- class BBox3

  A template class for 2D boundary box.

- class BSphere3

  Template class for 3D bounding sphere.

- class TVector3

  Template class for 3D geometric vectors.

- class Math3

  Comparison of 2D vectors with a tolerance.

Typedefs

- typedef BBox3< double > **BBox3d**
- typedef BBox3< float > **BBox3f**
- typedef BBox3< int > **BBox3i**
- typedef BSphere3< float > **BSphere3f**
- typedef BSphere3< double > **BSphere3d**
- typedef TVector3< short > **Vector3s**
- typedef TVector3< int > **Vector3i**
- typedef TVector3< float > **Vector3f**
- typedef TVector3< double > **Vector3d**
- typedef Math3< float > **Math3f**
- typedef Math3< double > **Math3d**

Functions

- template<class T_FROM> BBox3< double > **ConvD** (const BBox3< T_FROM > &u)
- template<class T_FROM> BBox3< float > **ConvF** (const BBox3< T_FROM > &u)
- TVector3< T > gml::TVector3::operator ∗ (const double d, const TVector3< T > &u)

  Multiplication of scalar d by TVector u.

- double gml::TVector3::SqrLength (const TVector3< T > &u)

  Squared length of given vector.

- double gml::TVector3::Length (const TVector3< T > &u)

  Length of the vector.

- double gml::TVector3::DotProduct (const TVector3< T > &v1, const TVector3< T > &v2)

  Dot product.

- TVector3< T > gml::TVector3::CrossProduct (const TVector3< T > &v1, const TVector3< T > &v2)

Cross product.

- double gml::TVector3::Cos (const TVector3< T > &a, const TVector3< T > &b)
  cos between two vectors
- double gml::TVector3::Sin (const TVector3< T > &a, const TVector3< T > &b)
  sin between two vectors
- TVector3< T_TO > gml::TVector3::Conv (const TVector3< T_FROM > &u)
  Convert TVector3<T_FROM> to TVector3<T_TO>.
- TVector3< float > gml::TVector3::ConvF (const TVector3< T > &u)
  Convert TVector3<T> to TVector3<float>.
- TVector3< double > gml::TVector3::ConvD (const TVector3< T > &u)
  Convert TVector3<T> to TVector3<double>.
- TVector3< int > gml::TVector3::ConvI (const TVector3< T > &u)
  Convert TVector3<T> to TVector3<int>.
- TVector3< short > gml::TVector3::ConvS (const TVector3< T > &u)
  Convert TVector3<T> to TVector3<int>.

7.9 Math

Modules

- Math2x2
- Math3x3
- Math4x4
- Math2
- Math3
- Math4

Classes

- class Math

  A template class for precision related staff.

- class TQuaternion

  quaternion template

Defines

- #define **PI** 3.1415926535897

Typedefs

- typedef Math< float > **MathF**
- typedef Math< double > **MathD**
- typedef TQuaternion< float > **Quaternionf**
- typedef TQuaternion< double > **Quaterniond**

Functions

- double **ToDegrees** (double radians)
- double **ToRadians** (double degrees)
- template<class T> void gml::Clip (T &v, const double a_min, const double a_max)

  Clip a value of given variable by given range for different types T.

- template<class T> T gml::Clipped (const T v, const double a_min, const double a_max)

  Clip a value of given variable by given range for different types T.

- template<class T> T gml::Min (const T u, const T v)

  Find minimal value for different type T.

- template<class T> T gml::Min3 (const T u, const T v, const T w)

  Find minimal value for different type T within three vars.

- template<class T> T gml::Max (const T u, const T v)

Find maximal value for different type T.

- template<class T> T gml::Max3 (const T u, const T v, const T w)

  Find maximal value for different type T within three vars.

- template<class T> bool **operator==** (const TQuaternion< T > &q1, const TQuaternion< T > &q2)
- template<class T> bool **operator!=** (const TQuaternion< T > &q1, const TQuaternion< T > &q2)

- template<class T> TQuaternion< T > **operator** * (const TQuaternion< T > &q1, const TQuaternion< T > &q2)
- TQuaternion< T_TO > gml::TQuaternion::Conv (const TQuaternion< T_FROM > &u)

  Convert TQuaternion<T_FROM> to TQuaternion<T_TO>.

- TQuaternion< float > gml::TQuaternion::ConvF (const TQuaternion< T > &u)

  Convert TQuaternion<T> to TQuaternion<float>.

- TQuaternion< double > gml::TQuaternion::ConvD (const TQuaternion< T > &u)

  Convert TQuaternion<T> to TQuaternion<double>.

Variables

- const double **RAD_TO_DEG** = 57.295779513082320876798154814105 2
- const double **DEG_TO_RAD** = 0.017453292519943295769236907684886 1
- const int **QUATERNION_NORMALIZATION_THRESHOLD** = 64

7.10 Math2x2

Classes

- class TMatrix2x2

  Template class for 2x2 matrix.

Typedefs

- typedef TMatrix2x2< double > **Matrix2x2d**
- typedef TMatrix2x2< float > **Matrix2x2f**

Functions

- TMatrix2x2< T_TO > gml::TMatrix2x2::Conv (const TMatrix2x2< T_FROM > &u)

  Convert TMatrix2x2<T_FROM> to TMatrix2x2<T_TO>.

- TMatrix2x2< float > gml::TMatrix2x2::ConvF (const TMatrix2x2< T > &u)

  Convert TMatrix2x2<T> to TMatrix2x2<float>.

- TMatrix2x2< double > gml::TMatrix2x2::ConvD (const TMatrix2x2< T > &u)

  Convert TMatrix2x2<T> to TMatrix2x2<double>.

7.11 Math3x3

Classes

- class TMatrix3x3

  Template class for 3x3 matrix.

Typedefs

- typedef TMatrix3x3< double > **Matrix3x3d**
- typedef TMatrix3x3< float > **Matrix3x3f**

Functions

- template<class T> TMatrix3x3< T > **MultVectStr** (TVector3< T > &v1, TVector3< T > &v2)
- TMatrix3x3< T_TO > gml::TMatrix3x3::Conv (const TMatrix3x3< T_FROM > &u)

  Convert TMatrix3x3<T_FROM> to TMatrix3x3<T_TO>.

- TMatrix3x3< float > gml::TMatrix3x3::ConvF (const TMatrix3x3< T > &u)

  Convert TMatrix3x3<T> to TMatrix3x3<float>.

- TMatrix3x3< double > gml::TMatrix3x3::ConvD (const TMatrix3x3< T > &u)

  Convert TMatrix3x3<T> to TMatrix3x3<double>.

7.12 Math4x4

Classes

- class TMatrix4x4

  Template class for 4x4 matrix.

Typedefs

- typedef TMatrix4x4< double > **Matrix4x4d**
- typedef TMatrix4x4< float > **Matrix4x4f**

Functions

- TMatrix4x4< T_TO > gml::TMatrix4x4::Conv (const TMatrix4x4< T_FROM > &u)

  Convert TMatrix4x4<T_FROM> to TMatrix4x4<T_TO>.

- TMatrix4x4< float > gml::TMatrix4x4::ConvF (const TMatrix4x4< T > &u)

  Convert TMatrix4x4<T> to TMatrix4x4<float>.

- TMatrix4x4< double > gml::TMatrix4x4::ConvD (const TMatrix4x4< T > &u)

  Convert TMatrix4x4<T> to TMatrix4x4<double>.

7.13 Math4

Classes

- class TVector4

  Template class for 4D geometric vectors.

- class Math4

  Comparison of 4D vectors with a tolerance.

Typedefs

- typedef TVector4< short > **Vector4s**
- typedef TVector4< int > **Vector4i**
- typedef TVector4< float > **Vector4f**
- typedef TVector4< double > **Vector4d**
- typedef Math4< float > **Math4f**
- typedef Math4< double > **Math4d**

Functions

- TVector4< T > gml::TVector4::operator * (const double d, const TVector4< T > &u)

  Multiplication of scalar d by TVector u.

- double gml::TVector4::SqrLength (const TVector4< T > &u)

  Squared length of given vector.

- double gml::TVector4::Length (const TVector4< T > &u)

  Length of the vector.

- double gml::TVector4::DotProduct (const TVector4< T > &v1, const TVector4< T > &v2)

  Dot product.

- TVector4< T_TO > gml::TVector4::Conv (const TVector4< T_FROM > &u)

  Convert TVector4<T_FROM> to TVector4<T_TO>.

- TVector4< float > gml::TVector4::ConvF (const TVector4< T > &u)

  Convert TVector4<T> to TVector4<float>.

- TVector4< double > gml::TVector4::ConvD (const TVector4< T > &u)

  Convert TVector4<T> to TVector4<double>.

- TVector4< int > gml::TVector4::ConvI (const TVector4< T > &u)

  Convert TVector4<T> to TVector4<int>.

- TVector4< short > gml::TVector4::ConvS (const TVector4< T > &u)

  Convert TVector4<T> to TVector4<int>.

7.14 Net

Classes

- class Socket

  A wrapper for Win32 stream socket.

7.15 Utils3D

Classes

- class **ModeBrowser**
- class **TrackballBrowser**
- class **WalkBrowser**
- class Browser

  3D navigation helper class

- class Camera

  Class representing 3d virtual camera.

- class FrameCounter

  Simple frame rate (FPS) counter.

- class Viewport

  Class representing 3d viewport (a window with accociated camera).

Functions

- void **CalculateClipPlanes** (const Viewport &view, const BBox3d &domain, double &p_near, double &p_far)

7.16 UtilsGL

Classes

- class GLRC

  OpenGL context handling

  Attention:

  Include windows.h before using this file! Usage:

```
GLRC glrc;
// inside window creation
glrc.Create(wnd);
// before rendering into window
glrc.MakeCurrent();
// ...
// display in the screen (in case of double buffering)
glrc.SwapBuffers();
// ..
// delete gl context (may be avoided, because called in destructor anyway)
glrc.Destroy();
```

  .

- class GLRC

  OpenGL context handling

  Attention:

  Include windows.h before using this file! Usage:

```
GLRC glrc;
// inside window creation
glrc.Create(wnd);
// before rendering into window
glrc.MakeCurrent();
// ...
// display in the screen (in case of double buffering)
glrc.SwapBuffers();
// ..
// delete gl context (may be avoided, because called in destructor anyway)
glrc.Destroy();
```

  .

Functions

- int gml::InitGLExtensions (const char ∗origReqExts)

  Return 1 if all extensions listed in origReqExts are supported by GL implementation.

- const char ∗ gml::GetUnsupportedGLExtensions ()

  Return a list of unsupported extensions (should be called after InitGLExtensions).

- void **RenderBBox** (const gml::BBox3f &bbox, const gml::ColorRGBAf &color)

7.16.1 Function Documentation

7.16.1.1 int InitGLExtensions (const char ∗ *origReqExts*)

Return 1 if all extensions listed in origReqExts are supported by GL implementation.

Extensions should be separated with spaces, like "GL_ARB_multitexture " "GL_NV_evaluators " "GL_-NV_texture_shader "

Chapter 8

Graphics and Media Lab CSL Class Documentation

8.1 BBox2 Class Template Reference

A template class for 2D boundary box.

`#include <gmlbbox2.h>`

Public Member Functions

Constructors

- BBox2 ()
  no initialization
- BBox2 (const TVector2< T > &point)
  from point
- BBox2 (const TVector2< T > &point1, const TVector2< T > &point2)
  from two points

Comparison operators and methods

- bool NotEmpty () const
  Test for validity.
- bool IsDot () const
  If the box is actually a dot.
- bool Includes (const TVector2< T > &point) const
  Test for point inclusion.
- bool Includes (const BBox2< T > &box) const

Test for box inclusion.

- bool Intersects (const BBox2< T > &box) const

  Test for intersection.

Miscellanious methods

- void Include (const TVector2< T > &point)

  Expand the box to include the given point.

- void Include (const TVector2< T > *points, int npoints)

  Expand the box to include the given points.

- void Include (const BBox2< T > &box)

  Expand the box to include the given box.

- void Include (const BBox2< T > *bboxes, int nbboxes)

  Expand the box to include the given boxes.

- void Intersect (const BBox2< T > &box)

  Intersect with given box.

- void SetRect (const T &in_Left, const T &in_Top, const T &in_Right, const T &in_Bottom)

  Set box sizes.

- void ExpandBy (const T &in_Value)

  Adjust size by given value (give < 0 to deflate).

- void ExpandBy (const T &in_HValue, const T &in_VValue)

  Adjust sizes by given value (give < 0 to deflate).

- void Translate (const TVector2< T > &vct)

  Translate the box to given vector.

- BBox2< T > Translated (const TVector2< T > &vct) const

  Returns box translated by vct.

- TVector2< T > Diag () const

  Return diagonal.

- TVector2< T > Center () const

  Return Center of this box.

- double Volume () const

  Return value of the volume.

- double Width () const

  Return value of width.

- double Height () const

  Return value of height.

- **BBox2** (const tagRECT &p)
- BBox2< T > & **operator=** (const tagRECT &p)
- **operator tagRECT** ()
- **operator CvRect** ()

Public Attributes

- TVector2< T > vmin

  Minimal boundary.

- TVector2< T > vmax

  Maximal boundary.

8.1.1 Detailed Description

template<class T> class gml::BBox2< T >

A template class for 2D boundary box.

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

- gmlbbox2.h

8.2 BBox3 Class Template Reference

A template class for 2D boundary box.

`#include <gmlbbox3.h>`

Public Member Functions

Constructors

- BBox3 ()
  no initialization
- BBox3 (const TVector3< T > &point)
  from point
- BBox3 (const TVector3< T > &point1, const TVector3< T > &point2)
  from two points

Comparison operators and methods

- bool NotEmpty () const
  Test for validity.
- bool IsDot () const
  If the box is actually a dot.
- bool Includes (const TVector3< T > &point) const
  Test for point inclusion.
- bool Includes (const BBox3< T > &box) const
  Test for box inclusion.
- bool Intersects (const BBox3< T > &box) const
  Test for intersection.

Miscellanious methods

- void Include (const TVector3< T > &point)
  Expand the box to include the given point.
- void Include (const TVector3< T > *points, int npoints)
  Expand the box to include the given points.
- void Include (const BBox3< T > &box)
  Expand the box to include the given box.
- void Include (const BBox3< T > *bboxes, int nbboxes)

Expand the box to include the given boxes.

- void Intersect (const BBox3< T > &box)
  Intersect with given box.
- void ExpandBy (const T &in_Value)
  Adjust size by given value (give < 0 to deflate).
- void ExpandBy (const TVector3< T > &expand_vec)
  Adjust sizes by given value (give < 0 to deflate).
- void Translate (const TVector3< T > &vct)
  Translate the box to given vector.
- BBox3< T > Translated (const TVector3< T > &vct) const
  Returns box translated by vct.
- TVector3< T > Diag () const
  Return diagonal.
- TVector3< T > Center () const
  Return Center of this box.
- double Volume () const
  Return value of the volume.

Public Attributes

- TVector3< T > vmin
  Minimal boundary.
- TVector3< T > vmax
  Maximal boundary.

8.2.1 Detailed Description

template<class T> class gml::BBox3< T >

A template class for 2D boundary box.

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

- gmlbbox3.h

8.3 Browser Class Reference

3D navigation helper class

`#include <gmlbrowser.h>`

Public Types

- enum **NAV_MODE** { **NAV_TRACKBALL**, **NAV_WALK** }

Public Member Functions

- **Browser** (Viewport &viewport)
- void **SetNavigationMode** (NAV_MODE nav_mode)
- ModeBrowser * **GetModeBrowser** ()
- void **SetDomain** (const BBox3f &scene_size, bool update_position=true)
- BBox3f **GetDomain** () const
- void **MouseButton** (const Vector2i &pos, int button, bool pressed)
- void **MouseMotion** (const Vector2i &pos, int button)
- void **MousePassiveMotion** (const Vector2i &pos)
- void **MouseWheel** (const Vector2i &pos, int delta)
- bool **KeyDown** (int key, int options)
- bool **KeyUp** (int key, int options)

8.3.1 Detailed Description

3D navigation helper class

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

- gmlbrowser.h
- gmlbrowser.cpp

8.4 BSphere3 Class Template Reference

Template class for 3D bounding sphere.

`#include <gmlbsphere3.h>`

Miscellanious methods

- void Include (const TVector3< T > &point)
  Expand the sphere to include given point.
- void Include (const BSphere3 &bsphere)
  Expand the sphere to include given sphere.
- T radius
  Radius of the sphere.
- TVector3< T > center
  Center point.

Public Member Functions

Constructors

- BSphere3 ()
  default constructor
- BSphere3 (const TVector3< T > &center0, const double radius0=0)
  initialize class members by defined values
- BSphere3 (const BBox3< T > &bbox)
  construct sphere from given bounding box

8.4.1 Detailed Description

template<class T> class gml::BSphere3< T >

Template class for 3D bounding sphere.

Parameters:
T - template type of TVector elements

Warning:
Use it with some SIGNED integral type, or a real type

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

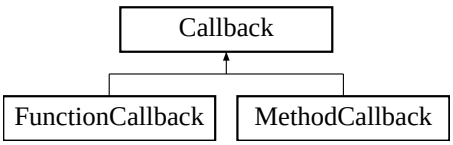
- gmlbsphere3.h

8.5 Callback Class Reference

Base class for callback with no arguments and no return value.

`#include <gmlcallback.h>`

Inheritance diagram for Callback::



Public Member Functions

- virtual void **Call** ()=0
- void Call () const

  Callback method.

8.5.1 Detailed Description

Base class for callback with no arguments and no return value.

8.5.2 Member Function Documentation

8.5.2.1 void Call () const [inline]

Callback method.

This method will be called for the callback with the appropriate arguments and must be defined by a derived class.

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

- gmlcallback.h

8.6 Camera Class Reference

Class representing 3d virtual camera.

`#include <gmlcamera.h>`

Public Member Functions

- void SetViewAngle (float view_angle)
  set camera view angle (angle should be between 0 and 180 degrees)
- float GetViewAngle () const
  return camera view angle
- void SetDistanceToTarget (float target_distance)
  set distance from camera to target
- float GetDistanceToTarget () const
  return distance from camera to target
- const Matrix4x4d & GetTransform () const
  set view transformation
- void SetTransform (const Matrix4x4d &view_transform)
  return view transformation
- Vector3d GetViewVector () const
  get view vector of camera (normalized)
- Vector3d GetUpVector () const
  get up vector of camera (normalized)
- Vector3d GetRightVector () const
  get up vector of camera (normalized)
- Vector3d **GetObserver** () const
- void **SetObserver** (const Vector3d &obs)
- Matrix4x4d **GetRotation** () const
- void **SetRotation** (const Matrix4x4d &rot)

Protected Attributes

- Matrix4x4d m_view_transform
  OGL ready viewing transform (global -> eye).
- Matrix4x4d m_inv_view_transform
  inverse view transform (eye -> global)
- float m_view_angle

view angle (degrees)

- float m_target_distance

  distance to target

8.6.1 Detailed Description

Class representing 3d virtual camera.

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

- gmlcamera.h

8.7 CIniFile Class Reference

Handling of INI-files.

`#include <gmlinifile.h>`

Public Member Functions

- bool **IsSuchKeyInFile** (const char *key)
- bool **IsSuchNameInThisKey** (const char *key, const char *name)
- CIniFile ()

  Default constructor.

- CIniFile (const char *inipath)

  Constructor, can specify pathname here instead of using SetPath later.

- virtual ~CIniFile ()

  Default destructor.

- void SetPath (const char *newpath)

  Sets path of ini file to read and write from.

- bool ReadFile ()

  Reads ini file specified using CIniFile::SetPath().

- void WriteFile ()

  Writes data stored in class to ini file.

- void Reset ()

  Deletes all stored ini data.

- int GetNumKeys ()

  number of keys currently in the ini

- int GetNumValues (const char *keyname)

  number of values stored for specified key

- std::string GetValue (const char *keyname, const char *valuename)

  gets value of [keyname] valuename =

- void GetValue (const char *keyname, const char *valuename, char *out_pcBuf, int in_iMaxChar)

  gets value of [keyname] valuename =

- int GetValueI (const char *keyname, const char *valuename)

  gets value of [keyname] valuename =

- double GetValueF (const char *keyname, const char *valuename)

  gets value of [keyname] valuename =

- bool SetValue (const char *key, const char *valuename, const char *value, bool create=1)

sets value of [keyname] valuename =.

- bool SetValueI (const char *key, const char *valuename, int value, bool create=1)
  sets value of [keyname] valuename =.
- bool SetValueF (const char *key, const char *valuename, double value, bool create=1)
  sets value of [keyname] valuename =.
- bool DeleteValue (const char *keyname, const char *valuename)
  deletes specified value
- bool DeleteKey (const char *keyname)
  deletes specified key and all values contained within

Public Attributes

- std::string error
  will contain error info if one occurs ended up not using much, just in ReadFile and GetValue

8.7.1 Detailed Description

Handling of INI-files.

8.7.2 Member Function Documentation

8.7.2.1 bool ReadFile ()

Reads ini file specified using CIniFile::SetPath().

Returns:
true if successful, false otherwise

8.7.2.2 std::string GetValue (const char * *keyname*, const char * *valuename*)

gets value of [keyname] valuename =

Returns:
"", or 0 if key/value not found. Sets error member to show problem

8.7.2.3 void GetValue (const char * *keyname*, const char * *valuename*, char * *out_pcBuf*, int *in_iMaxChar*)

gets value of [keyname] valuename =

Returns:
"", or 0 if key/value not found. Sets error member to show problem

8.7.2.4 int GetValueI (const char * *keyname*, const char * *valuename*)

gets value of [keyname] valuename =

Returns:
"", or 0 if key/value not found. Sets error member to show problem

8.7.2.5 double GetValueF (const char * *keyname*, const char * *valuename*)

gets value of [keyname] valuename =

Returns:
"", or 0 if key/value not found. Sets error member to show problem

8.7.2.6 bool SetValue (const char * *key*, const char * *valuename*, const char * *value*, bool *create* = 1)

sets value of [keyname] valuename =.

specify the optional paramter as false (0) if you do not want it to create the key if it doesn't exist.

Returns:
true if data entered, false otherwise

8.7.2.7 bool SetValueI (const char * *key*, const char * *valuename*, int *value*, bool *create* = 1)

sets value of [keyname] valuename =.

specify the optional paramter as false (0) if you do not want it to create the key if it doesn't exist.

Returns:
true if data entered, false otherwise

8.7.2.8 bool SetValueF (const char * *key*, const char * *valuename*, double *value*, bool *create* = 1)

sets value of [keyname] valuename =.

specify the optional paramter as false (0) if you do not want it to create the key if it doesn't exist.

Returns:
true if data entered, false otherwise

8.7.2.9 bool DeleteValue (const char * *keyname*, const char * *valuename*)

deletes specified value

Returns:
true if value existed and deleted, false otherwise

8.7.2.10 bool DeleteKey (const char * *keyname*)

deletes specified key and all values contained within

Returns:
returns true if key existed and deleted, false otherwise

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

- gmlinifile.h
- gmlinifile.cpp

8.8 DIntelImage Class Reference

Intel libraries IplImage wrap class.

`#include <IntelImage.h>`

Public Member Functions

- void DeleteImages ()
  Free image memory (called automatically).
- bool CheckImage (int width, int height, int in_iChannels)
  Check if image matches the specified size and channels number.
- IplImage * GetImage ()
  Get pointer to stored image.
- int width ()
  Get width of the image.
- int height ()
  Get height of the image.
- int step ()
  Get image aligned width.
- unsigned char * GetLineFast (int in_iLine)
  Get pointer to line.
- unsigned char * GetLineSafe (int in_iLine)
  Get pointer to line.

Creation

- DIntelImage ()
  Default constructor.
- void CreateImage (int width, int height, int depth=IPL_DEPTH_8U, int nChannels=3)
  Create an empty image with specified size, bit depth and channels number.
- void AttachImage (IplImage *in_pImage)
  Attach IplImage. The image is replaced by in_pImage, memory management will be done by DIntelImage from now on.
- void CloneImage (IplImage *in_pImage)
  Create a full copy of the in_pImage.
- void CopyOf (DIntelImage &in_Image, int desired_color=-1)
  Create a copy of the given DIntelImage with possibility of color <-> grayscale transfer.

Relation to SimpleImage

- gml::SimpleImage * **CopyToSimpleImage** ()

Destruction

- IplImage * DetachImage ()
  Detach IplImage. The detached IplImage should be destroyed by the programmer.
- virtual ~DIntelImage ()
  Destructor.

Filework

- bool LoadImage (const char *in_sFileName)
  Load image from file.
- void SaveImage (const char *in_sFileName)
  Save image to file.

Image drawing (actually, painting) in MFC

- void DrawImage (CDC *in_pDC, CPoint in_vOrigin, float in_dRatio)
  Paint image in MFC device context with specified offset and scale.
- void DrawImage (CDC *in_pDC, CPoint in_vOrigin, int in_iHeight=-1)
  Paint image in MFC device context with specified offset and destination image height.
- void DrawImage (CDC *in_pDC, gml::BBox2i in_SrcBox, gml::BBox2i in_TrgBox)
  Paint part of the image in MFC device context in specified window.

Image drawing (actually, painting) in Borland Object Window Library

- void DrawImage (Graphics::TBitmap *in_pTarget, TPoint in_vOrigin, float in_dRatio=1)
  Paint image in Borland OWL TBitmap with specified offset and scale.

Protected Member Functions

Forbidden operations

- DIntelImage (DIntelImage &)
  Copy constructor (not implemented).
- DIntelImage operator= (DIntelImage &)
  Assignment operator (not implemented).

Protected Attributes

- IplImage * m_pImage

  Pointer to stored IplImage.

8.8.1 Detailed Description

Intel libraries IplImage wrap class.

8.8.2 Member Function Documentation

8.8.2.1 void CopyOf (DIntelImage & *in_Image*, int *desired_color* = -1)

Create a copy of the given DIntelImage with possibility of color <-> grayscale transfer.

Parameters:

in_Image image to copy;

desired_color contols the color properties of the resulting image (-1 - do not change color/grayscale properties, 0 - the image becomes grayscale, 1 - image becomes color).

8.8.2.2 bool LoadImage (const char * *in_sFileName*)

Load image from file.

Load image from a specified filename. Returns true in case of success, false on error. Depends on the highgui.dll and (c)vlgrfmts.dll installed on your machine (both libraries are provided with Intel OpenCV), but usually jpeg, bmp, tiff, pxm are supported.

8.8.2.3 void SaveImage (const char * *in_sFileName*)

Save image to file.

Save image to a specified filename. The image format is taken from the file extension (jpg, bmp, tif, pxm). Depends on the highgui.dll and (c)vlgrfmts.dll installed on your machine (both libraries are provided with Intel OpenCV), but usually jpeg, bmp, tiff, pxm are supported.

8.8.2.4 void DrawImage (CDC * *in_pDC*, CPoint *in_vOrigin*, float *in_dRatio*)

Paint image in MFC device context with specified offset and scale.

Function is available only if GML_USE_MFC is defined at the time of compilation. The image is scaled to the given ratio and painted in the CDC with the specified offset

8.8.2.5 void DrawImage (CDC * *in_pDC*, CPoint *in_vOrigin*, int *in_iHeight* = -1)

Paint image in MFC device context with specified offset and destination image height.

Function is available only if GML_USE_MFC is defined at the time of compilation. The image is scaled to fit the given height (aspect ratio is preserved) and painted in the CDC with the specified offset

8.8.2.6 void DrawImage (CDC * *in_pDC*, gml::BBox2i *in_SrcBox*, gml::BBox2i *in_TrgBox*)

Paint part of the image in MFC device context in specified window.

Function is available only if GML_USE_MFC is defined at the time of compilation. The image is scaled to fit the given window (aspect ratio is preserved)

8.8.2.7 void DrawImage (Graphics::TBitmap * *in_pTarget*, TPoint *in_vOrigin*, float *in_dRatio* = 1)

Paint image in Borland OWL TBitmap with specified offset and scale.

Function is available only if GML_USE_OWL is defined at the time of compilation. The image is scaled to the given ratio and painted in the TBitmap with the specified offset

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

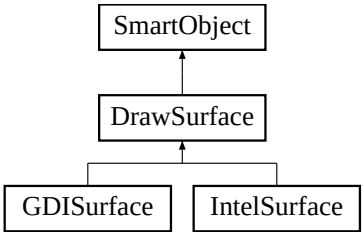
- IntelImage.h
- IntelImage.cpp

8.9 DrawSurface Class Reference

Generic image class.

`#include <gmldrawsurface.h>`

Inheritance diagram for DrawSurface::



Public Member Functions

- DrawSurface ()

  Constructor..

- virtual ~DrawSurface ()

  Empty destructor.

- virtual void **DrawLine** (int in_iX0, int in_iY0, int in_iX1, int in_iY1, Color3ub in_Color=Color3ub(255, 255, 255))=0
- virtual void **DrawLine** (gml::Vector2i in_vStart, gml::Vector2i in_vEnd, Color3ub in_Color=Color3ub(255, 255, 255))=0
- virtual void **FrameRect** (int in_iX0, int in_iY0, int in_iX1, int in_iY1, Color3ub in_Color=Color3ub(255, 255, 255))=0
- virtual void **FrameRect** (gml::Vector2i in_vMin, gml::Vector2i in_vMax, Color3ub in_Color=Color3ub(255, 255, 255))=0
- virtual void **Circle** (int in_iX0, int in_iY0, int in_iRadius, Color3ub in_Color=Color3ub(255, 255, 255))=0
- virtual void **Circle** (gml::Vector2i in_vCenter, int in_iRadius, Color3ub in_Color=Color3ub(255, 255, 255))=0
- virtual void **PolyLine** (gml::Vector2i *in_pVectors, int in_iPoints, Color3ub in_Color=Color3ub(255, 255, 255))=0
- virtual void **PolyGon** (gml::Vector2i *in_pVectors, int in_iPoints, Color3ub in_Color=Color3ub(255, 255, 255))=0

8.9.1 Detailed Description

Generic image class.

Some functionality should be defined in children

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

- gmldrawsurface.h

8.10 File Class Reference

A class for working with files (a wrapped to FILE ∗).

`#include <gmlfile.h>`

Methods

- bool Open (const char ∗mode)

  Opening of file according to mode. The list of allowed modes:.

- bool Opened () const

  True if file was previously opened.

- bool Close ()

  Close the file.

- bool **ReadStr** (String &out_buff, int len_buff, bool ∗out_nl=NULL)
- bool **ReadStr** (char ∗out_buff, int len_buff, bool ∗out_nl=NULL)
- bool **ReadStr** (String &out_buff)
- void **Read** (BYTE ∗out_buff, int len_buff, int &out_size)
- void **WriteStr** (const String &str)
- void **WriteStr** (const char ∗out_buff,...)
- void **Write** (const BYTE ∗buff, int size)
- void **Flush** ()
- bool **Remove** ()
- bool **Rename** (const PathString &new_file_name)
- bool **IsError** ()
- PathString **PathName** () const
- int **Printf** (const char ∗format,...)
- bool **Seek** (long offset)
- bool **SeekCur** (long offset)
- bool **SeekEnd** (long offset)
- long FilePos () const

  Gets current file offset.

- long FileSize () const

  Returns the size of the file in bytes.

- bool **Copy** (const PathString &name_from, const PathString &name_to)
- enum { **MODE_TEXT**, **MODE_BINARY** }
- enum { **MODE_READ**, **MODE_WRITE**, **MODE_READ_WRITE** }
- FILE ∗ **fd**
- int **mode_tb**
- int **mode_rw**

Public Member Functions

Constructors

- File (const PathString &full_pathname)
  Sets a full path to the file only.
- File (const File &file)
  Copy constructor.
- ~File ()
  Destructor. File will be closed if it was opened.

Protected Types

8.10.1 Detailed Description

A class for working with files (a wrapped to FILE *).

8.10.2 Constructor & Destructor Documentation

8.10.2.1 File (const File & *file*)

Copy constructor.

File may not be opened. Debug version asserts if this condition is violated.

8.10.3 Member Function Documentation

8.10.3.1 bool Open (const char * *mode*)

Opening of file according to mode. The list of allowed modes:.

Parameters:

mode - i/o mode of the file to open

- "r" - open for reading only; the file must exist.
- "w" - create a new file or open an existing file for writing.
- "a" - open for writing at the end-of-file or create for writing if the file does not exist.
- "r+" - open an existing file for reading and writing.
- "w+" - open an empty file for reading and writing.
- "a+" - open for reading and appending; create a file if it does not exist.

Note:

Files are open in text mode as a default. Character 'b' may be appended to the above modes to open file in binary mode ('b' may also precede '+' sign). Path to file must be specified. Debug version asserts if this condition is violated.

8.10.3.2 bool Close ()

Close the file.

Returns:

- true (if there were no problems when working with the file and the file was closed successfully).
- false (either I/O error while working with file or closing error).

Note:

File must be opened. Debug version asserts if this condition is violated

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

- gmlfile.h
- gmlfile.cpp

8.11 FileFinder Class Reference

`#include <gmlfilefinder.h>`

Public Member Functions

- bool FindFiles (const char *filefilter="*", const gml::PathString &="", int recursionlevel=-1)
- gml::PathString **GetStartDir** ()
- std::vector< gml::PathString > & **GetCompleteFileList** ()

Protected Member Functions

- **FileFinder** (const FileFinder &right)
- const FileFinder & **operator=** (const FileFinder &right)

8.11.1 Detailed Description

A class which enumerate all the files and directories under specified start directory

Todo
: add directory structure handling

8.11.2 Member Function Documentation

8.11.2.1 bool FindFiles (const char * *filefilter* = "*", const gml::PathString & = "", int *recursionlevel* = -1)

search the startdir for files with a specified filename or extension filter. startdir == NULL => from the current directory recursion level == -1 => all sub-levels

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

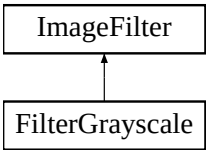
- gmlfilefinder.h

8.12 FilterGrayscale Class Reference

Creates a grayscale version of a color bitmap.

`#include <gmlfiltergrayscale.h>`

Inheritance diagram for FilterGrayscale::



Public Types

- enum METHOD {
  M_R, **M_G**, **M_B**, **M_AVERAGE**,
  M_EYE }
  method to convert from RGB to LUM: take RED, GREEN, BLUE, or simple average color, or use "human eye" formula

Public Member Functions

- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const
  Applies the Filter to pBmpSource and stores the result in pBmpDest.
- void **SetMethod** (METHOD m)

8.12.1 Detailed Description

Creates a grayscale version of a color bitmap.

8.12.2 Member Function Documentation

8.12.2.1 virtual void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented from ImageFilter.

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

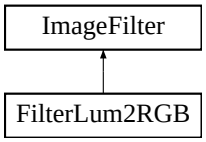
- gmlfiltergrayscale.h

8.13 FilterLum2RGB Class Reference

Creates a Lum2RGB version of a color bitmap.

`#include <gmlfilterlum2rgb.h>`

Inheritance diagram for FilterLum2RGB::



Public Member Functions

- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const
  Applies the Filter to pBmpSource and stores the result in pBmpDest.

8.13.1 Detailed Description

Creates a Lum2RGB version of a color bitmap.

8.13.2 Member Function Documentation

8.13.2.1 virtual void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented from ImageFilter.

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

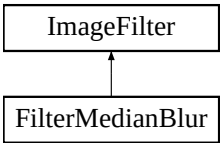
- gmlfilterlum2rgb.h

8.14 FilterMedianBlur Class Reference

Creates a MedianBlur version of a color bitmap.

`#include <gmlfiltermedian.h>`

Inheritance diagram for FilterMedianBlur::



Public Member Functions

- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const
  Applies the Filter to pBmpSource and stores the result in pBmpDest.
- virtual void **SetRadius** (int r)

8.14.1 Detailed Description

Creates a MedianBlur version of a color bitmap.

8.14.2 Member Function Documentation

8.14.2.1 virtual void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented from ImageFilter.

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

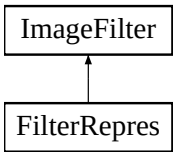
- gmlfiltermedian.h

8.15 FilterRepres Class Reference

Transferres between different representations.

`#include <gmlfilterrepres.h>`

Inheritance diagram for FilterRepres::



Public Member Functions

- **FilterRepres** (Image::REPRES in_Repres=Image::R_BYTE, double in_dCoeff=1, bool in_bClip-TypeBounds=true)
- void **SetParam** (Image::REPRES in_Repres=Image::R_BYTE, double in_dCoeff=1, bool in_bClip-TypeBounds=true)
- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const

  Applies the Filter to pBmpSource and stores the result in pBmpDest.

8.15.1 Detailed Description

Transferres between different representations.

8.15.2 Member Function Documentation

8.15.2.1 virtual void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented from ImageFilter.

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

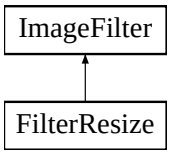
- gmlfilterrepres.h

8.16 FilterResize Class Reference

Creates a MedianBlur version of a color bitmap.

`#include <gmlfilterresize.h>`

Inheritance diagram for FilterResize::



Public Types

- enum **RESIZE_TYPE** { **FASTEST** = 1, **BILINEAR** }

Public Member Functions

- **FilterResize** (int in_iDstWdt, int in_iDstHgt, RESIZE_TYPE in_Type)
- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const

  Applies the Filter to pBmpSource and stores the result in pBmpDest.

- virtual void **SetResizeParam** (int in_iDstWdt, int in_iDstHgt, RESIZE_TYPE in_Type)

8.16.1 Detailed Description

Creates a MedianBlur version of a color bitmap.

8.16.2 Member Function Documentation

8.16.2.1 virtual void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented from ImageFilter.

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

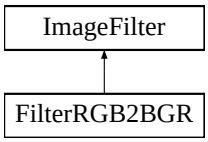
- gmlfilterresize.h

8.17 FilterRGB2BGR Class Reference

Creates a Lum2RGB version of a color bitmap.

`#include <gmlfilterrgb2bgr.h>`

Inheritance diagram for FilterRGB2BGR::



Public Member Functions

- virtual void ApplyInPlace (Image *pBmp) const
  In-Place Apply.

8.17.1 Detailed Description

Creates a Lum2RGB version of a color bitmap.

8.17.2 Member Function Documentation

8.17.2.1 virtual void ApplyInPlace (Image * *pBmp*) const [virtual]

In-Place Apply.

Applies the filter to pBmp. The base-class version copies the bitmap after calling Apply (pBmp, pTempBmp).

Reimplemented from ImageFilter.

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

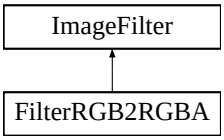
- gmlfilterrgb2bgr.h

8.18 FilterRGB2RGBA Class Reference

Creates a grayscale version of a color bitmap.

`#include <gmlfilterrgb2rgba.h>`

Inheritance diagram for FilterRGB2RGBA::



Public Member Functions

- **FilterRGB2RGBA** (const gml::Color3ub &in_Color=gml::Color3ub(255, 255, 255))
- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const

  Applies the Filter to pBmpSource and stores the result in pBmpDest.

- void **SetTransparentColor** (const gml::Color3ub &in_Color)

8.18.1 Detailed Description

Creates a grayscale version of a color bitmap.

8.18.2 Member Function Documentation

8.18.2.1 void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented from ImageFilter.

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

- gmlfilterrgb2rgba.h
- gmlfilterrgb2rgba.cpp

8.19 FrameCounter Class Reference

Simple frame rate (FPS) counter.

`#include <gmlframecounter.h>`

Public Member Functions

- FrameCounter ()
  constructor
- ~FrameCounter ()
  destructor
- void SetUpdateSpeed (double ms)
  Set counted update speed in milliseconds.
- double GetUpdateSpeed () const
  Return counter update speed.
- void SetMinimumFrameCount (int count)
  Set minimum number of frames needed to calculate FPS.
- double GetMinimumFrameCount () const
  Return counter update speed.
- void Reset ()
  Signal that animation cycle is started.
- void FrameStarted ()
  Signal that current frame is to be drawn.
- void FrameFinished ()
  Signal that current frame is finished.
- bool Updated () const
  Tells whether FPS has been updated.
- double GetFrameCount () const
  Return the framecount (frames per second).
- double GetMeanFrameTime () const
  Return the mean frame time (msec).

8.19.1 Detailed Description

Simple frame rate (FPS) counter.

8.19.2 Member Function Documentation

8.19.2.1 void SetMinimumFrameCount (int *count*) [inline]

Set minimum number of frames needed to calculate FPS.

FPS will be made and Updated() flag will be set only if time since latest update > update speed and number of drawn frames (calls to FrameFinished()) > minimum number of frames

8.19.2.2 void Reset ()

Signal that animation cycle is started.

Probably should be called before first frame

8.19.2.3 void FrameStarted ()

Signal that current frame is to be drawn.

Should be called before each scene display

8.19.2.4 void FrameFinished ()

Signal that current frame is finished.

Should be called after each scene display

8.19.2.5 bool Updated () const

Tells whether FPS has been updated.

Check this after call to FrameFinished. Flag cleared after next call to FrameFinished

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

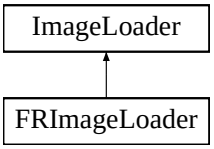
- gmlframecounter.h
- gmlframecounter.cpp

8.20 FRImageLoader Class Reference

Image loader based on FreeImage third-party library.

`#include <gmlfrimageloader.h>`

Inheritance diagram for FRImageLoader::



Public Member Functions

- virtual bool **EnumerateLoadableFormats** (void(∗f)(IMAGE_FORMAT f))
- virtual bool **EnumerateSaveableFormats** (void(∗f)(IMAGE_FORMAT f))
- virtual IMAGE_FORMAT **CheckBitMapFile** (const std::string &path)
- virtual bool **LoadBitMap** (const std::string &path, Image &out_bitmap, IMAGE_FORMAT format=FORMAT_UNKNOWN, int flags=0)
- virtual bool **SaveBitMap** (const std::string &path, const Image &bitmap, IMAGE_FORMAT format, int flags=0)

8.20.1 Detailed Description

Image loader based on FreeImage third-party library.

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

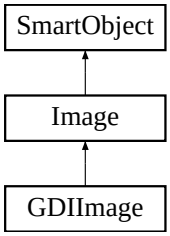
- gmlfrimageloader.h
- gmlfrimageloader.cpp

8.21 GDIImage Class Reference

Manipulates uncompressed device- and platform-independent bitmaps.

`#include <gmlgdiimage.h>`

Inheritance diagram for GDIImage::



Public Member Functions

- GDIImage ()
  Creates an empty bitmap.
- virtual ~GDIImage ()
  Destroys the bitmap.
- GDIImage (const Image &Orig)
  Copy constructor.
- GDIImage (const GDIImage &Orig)
  Copy constructor.
- GDIImage & operator= (const Image &Orig)
  Assignment operator.
- GDIImage & **operator=** (const GDIImage &Orig)
- long GetMemUsed ()
  Returns the amount of memory used by the object.
- long GetBytesPerLine () const
  Returns number of bytes used per line.
- const BYTE * **GetRawData** () const
- BYTE * GetRawData ()
  Access to raw data array. Warning!!! Dangerous function! Use with care.
- void PaintImage (HDC in_hDC, DWORD dwRop=SRCCOPY)
  Paint image.
- void PaintImage (HDC in_hDC, POINT in_ptDstOrg, DWORD dwRop=SRCCOPY)
  Paint image at a specified offset.

- void PaintImage (HDC in_hDC, POINT in_ptDstOrg, SIZE in_DstSize, DWORD dwRop=SRCCOPY)

  Fit image to specified rectangle.

- void PaintImageRect (HDC in_hDC, POINT in_ptDstOrg, RECT in_SrcRect, SIZE in_DstSize, DWORD dwRop=SRCCOPY)

  Fit image fragment to specified rectangle.

- virtual gml::DrawSurface * GetDrawSurface ()

  Draw surface.

- virtual bool FastResizeImage (double in_dScale, gml::GDIImage &out_ResImg)

  Fast GDI image resize.

Protected Member Functions

- virtual void **ConstructorInitLocals** ()
- virtual bool InternalCreate (int Width, int Height, FORMAT format, REPRES repres, ORIENT orient=O_BOTTOMLEFT)
- virtual void FreeMembers ()

  Delete memory allocated by member variables.

- virtual void InitLineArray ()

  Initialize internal table of line addresses.

- virtual void InternalChangeOrientation (ORIENT new_orient)
- virtual void **InternalPaint** (HDC in_hDC, int in_iDstOrgX, int in_iDstOrgY, int in_iDstWdt, int in_iDstHgt, int in_iSrcOrgX, int in_iSrcOrgY, int in_iSrcWdt, int in_iSrcHgt, DWORD dwRop)

Static Protected Member Functions

- long **GetBytesPerLine** (int width, FORMAT f, REPRES r)

Protected Attributes

- BYTE * m_pBits

  Pointer to the bits.

- HDC m_hDC

  Internal device context.

- BITMAPINFO m_BMInfo

  Bitmap info structure.

- RGBQUAD m_pPalette [255]

  Palette for 1-channel images.

- RGBQUAD * **m_pColorTable**

- HBITMAP m_hBMP

  BITMAP handel.

- HGDIOBJ m_hOldHandle

  Neded for cleanup.

- int m_iBytesPerLine

  Bytes per line in the created DIB.

Friends

- class **GDISurface**

8.21.1 Detailed Description

Manipulates uncompressed device- and platform-independent bitmaps.

The data is stored sequentially without padding in the bitmap. The class implements exactly the interface defined by gml::Image without additions.

8.21.2 Member Function Documentation

8.21.2.1 virtual bool InternalCreate (int *Width*, int *Height*, FORMAT *format*, REPRES *repres*, ORIENT *orient* = O_BOTTOMLEFT) [protected, virtual]

Create a new bitmap with uninitialized bits. (Assume no memory is allocated yet.)

Implements Image.

8.21.2.2 virtual void InternalChangeOrientation (ORIENT *new_orient*) [protected, virtual]

Change image orientation to given state

Warning:
you should set m_orient variable after actual operation

Implements Image.

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

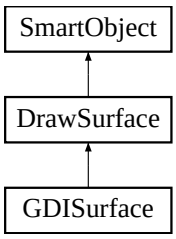
- gmlgdiimage.h

8.22 GDISurface Class Reference

Generic image class.

`#include <gmlgdisurface.h>`

Inheritance diagram for GDISurface::



Public Member Functions

- GDISurface (gml::GDIImage *in_pImage)

  Constructor..

- virtual ~GDISurface ()

  Empty destructor.

- void **DrawLine** (int in_iX0, int in_iY0, int in_iX1, int in_iY1, Color3ub in_Color=Color3ub(255, 255, 255))
- void **DrawLine** (gml::Vector2i in_vStart, gml::Vector2i in_vEnd, Color3ub in_Color=Color3ub(255, 255, 255))
- void **FrameRect** (int in_iX0, int in_iY0, int in_iX1, int in_iY1, Color3ub in_Color=Color3ub(255, 255, 255))
- void **FrameRect** (gml::Vector2i in_vMin, gml::Vector2i in_vMax, Color3ub in_Color=Color3ub(255, 255, 255))
- void **Circle** (int in_iX0, int in_iY0, int in_iRadius, Color3ub in_Color=Color3ub(255, 255, 255))
- void **Circle** (gml::Vector2i in_vCenter, int in_iRadius, Color3ub in_Color=Color3ub(255, 255, 255))

- void **PolyLine** (gml::Vector2i *in_pVectors, int in_iPoints, Color3ub in_Color=Color3ub(255, 255, 255))
- void **PolyGon** (gml::Vector2i *in_pVectors, int in_iPoints, Color3ub in_Color=Color3ub(255, 255, 255))

Protected Member Functions

- DWORD **GetColor** (gml::Color3ub in_Color)

Protected Attributes

- GDIImage * **m_pImage**

8.22.1 Detailed Description

Generic image class.

Some functionality should be defined in children

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

- gmlgdisurface.h

8.23 GLRC Class Reference

OpenGL context handling

Attention:
Include windows.h before using this file! Usage:

```
GLRC glrc;
// inside window creation
glrc.Create(wnd);
// before rendering into window
glrc.MakeCurrent();
// ...
// display in the screen (in case of double buffering)
glrc.SwapBuffers();
// ..
// delete gl context (may be avoided, because called in destructor anyway)
glrc.Destroy();
```

.

`#include <gmlglrc.h>`

Public Member Functions

Class Construction/Destruction

- GLRC (HWND wnd)

  Constructor from Window Handle (HWND).

- GLRC (HDC hdc)

  Constructor form Device Context Handle(HDC).

- void Destroy ()

  Destory GL context (also called in destructor).

- virtual ~GLRC ()

Context members

- virtual bool Create ()

  Member, needed to create context. Call this before use of OpenGL.

- bool IsCurrent () const

  Member, that return true if opengl is ready to draw.

- bool MakeCurrent ()

  Prepare current context for use.

- void SwapBuffers ()

  Call this insted of opengl's swap buffer, at the end of the drawing.

Protected Attributes

- bool m_created

  true inside Create() / Destory() pair

- HWND m_wnd

  Window handle.

- HDC m_dc

  Device Context handle.

- HGLRC m_glrc

  OpenGL Context handle.

8.23.1 Detailed Description

OpenGL context handling

Attention:

Include windows.h before using this file! Usage:

```
GLRC glrc;
// inside window creation
glrc.Create(wnd);
// before rendering into window
glrc.MakeCurrent();
// ...
// display in the screen (in case of double buffering)
glrc.SwapBuffers();
// ..
// delete gl context (may be avoided, because called in destructor anyway)
glrc.Destroy();
```

.

Todo

Add possibility to select pixel format (double/single buffer, stencil, etc)

8.23.2 Constructor & Destructor Documentation

8.23.2.1 virtual ~GLRC () [virtual]

Virtual Destructor

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

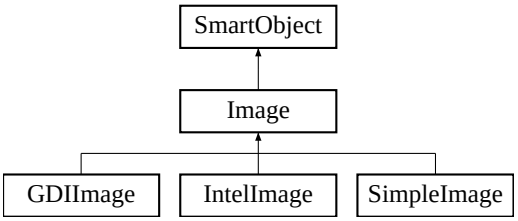
- gmlglrc.h

8.24 Image Class Reference

Generic image class.

`#include <gmlimage.h>`

Inheritance diagram for Image::



Public Types

- enum **FORMAT** { **F_L**, **F_RGB**, **F_RGBA**, **F_BGR** }
- enum **REPRES** {
  R_BYTE, **R_WORD**, **R_DWORD**, **R_FLOAT**,
  R_DOUBLE }
- enum ORIENT { O_TOPLEFT, O_BOTTOMLEFT }
  line order in the image

Public Member Functions

- Image ()
- virtual ~Image ()
  Empty destructor.
- **Image** (Image &in_Image)
- Image & operator= (Image const &Orig)
- bool const operator== (Image const &Other)
- virtual void Create (int Width, int Height, FORMAT format, REPRES repres, ORIENT orient=O_-BOTTOMLEFT)
- void CreateCopy (const Image &rSrImage)
  Creates a copy of rSrImage, converting color depth if nessesary.
- bool CreateFromDIB (struct tagBITMAPINFO *in_pDIB, BYTE *in_pcData=NULL)
  Creates from a bitmap header.
- void CreateFilteredCopy (const Image &rSrImage, const ImageFilter &rFilter)
- void **CreateComposition** (const std::vector< Image * > &src, const ImageComposer &comp)
- bool **Created** () const
- void **Clear** ()
- void ApplyFilter (const ImageFilter &filter)
  Applies a filter to the bitmap.

- void Decompose (const ImageDecomposer &decomposer, std::vector< Image * > &out_res)

  Applies a decompositor to the bitmap.

- bool ChangeOrientation (ORIENT new_orient)

  Change orientation of the image to new one.

- bool ChangeFormat (FORMAT new_format)

  Change image format.

- bool ChangeRepres (REPRES new_repres)

  Change image pixel representetion.

- int **GetWidth** () const
- int **GetHeight** () const
- FORMAT **GetFormat** () const
- REPRES **GetRepres** () const
- ORIENT **GetOrient** () const
- virtual long **GetMemUsed** ()=0
- int **GetElemSize** () const
- virtual long GetBytesPerLine () const =0

  Returns number of bytes used per line.

- BYTE ** GetLineArray () const

  Returns pointer to an array containing the starting addresses of the bitmap lines.

- virtual void Lock (bool bReadable, bool bWriteable)
- virtual void Unlock ()

  Unlocks the Bitmap surface.

- bool **IsLocked** () const
- bool **AlmostEqual** (const Image &Bmp, int epsilon) const
- void **PaintImage** (CDC *in_pDC, CPoint dst_org, CSize dst_size, DWORD dwRop=SRCCOPY)
- virtual gml::DrawSurface * **GetDrawSurface** ()=0

Static Public Member Functions

- int **Format2Channels** (FORMAT f)
- int **Repres2Size** (REPRES r)
- int **ElemSize** (FORMAT f, REPRES r)

Protected Member Functions

- virtual bool InternalCreate (int Width, int Height, FORMAT format, REPRES repres, ORIENT orient=O_BOTTOMLEFT)=0
- virtual void FreeMembers ()=0

  Delete memory allocated by member variables.

- virtual void InitLineArray ()=0

Initialize internal table of line addresses.

- void InternalCopy (const Image &rSrImage)
- void InitLocals (int width, int height, FORMAT format, REPRES repres, ORIENT orient=O_-BOTTOMLEFT)

  Can be called from internalCreate() to initialize object state.

- virtual void InternalChangeOrientation (ORIENT new_orient)=0

Protected Attributes

- int **m_width**
- int **m_height**
- bool **m_created**
- FORMAT **m_format**
- REPRES **m_repres**
- ORIENT m_orient

  current orientation order

- BYTE ** m_line_array

  Table of the starting addresses of the lines.

- int m_lock_count

  Number of times the bitmap was locked.

8.24.1 Detailed Description

Generic image class.

Some functionality should be defined in children

8.24.2 Member Enumeration Documentation

8.24.2.1 enum ORIENT

line order in the image

Enumeration values:

O_TOPLEFT lines are going left-right, top-bottom

O_BOTTOMLEFT lines are left-right, bottom-top

8.24.3 Constructor & Destructor Documentation

8.24.3.1 Image ()

Empty constructor. Constructors in derived classes create a small empty bitmap to ensure that the object is always in a sane state.

8.24.4 Member Function Documentation

8.24.4.1 bool const operator== (Image const & *Other*)

Test for equality. This function actually tests every pixel, so it's not fast. It's meant mainly for use in asserts and such.

8.24.4.2 virtual void Create (int *Width*, int *Height*, FORMAT *format*, REPRES *repres*, ORIENT *orient* = O_BOTTOMLEFT) [virtual]

Creates a new empty bitmap. Memory for the bits is allocated but not initialized. Previous contents of the bitmap object are discarded

8.24.4.3 bool CreateFromDIB (struct tagBITMAPINFO * *in_pDIB*, BYTE * *in_pcData* = NULL)

Creates from a bitmap header.

Parameters:

in_pDIB Óêàçàòåëü íà BITMAPINFO (õ240àíèëèùå ñâîéñòâ DIB'à)

in_pcData Óêàçàòåëü íà ïèêñåëüíûå bitmap äàííûå. Åñëè óñòàíîâëåí â NULL - ñ247èòàåòñÿ, 247òî äàííûå 240àñîîëãà254òñÿ ñ240àçó çà BITMAPINFO

8.24.4.4 void CreateFilteredCopy (const Image & *rSrImage*, const ImageFilter & *rFilter*)

Creates a copy of rSrImage, applying rFilter on the way. Depending on the filter called, this is often much faster than CreateCopy() followed by ApplyFilter().

8.24.4.5 bool ChangeOrientation (ORIENT *new_orient*)

Change orientation of the image to new one.

Note:

if it already set, nothing will changed

8.24.4.6 bool ChangeFormat (FORMAT *new_format*)

Change image format.

Note:

if it already set, nothing will changed, currently implemented only for RGB <-> BGR change

8.24.4.7 bool ChangeRepres (REPRES *new_repres*)

Change image pixel representetion.

Note:

if it already set, nothing will changed

8.24.4.8 virtual void Lock (bool *bReadable*, bool *bWriteable*) [virtual]

Locks bitmap. GetLineArray() and other direct-access methods should only be called if the bitmap is locked. Lock and Unlock keep a lock count.

8.24.4.9 virtual bool InternalCreate (int *Width*, int *Height*, FORMAT *format*, REPRES *repres*, ORIENT *orient* = O_BOTTOMLEFT) [protected, pure virtual]

Create a new bitmap with uninitialized bits. (Assume no memory is allocated yet.)

Implemented in GDIImage, IntelImage, and SimpleImage.

8.24.4.10 void InternalCopy (const Image & *rSrImage*) [protected]

Creates a new Image as copy of rSrImage. Assumes there is no memory allocated yet.

8.24.4.11 virtual void InternalChangeOrientation (ORIENT *new_orient*) [protected, pure virtual]

Change image orientation to given state

Warning:
you should set m_orient variable after actual operation

Implemented in GDIImage, IntelImage, and SimpleImage.

8.24.5 Member Data Documentation

8.24.5.1 int m_lock_count [protected]

Number of times the bitmap was locked.

Default is m_LockCount always >= 1, so access to bits is always possible.

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

- gmlimage.h

8.25 ImageComposer Class Reference

Produce single image from several reference images.

`#include <gmlimagecomposer.h>`

Public Member Functions

- virtual void Apply (const std::vector< Image * > &source, Image *pBmpDest) const

8.25.1 Detailed Description

Produce single image from several reference images.

8.25.2 Member Function Documentation

8.25.2.1 virtual void Apply (const std::vector< Image * > & *source*, Image * *pBmpDest*) const [virtual]

Applies the Composer to *(pBmpSource[0]).. *(pBmpSource[n_images - 1]) and stores the result in p-BmpDest.

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

- gmlimagecomposer.h

8.26 ImageDecomposer Class Reference

Decompose an image into several destination images.

`#include <gmlimagecomposer.h>`

Public Member Functions

- virtual void Apply (Image *pBmpSource, std::vector< Image * > &out_dest) const
  Decompase pBmpSource to several images which are stored in out_dest.

8.26.1 Detailed Description

Decompose an image into several destination images.

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

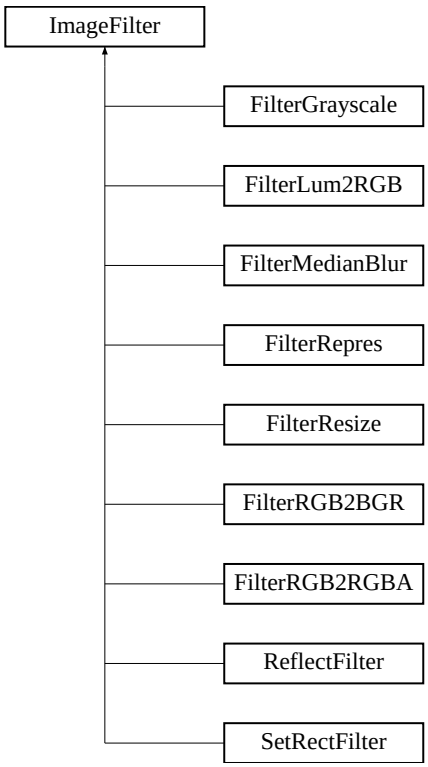
- gmlimagecomposer.h

8.27 ImageFilter Class Reference

Interface for generic image modifier.

`#include <gmlimagefilter.h>`

Inheritance diagram for ImageFilter::



Public Member Functions

- virtual void ApplyInPlace (Image *pBmp) const
  In-Place Apply.
- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const
  Applies the Filter to pBmpSource and stores the result in pBmpDest.

8.27.1 Detailed Description

Interface for generic image modifier.

8.27.2 Member Function Documentation

8.27.2.1 virtual void ApplyInPlace (Image * *pBmp*) const `[virtual]`

In-Place Apply.

Applies the filter to pBmp. The base-class version copies the bitmap after calling Apply (pBmp, pTempBmp).

Reimplemented in FilterRGB2BGR.

8.27.2.2 virtual void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented in FilterGrayscale, FilterLum2RGB, FilterMedianBlur, FilterRepres, FilterResize, FilterRGB2RGBA, ReflectFilter, and SetRectFilter.

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

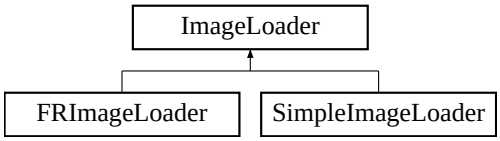
- gmlimagefilter.h

8.28 ImageLoader Class Reference

Loading and saving bitmaps to files (base class).

`#include <gmlimageloader.h>`

Inheritance diagram for ImageLoader::



Public Types

- enum **IMAGE_FORMAT** {
  FORMAT_BMP, **FORMAT_ICO**, **FORMAT_JPEG**, **FORMAT_PNG**,
  FORMAT_PGM, **FORMAT_PGMRAW**, **FORMAT_PPM**, **FORMAT_TIFF**,
  FORMAT_TARGA, **FORMAT_UNKNOWN** }

Public Member Functions

- virtual bool **EnumerateLoadableFormats** (void(∗f)(IMAGE_FORMAT f))=0
- virtual bool **EnumerateSaveableFormats** (void(∗f)(IMAGE_FORMAT f))=0
- virtual IMAGE_FORMAT **CheckBitMapFile** (const std::string &path)=0
- virtual bool **LoadBitMap** (const std::string &path, Image &out_bitmap, IMAGE_FORMAT format=FORMAT_UNKNOWN, int flags=0)=0
- virtual bool **SaveBitMap** (const std::string &path, const Image &bitmap, IMAGE_FORMAT format=FORMAT_UNKNOWN, int in_iQuality=100, int flags=0)=0

Static Public Member Functions

- std::string GetFormatExtension (IMAGE_FORMAT format)
  Get file extension for particular image file format.

8.28.1 Detailed Description

Loading and saving bitmaps to files (base class).

8.28.2 Member Function Documentation

8.28.2.1 std::string GetFormatExtension (IMAGE_FORMAT *format*) `[static]`

Get file extension for particular image file format.

Parameters:
Image format

Returns:
A string containing required extension

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

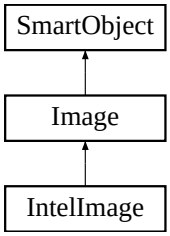
- gmlimageloader.h

8.29 IntelImage Class Reference

Manipulates uncompressed device- and platform-independent bitmaps.

`#include <gmlintelimage.h>`

Inheritance diagram for IntelImage::



Public Member Functions

- IntelImage ()
  Creates an empty bitmap.
- virtual ~IntelImage ()
  Destroys the bitmap.
- IntelImage (const Image &Orig)
  Copy constructor.
- IntelImage (const IntelImage &Orig)
  Copy constructor.
- IntelImage & operator= (const Image &Orig)
  Assignment operator.
- IntelImage & **operator=** (const IntelImage &Orig)
- long GetMemUsed ()
  Returns the amount of memory used by the object.
- long GetBytesPerLine () const
  Returns number of bytes used per line.
- const BYTE * **GetRawData** () const
- BYTE * GetRawData ()
  Access to raw data array. Dangerous function! Use with care.
- _IplImage * GetImage ()
  Access to intel image structure! Use with care.
- virtual gml::DrawSurface * GetDrawSurface ()
  Draw surface.

Protected Member Functions

- virtual void **ConstructorInitLocals** ()
- virtual bool InternalCreate (int Width, int Height, FORMAT format, REPRES repres, ORIENT orient=O_BOTTOMLEFT)
- virtual void FreeMembers ()

  Delete memory allocated by member variables.

- virtual void InitLineArray ()

  Initialize internal table of line addresses.

- virtual void InternalChangeOrientation (ORIENT new_orient)

Protected Attributes

- _IplImage * m_pImage

  Pointer to stored IplImage.

- BYTE * **m_pBits**
- int **m_iBytesPerLine**

8.29.1 Detailed Description

Manipulates uncompressed device- and platform-independent bitmaps.

The data is stored sequentially without padding in the bitmap. The class implements exactly the interface defined by gml::Image without additions.

8.29.2 Member Function Documentation

8.29.2.1 virtual bool InternalCreate (int *Width*, int *Height*, FORMAT *format*, REPRES *repres*, ORIENT *orient* = O_BOTTOMLEFT) [protected, virtual]

Create a new bitmap with uninitialized bits. (Assume no memory is allocated yet.)

Implements Image.

8.29.2.2 virtual void InternalChangeOrientation (ORIENT *new_orient*) [protected, virtual]

Change image orientation to given state

Warning:
you should set m_orient variable after actual operation

Implements Image.

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

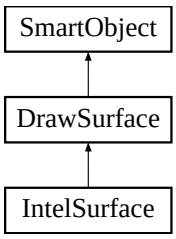
- gmlintelimage.h

8.30 IntelSurface Class Reference

Generic image class.

`#include <gmlintelsurface.h>`

Inheritance diagram for IntelSurface::



Public Member Functions

- IntelSurface (gml::IntelImage *in_pImage)

  Constructor..

- virtual ~IntelSurface ()

  Empty destructor.

- void **DrawLine** (int in_iX0, int in_iY0, int in_iX1, int in_iY1, Color3ub in_Color=Color3ub(255, 255, 255))
- void **DrawLine** (gml::Vector2i in_vStart, gml::Vector2i in_vEnd, Color3ub in_Color=Color3ub(255, 255, 255))
- void **FrameRect** (int in_iX0, int in_iY0, int in_iX1, int in_iY1, Color3ub in_Color=Color3ub(255, 255, 255))
- void **FrameRect** (gml::Vector2i in_vMin, gml::Vector2i in_vMax, Color3ub in_Color=Color3ub(255, 255, 255))
- void **Circle** (int in_iX0, int in_iY0, int in_iRadius, Color3ub in_Color=Color3ub(255, 255, 255))
- void **Circle** (gml::Vector2i in_vCenter, int in_iRadius, Color3ub in_Color=Color3ub(255, 255, 255))

- void **PolyLine** (gml::Vector2i *in_pVectors, int in_iPoints, Color3ub in_Color=Color3ub(255, 255, 255))
- void **PolyGon** (gml::Vector2i *in_pVectors, int in_iPoints, Color3ub in_Color=Color3ub(255, 255, 255))

Protected Member Functions

- int **GetColor** (gml::Color3ub in_Color)

Protected Attributes

- IntelImage * **m_pImage**

8.30.1 Detailed Description

Generic image class.

Some functionality should be defined in children

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

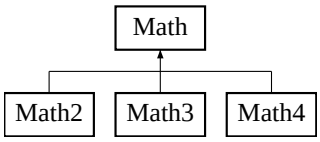
- gmlintelsurface.h

8.31 Math Class Template Reference

A template class for precision related staff.

`#include <gmlmath.h>`

Inheritance diagram for Math::



Static Public Member Functions

- bool AboutZero (const double v, const double tolerance)

  Comparisons with given tolerance.

- bool **AboutZero** (const double v)
- bool **NearZero** (const double v)
- bool **AboutEqual** (const double v1, const double v2, const double tolerance)
- bool **AboutEqual** (const double v1, const double v2)
- bool **NearEqual** (const double v1, const double v2)
- int SignAbout (const double v, const double tolerance)

  Determine sing of value with some tolerance around zero.

- int **SignAbout** (const double v)
- int **SignNear** (const double v)

Static Public Attributes

- const double TOLERANCE = 1.0E-05f

  default tolerance (type dependent)

- const double MIN_VALUE = numeric_limits<float>::min()

  Maximum values for float and double types (rounded down).

- const double MAX_VALUE = numeric_limits<float>::max()

  Minimal positive values for float and double types (rounded up).

- const double EPSILON = numeric_limits<float>::epsilon()

  minimum positive floating point number x such that 1.0 + x != 1.0

8.31.1 Detailed Description

template<class T> class gml::Math< T >

A template class for precision related staff.

Parameters:
T is a real type like float, double

Note:
use typedefs MathD, MathF

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

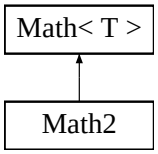
- gmlmath.h
- gmlmath.cpp

8.32 Math2 Class Template Reference

Comparison of 2D vectors with a tolerance.

`#include <gmlvector2.h>`

Inheritance diagram for Math2::



Static Public Member Functions

- bool **AboutZero** (const TVector2< T > &v, const double tolerance)
- bool **AboutZero** (const TVector2< T > &v)
- bool **NearZero** (const TVector2< T > &v)
- bool **AboutEqual** (const TVector2< T > &v1, const TVector2< T > &v2, const double tolerance)
- bool **AboutEqual** (const TVector2< T > &v1, const TVector2< T > &v2)
- bool **NearEqual** (const TVector2< T > &v1, const TVector2< T > &v2)

8.32.1 Detailed Description

template<class T> class gml::Math2< T >

Comparison of 2D vectors with a tolerance.

Parameters:
T - some REAL type (float, double)

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

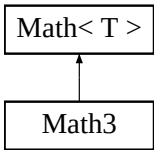
- gmlvector2.h

8.33 Math3 Class Template Reference

Comparison of 2D vectors with a tolerance.

`#include <gmlvector3.h>`

Inheritance diagram for Math3::



Static Public Member Functions

- bool **AboutZero** (const TVector3< T > &v, const double tolerance)
- bool **AboutZero** (const TVector3< T > &v)
- bool **NearZero** (const TVector3< T > &v)
- bool **AboutEqual** (const TVector3< T > &v1, const TVector3< T > &v2, const double tolerance)
- bool **AboutEqual** (const TVector3< T > &v1, const TVector3< T > &v2)
- bool **NearEqual** (const TVector3< T > &v1, const TVector3< T > &v2)

8.33.1 Detailed Description

template<class T> class gml::Math3< T >

Comparison of 2D vectors with a tolerance.

Parameters:

T - some REAL type (float, double)

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

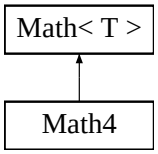
- gmlvector3.h

8.34 Math4 Class Template Reference

Comparison of 4D vectors with a tolerance.

`#include <gmlvector4.h>`

Inheritance diagram for Math4::



Static Public Member Functions

- bool **AboutZero** (const TVector4< T > &v, const double tolerance)
- bool **AboutZero** (const TVector4< T > &v)
- bool **NearZero** (const TVector4< T > &v)
- bool **AboutEqual** (const TVector4< T > &v1, const TVector4< T > &v2, const double tolerance)
- bool **AboutEqual** (const TVector4< T > &v1, const TVector4< T > &v2)
- bool **NearEqual** (const TVector4< T > &v1, const TVector4< T > &v2)

8.34.1 Detailed Description

template<class T> class gml::Math4< T >

Comparison of 4D vectors with a tolerance.

Parameters:
T - some REAL type (float, double)

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

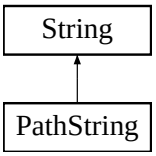
- gmlvector4.h

8.35 PathString Class Reference

Processing of the path to file.

`#include <gmlpathstring.h>`

Inheritance diagram for PathString::



Public Member Functions

Constructors

- **PathString** (const string &rhs)
- **PathString** (const string &rhs, size_type pos, size_type n)
- **PathString** (const char *s, size_type n)
- **PathString** (const char *s)
- **PathString** (size_type n, char c)
- **PathString** (const_iterator first, const_iterator last)
- PathString (const string &path, const string &filename, const string &extension=string())

  Constructs an object from path+filename+extension.

Path handling

- string Path () const

  Returns extracted path from full path to the file.

- string Filename () const

  Returns full name of the file (with extension).

- string Basename () const

  Returns base name of the file (without extension).

- string Extension () const

  Returns extension of the file (dot is not included).

- void Split (string *path, string *filename, string *extension=NULL) const

  Splits path name into path, base filename and extension.

- bool MakeFullPath (const PathString &relative_to=PathString())

File or directory handling

- bool IsFile () const

Check: 'this' string is the name of the real plain file.

- bool IsDir () const

  Check: 'this' string is the name of the directory.

- bool CurDir ()

  Get current working directory to 'this' pathname.

- bool ChDir () const

  Change current working directory to 'this' pathname.

- bool MkDir () const

  Create directory with name specified by 'this' pathname.

- bool RmDir () const

  Remove directory with name specified by 'this' pathname.

- bool ExePath ()

  Get full path to the current program.

- bool ListFileName (list< PathString > &filelist, PathString *mask) const

Static Public Attributes

- const char PATH_SEPARATOR = '\\'

  char path separator for currents OS

- const string PATH_SEPARATOR_STR = string("\\")

  string path separator for current OS

Protected Member Functions

- bool **CheckMask** (const PathString &mask) const
- void **CorrectSeparators** ()

8.35.1 Detailed Description

Processing of the path to file.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 PathString (const string & *path*, const string & *filename*, const string & *extension* = string())

Constructs an object from path+filename+extension.

The path separator will be added if string path does not end by path separator. The dot will be added between filename and extension if extension does not begin by dot.

Note:
If parameter extension is absent then filename contains the full name of the file.

8.35.3 Member Function Documentation

8.35.3.1 string Path () const

Returns extracted path from full path to the file.

Note:
In case if path cannot be extracted, an empty string will be returned.

8.35.3.2 string Filename () const

Returns full name of the file (with extension).

Note:
If full name is not extracted, an empty string will be returned.

8.35.3.3 string Basename () const

Returns base name of the file (without extension).

Note:
If base name is not extracted, an empty string will be returned.

8.35.3.4 string Extension () const

Returns extension of the file (dot is not included).

Note:
If extension is not extracted, an empty string will be returned.

8.35.3.5 void Split (string * *path*, string * *filename*, string * *extension* = NULL) const

Splits path name into path, base filename and extension.

Note:
If parameter extension is absent then filename will contain file name with extension.

8.35.3.6 bool MakeFullPath (const PathString & *relative_to* = PathString())

Expand current path to full path relative to 'relative_to' argument.

Warning:
relative_to must be a full path or NULL if to work with current directory

8.35.3.7 bool RmDir () const

Remove directory with name specified by 'this' pathname.

Note:
Directory must be empty

8.35.3.8 bool ListFileName (list< PathString > & *filelist*, PathString * *mask*) const

Non-obligatory parameter mask defines mask for the filtration.

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

- gmlpathstring.h
- gmlpathstring.cpp

8.36 Ref Class Template Reference

Smart pointer class Smart pointers provide automatic memory management, so that programmer never cares on freeing memory. Smart pointers mechanism frees an object when there are no any references to it. Smart pointers should be used instead of standard C pointers:

```
Object* obj; // standard pointer, should be deleted after use
gml::Ref<Object> obj; // smart pointer frees memory when necessary
```

Attention:

Smart pointers works only with objects that support AddRef() and Release() operations. It is assumed that object keeps a reference count (a number of smapt pointers pointing to it). This reference count is incremented in AddRef() command and decremented in Release(). Also Release() performs check whether reference count is 0 and calls 'delete this'. GML provides special base class for this: gml::SmartObject. If you want to use smart pointers either derive your object from this class or provide own implementation of AddRef() and Release() methods. 1) initialization Smart pointers are initiliazed just like the standard pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
```

Smart pointers must never be defined like this: gml::Rev<Object>* obj; !!! 2) use of smart pointers Smart pointers are dereferenced like normal pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
obj->SomeMethod();
(*obj).SomeMethod();
```

3) smart pointers as parameters Smart pointers are implicitly casted to normal pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
Object* obj1 =  obj;
```

This operation is dangerous unless used properly. You should use it like follows. Smart pointers may be passed to functions and returned from functions like normal pointers.

```
...
Object* SomeMethod(Object* obj);
...
gml::Ref<Object> obj =  new Object(1,2,3);
gml::Ref<Object> obj1 = SomeMethod(obj);
```

The general rule is this: **Use ONLY smart pointers when storing objects. Use normals pointers when passing smart pointers as parameters or return values** So the following is forbidden:

```
class SomeClass
  {
  private:
    Object* obj;
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      } // at this point obj will be deleted, and m_obj will be undefined!!
  };
```

Correct use:

```
class SomeClass
  {
  private:
```

```
  gml::Ref<Object> obj;
public:
  void CreateObject()
    {
    gml::Ref<Object> obj = new Object(1,2,3);
    m_obj = obj;
    }
};
```

.

`#include <gmlref.h>`

Public Member Functions

- Ref (T *real_ptr=NULL)

  Default contructor.

- Ref (const Ref &rhs)

  Copy constructor.

- ~Ref ()

  Destructor.

- Ref & operator= (const Ref &rhs)

  Overloaded operator =.

- T * operator -> () const

  Overloaded operator.

- T & operator * () const

  Overloaded operator.

- int IsSet () const

  Whether this pointer is not null.

- operator T * () const

  Implicit conversion to C pointer.

- void Set (T *real_ptr)

  Forced assignment of C pointer to this reference

  Attention:
  Danger function, use it carefully.

- void Clear ()

  Cleans this reference.

Related Functions

(Note that these are not member functions.)

- Ref< newT > & ConvRef (const Ref< oldT > &old_ref)
  Casting & convertion of smart references.

8.36.1 Detailed Description

template<class T> class gml::Ref< T >

Smart pointer class Smart pointers provide automatic memory management, so that programmer never cares on freeing memory. Smart pointers mechanism frees an object when there are no any references to it. Smart pointers should be used instead of standard C pointers:

```
Object* obj; // standard pointer, should be deleted after use
gml::Ref<Object> obj; // smart pointer frees memory when necessary
```

Attention:

Smart pointers works only with objects that support AddRef() and Release() operations. It is assumed that object keeps a reference count (a number of smapt pointers pointing to it). This reference count is incremented in AddRef() command and decremented in Release(). Also Release() performs check whether reference count is 0 and calls 'delete this'. GML provides special base class for this: gml::SmartObject. If you want to use smart pointers either derive your object from this class or provide own implementation of AddRef() and Release() methods. 1) initialization Smart pointers are initiliazed just like the standard pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
```

Smart pointers must never be defined like this: gml::Rev<Object>* obj; !!! 2) use of smart pointers Smart pointers are dereferenced like normal pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
obj->SomeMethod();
(*obj).SomeMethod();
```

3) smart pointers as parameters Smart pointers are implicitly casted to normal pointers:

```
gml::Ref<Object> obj =  new Object(1,2,3);
Object* obj1 =  obj;
```

This operation is dangerous unless used properly. You should use it like follows. Smart pointers may be passed to functions and returned from functions like normal pointers.

```
...
Object* SomeMethod(Object* obj);
...
gml::Ref<Object> obj =  new Object(1,2,3);
gml::Ref<Object> obj1 = SomeMethod(obj);
```

The general rule is this: **Use ONLY smart pointers when storing objects. Use normals pointers when passing smart pointers as parameters or return values** So the following is forbidden:

```
class SomeClass
  {
  private:
    Object* obj;
```

```
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      } // at this point obj will be deleted, and m_obj will be undefined!!
  };
```

Correct use:

```
class SomeClass
  {
  private:
    gml::Ref<Object> obj;
  public:
    void CreateObject()
      {
      gml::Ref<Object> obj = new Object(1,2,3);
      m_obj = obj;
      }
  };
```

.

Note:

Nevertheless, nobody restricts you from passing smart pointers as parameters and return values. Simple pointers are just have more natural interface. 4) casting of smart pointers Since smart pointers are implicitly casted to C pointers, casting rules are the same as for C pointers. The only issue appears when use need to cast from one smart pointer to another. In this case you need to use gml::ConvRef() operation:

```
gml::Ref<BaseObject> b;
gml::Ref<DerivedObject> c;
c = b; // this may not work in compilers which nor support template casting methods
c = gml::ConvRef<DerivedObject>(b); // OK
```

Attention:

This simple example not working (!) :

```
Object* CreateObject()
  {
  gml::Ref<Object> obj = new Object;
  return obj; // at this point reference count of obj is 1, so the obj is deleted
              // before returned from the function!
  }
```

Correct use is one of the following:

```
Object* CreateObject()
  {
  Object* obj = new Object;
  return obj;
  }

gml::Ref<Object> CreateObject()
  {
  gml::Ref<Object> obj = new Object;
  return obj;
  }
```

Warning:

It is forbidden to assign arrays created by new[] to smart pointers

```
gml::Ref<Object> obj = new Object[10]; // result is undefined!!!!!
```

All smart object should be create by new operator! The following is **strictly forbidden** :

```
Object obj;
gml::Ref<Object> = &obj;
```

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

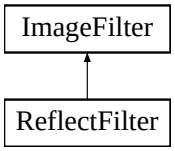
- gmlref.h

8.37 ReflectFilter Class Reference

Creates a grayscale version of a color bitmap.

`#include <gmlreflectfilter.h>`

Inheritance diagram for ReflectFilter::



Public Member Functions

- **ReflectFilter** (bool in_bVert, bool in_bHor)
- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const

  Applies the Filter to pBmpSource and stores the result in pBmpDest.

8.37.1 Detailed Description

Creates a grayscale version of a color bitmap.

8.37.2 Member Function Documentation

8.37.2.1 virtual void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented from ImageFilter.

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

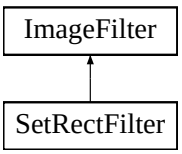
- gmlreflectfilter.h

8.38 SetRectFilter Class Reference

Creates a grayscale version of a color bitmap.

`#include <gmlsetrectfilter.h>`

Inheritance diagram for SetRectFilter::



Public Member Functions

- **SetRectFilter** (double in_dValue)
- **SetRectFilter** (double in_dValue, gml::Vector2i in_vMin, gml::Vector2i in_vMax)
- **SetRectFilter** (double in_dValue, int in_iMinX, int in_iMinY, int in_iMaxX, int in_iMaxY)
- void **SetParam** (double in_dValue)
- void **SetParam** (double in_dValue, gml::Vector2i in_vMin, gml::Vector2i in_vMax)
- void **SetParam** (double in_dValue, int in_iMinX, int in_iMinY, int in_iMaxX, int in_iMaxY)
- virtual void Apply (const Image *pBmpSource, Image *pBmpDest) const

  Applies the Filter to pBmpSource and stores the result in pBmpDest.

8.38.1 Detailed Description

Creates a grayscale version of a color bitmap.

8.38.2 Member Function Documentation

8.38.2.1 virtual void Apply (const Image * *pBmpSource*, Image * *pBmpDest*) const [virtual]

Applies the Filter to pBmpSource and stores the result in pBmpDest.

The base-class version copies the bitmap before calling ApplyInPlace (pBmpDest).

Reimplemented from ImageFilter.

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

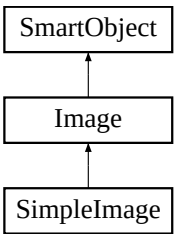
- gmlsetrectfilter.h

8.39 SimpleImage Class Reference

Manipulates uncompressed device- and platform-independent bitmaps.

`#include <gmlsimpleimage.h>`

Inheritance diagram for SimpleImage::



Public Member Functions

- SimpleImage ()

  Creates an empty bitmap.

- virtual ~SimpleImage ()

  Destroys the bitmap.

- SimpleImage (const Image &Orig)

  Copy constructor.

- SimpleImage (const SimpleImage &Orig)

  Copy constructor.

- SimpleImage & operator= (const Image &Orig)

  Assignment operator.

- SimpleImage & **operator=** (const SimpleImage &Orig)
- long GetMemUsed ()

  Returns the amount of memory used by the object.

- long GetBytesPerLine () const

  Returns number of bytes used per line.

- const BYTE * **GetRawData** () const
- BYTE * GetRawData ()

  Access to raw data array. Dangerous function! Use with care.

- virtual int **GetRawDataSize** () const
- virtual gml::DrawSurface * GetDrawSurface ()

  Draw surface.

Static Public Member Functions

- long GetMemNeeded (int width, int height, FORMAT f, REPRES r)

  Returns memory needed by a bitmap with the specified attributes.

- long GetBitsMemNeeded (int width, int height, FORMAT f, REPRES r)

  Returns memory needed by bitmap bits.

Protected Member Functions

- virtual void **ConstructorInitLocals** ()
- virtual bool InternalCreate (int Width, int Height, FORMAT format, REPRES repres, ORIENT orient=O_BOTTOMLEFT)
- virtual void FreeMembers ()

  Delete memory allocated by member variables.

- virtual void InitLineArray ()

  Initialize internal table of line addresses.

- virtual void InternalChangeOrientation (ORIENT new_orient)

Protected Attributes

- BYTE * m_pBits

  Pointer to the bits.

8.39.1 Detailed Description

Manipulates uncompressed device- and platform-independent bitmaps.

The data is stored sequentially without padding in the bitmap. The class implements exactly the interface defined by gml::Image without additions.

8.39.2 Member Function Documentation

8.39.2.1 virtual bool InternalCreate (int *Width*, int *Height*, FORMAT *format*, REPRES *repres*, ORIENT *orient* = O_BOTTOMLEFT) [protected, virtual]

Create a new bitmap with uninitialized bits. (Assume no memory is allocated yet.)

Implements Image.

8.39.2.2 virtual void InternalChangeOrientation (ORIENT *new_orient*) [protected, virtual]

Change image orientation to given state

Warning:
you should set m_orient variable after actual operation

Implements Image.

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

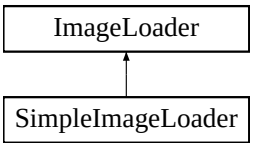
- gmlsimpleimage.h

8.40 SimpleImageLoader Class Reference

Image loader based on FreeImage third-party library.

`#include <gmlsimpleimageloader.h>`

Inheritance diagram for SimpleImageLoader::



Public Member Functions

- virtual bool **EnumerateLoadableFormats** (void(∗f)(IMAGE_FORMAT f))
- virtual bool **EnumerateSaveableFormats** (void(∗f)(IMAGE_FORMAT f))
- virtual IMAGE_FORMAT **CheckBitMapFile** (const std::string &path)
- virtual bool **LoadBitMap** (const std::string &path, Image &out_bitmap, IMAGE_FORMAT format=FORMAT_UNKNOWN, int flags=0)
- virtual bool **SaveBitMap** (const std::string &path, const Image &bitmap, IMAGE_FORMAT format=FORMAT_UNKNOWN, int in_iQuality=100, int flags=0)
- virtual void **GetExifMarker** (std::vector< unsigned char > &out_pcData)
- virtual void **SetExifMarker** (std::vector< unsigned char > &in_pcData)

8.40.1 Detailed Description

Image loader based on FreeImage third-party library.

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

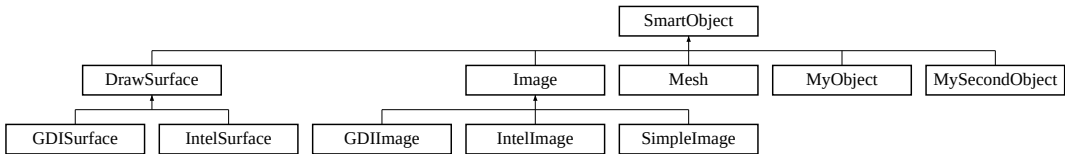
- gmlsimpleimageloader.h
- gmlbmp.cpp
- gmlsimpleimageloader.cpp

8.41 SmartObject Class Reference

Base class for objects with support for reference counting.

`#include <gmlsmartobject.h>`

Inheritance diagram for SmartObject::



Public Member Functions

- void AddRef ()

  Increment reference count.

- void Release ()

  Decrement reference count.

- virtual ~SmartObject ()

  Destructor is virtual.

8.41.1 Detailed Description

Base class for objects with support for reference counting.

See also:
gml::Ref

8.41.2 Member Function Documentation

8.41.2.1 void AddRef () [inline]

Increment reference count.

Warning:
This function is used internally, you should never call it

8.41.2.2 void Release () [inline]

Decrement reference count.

Warning:
This function is used internally, you should never call it

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

- gmlsmartobject.h

8.42 Socket Class Reference

A wrapper for Win32 stream socket.

`#include <gmlsocket.h>`

Public Member Functions

- bool **Create** (int nSocketPort=0, const char *lpszSocketAddress=NULL)
- bool **Bind** (int nSocketPort, const char *lpszSocketAddress=NULL)
- void **Close** ()
- bool **Connect** (const char *lpszHostAddress, int nHostPort)
- bool **Accept** (gml::Socket &rConnectedSocket)
- bool **Listen** (int nConnectionBacklog=5)
- int **Receive** (void *lpBuf, int nBufLen, int nFlags=0)
- int **Send** (const void *lpBuf, int nBufLen, int nFlags=0)
- bool **SendFile** (const std::string &out_filename)
- bool **ReceiveFile** (const std::string in_filename)

Static Public Member Functions

- bool InitSockets ()

  Initilize Win32 sockets (MUST be called before use of sockets).

8.42.1 Detailed Description

A wrapper for Win32 stream socket.

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

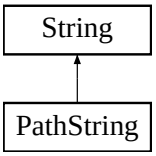
- gmlsocket.h
- gmlsocket.cpp

8.43 String Class Reference

Standard string with some extensions.

`#include <gmlstring.h>`

Inheritance diagram for String::



Public Member Functions

Standard constructors (the same as in std::string)

- String ()
  Empty string.
- String (const string &rhs)
  Copy constructor.
- String (const string &rhs, size_type pos, size_type n)
  Constructor from substring.
- String (const char *s, size_type n)
  From N characters of C string.
- String (const char *s)
  From C string.
- String (size_type n, char c)
  A sequence of n chars.
- String (const_iterator first, const_iterator last)
  two iterators point to a some string of list of chars

MFC, VCL - specific functions (use defines to enable them)

- CString AsCString () const
  Conversion to MFC string.

Utility functions

- int AsInt () const
  To integer.

- double AsDouble () const
  To double.
- void Format (const char *format,...)
  write a formatted string and a variable list of arguments to this object
- void FormatV (const char *format, va_list arglist)
  write a formatted string and a variable list of arguments to this object

8.43.1 Detailed Description

Standard string with some extensions.

Note:
use it instead of std::string

Warning:
If you wish to make classes derived from String (or std::string) please be careful NOT no provide own destructor. This is so because the base class std::string has no virtual destructor and in case of addressing via pointer to the base class destructor of derived class will not be called!

8.43.2 Constructor & Destructor Documentation

8.43.2.1 String (const string & *rhs*, size_type *pos*, size_type *n*) [inline]

Constructor from substring.

Parameters:
rhs is a source string

pos is a start position

n is number of characters

8.43.3 Member Function Documentation

8.43.3.1 CString AsCString () const [inline]

Conversion to MFC string.

Note:
works only if GML_USE_MFC is defined

8.43.3.2 int AsInt () const [inline]

To integer.

Returns:
return 0 if the input cannot be converted to a value of that type

8.43.3.3 double AsDouble () const [inline]

To double.

Returns:
return 0.0 if the input cannot be converted to a value of that type

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

- gmlstring.h
- gmlstring.cpp

8.44 TColor3 Class Template Reference

3 Color repsentation

`#include <gmlcolor.h>`

Public Member Functions

- TColor3 ()

  no initialization

- **TColor3** (T r0, T g0, T b0)
- TColor3< T > & **operator** *= (const double d)
- TColor3< T > **operator** * (const double d) const
- TColor3< T > & **operator+=** (const TColor3< T > &u)
- TColor3< T > & **operator-=** (const TColor3< T > &u)
- TColor3< T > **operator+** (const TColor3< T > &u) const
- TColor3< T > **operator-** (const TColor3< T > &u) const
- TColor3< T > & **operator/=** (const double d)
- TColor3< T > **operator/** (const double d) const
- operator T * ()

  Convert a TColor3 to array of (3) elements.

- operator const T * () const

  Convert a TColor3 to array of (3) elements.

Static Public Member Functions

- const TColor3< T > & Cast (const T *u)

  Treat array of (3) elements as a TColor3.

- TColor3< T > & Cast (T *u)

  Treat array of (3) elements as a TColor3.

Related Functions

(Note that these are not member functions.)

- TColor3< T > operator * (const double d, const TColor3< T > &u)

  Multiplication of scalar d by TColor3 u.

8.44.1 Detailed Description

template<class T> class gml::TColor3< T >

3 Color repsentation

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

- gmlcolor.h

8.45 TColor4 Class Template Reference

4 Color repsentation

`#include <gmlcolor.h>`

Public Member Functions

- TColor4 ()

  no initialization

- **TColor4** (T r0, T g0, T b0, T w0=1)
- TColor4< T > & **operator** *= (const double d)
- TColor4< T > **operator** * (const double d) const
- TColor4< T > & **operator+=** (const TColor4< T > &u)
- TColor4< T > & **operator-=** (const TColor4< T > &u)
- TColor4< T > **operator+** (const TColor4< T > &u) const
- TColor4< T > **operator-** (const TColor4< T > &u) const
- TColor4< T > & **operator/=** (const double d)
- TColor4< T > **operator/** (const double d) const
- operator T * ()

  Convert a TColor4 to array of (4) elements.

- operator const T * () const

  Convert a TColor4 to array of (4) elements.

Static Public Member Functions

- const TColor4< T > & Cast (const T *u)

  Treat array of (4) elements as a TColor4.

- TColor4< T > & Cast (T *u)

  Treat array of (4) elements as a TColor4.

Related Functions

(Note that these are not member functions.)

- TColor4< T > operator * (const double d, const TColor4< T > &u)

  Multiplication of scalar d by TColor4 u.

8.45.1 Detailed Description

template<class T> class gml::TColor4< T >

4 Color repsentation

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

- gmlcolor.h

8.46 TMatrix2x2 Class Template Reference

Template class for 2x2 matrix.

`#include <gmlmatrix2.h>`

Public Member Functions

- T **Det** ()
- TMatrix2x2< T > **MultVectStr** (TVector2< T > &v1, TVector2< T > &v2)

Constructors

- TMatrix2x2 ()
  No initialization.
- TMatrix2x2 (T r)
  Initialization by diagonal element.
- TMatrix2x2 (const T *m)
  Initialization by array of T.
- TMatrix2x2 (T a00, T a01, T a10, T a11)
  Direct initialization by components (row order).

Access to values

- T * RawArray ()
  Provide access to internal contents.
- void GetValue (T *mp) const
  Copy this matrix mp *array (! column-major order).*
- const T * GetValue () const
  Provide const access to the buffer.
- void SetValue (const T *mp)
  Copy data from mp *array.*
- void SetValue (T r)
  Initialization by a diagonal component.
- T & operator() (int row, int col)
  access operator (e.g. m(1,0) = 20)
- const T & operator() (int row, int col) const
  const access operator (e.g. a = m(1,0))
- T & Elem (int row, int col)

Return (row, col) element.

- const T & Elem (int row, int col) const
  Return const (row, col) element.

Row and column access

- void SetRow (int r, const TVector2< T > &t)
  Set row r *of this matrix to vector* t.
- void SetColumn (int c, const TVector2< T > &t)
  Set column c *of this matrix to vector* t.
- void GetRow (int r, TVector2< T > &t) const
  Set vector t *equal to row* r *of this matrix.*
- TVector2< T > GetRow (int r) const
  Return row r.
- void GetColumn (int c, TVector2< T > &t) const
  Set vector t *equal to column* c *of this matrix.*
- TVector2< T > GetColumn (int c) const
  Return column c.

Matrix-2-Matrix operations

- TMatrix2x2< T > & MultRight (const TMatrix2x2< T > &b)
  Multiplication to matrix b *(right).*
- TMatrix2x2< T > & MultLeft (const TMatrix2x2< T > &b)
  Multiplication to matrix b *(left).*
- TMatrix2x2< T > & operator *= (const TMatrix2x2< T > &mat)
  *this *= mat*
- TMatrix2x2< T > & operator+= (const TMatrix2x2< T > &mat)
  this += mat
- TMatrix2x2< T > **operator/** (const double d)

Matrix-2-Vector operations

- void MultMatrixVec (const TVector2< T > &src, TVector2< T > &dst) const
  *dst = M * src*
- void MultMatrixVec (TVector2< T > &src_and_dst) const
  *src_and_dst = M * src_and_dst*

- void MultVecMatrix (const TVector2< T > &src, TVector2< T > &dst) const
  *dst = src * M*
- void MultVecMatrix (TVector2< T > &src_and_dst) const
  *src_and_dst = src_and_dst * M*

Matrix-2-Scalar operations

- TMatrix2x2< T > & operator *= (const T &r)
  Multiplication by scalar r (diagonal).

Miscellaneous

- void MakeIdentity ()
  Make E-matrix from this one.
- TMatrix2x2< T > Inverse () const
  Return an inverse matrix.
- TMatrix2x2< T > Transpose () const
  Return transposed matrix.

Static Public Member Functions

- const TMatrix2x2< T > & Identity ()
  Return predefined identity matrix.

Protected Attributes

- T **m** [4]

Friends

- TMatrix2x2< T > operator * (const TMatrix2x2< T > &m1, const TMatrix2x2< T > &m2)
  *Matrix multiplication (return m1 * m2).*
- bool operator== (const TMatrix2x2< T > &m1, const TMatrix2x2< T > &m2)
  Equality operator.
- bool operator!= (const TMatrix2x2< T > &m1, const TMatrix2x2< T > &m2)
  Unequality operator.

Related Functions

(Note that these are not member functions.)

- TMatrix2x2< T_TO > Conv (const TMatrix2x2< T_FROM > &u)
  Convert TMatrix2x2<T_FROM> to TMatrix2x2<T_TO>.
- TMatrix2x2< float > ConvF (const TMatrix2x2< T > &u)
  Convert TMatrix2x2<T> to TMatrix2x2<float>.
- TMatrix2x2< double > ConvD (const TMatrix2x2< T > &u)
  Convert TMatrix2x2<T> to TMatrix2x2<double>.

8.46.1 Detailed Description

template<class T> class gml::TMatrix2x2< T >

Template class for 2x2 matrix.

Parameters:
T - template type of matrix elements

Warning:
Use it with some SIGNED integral type, or a real type Elements are stored in column-major order (OpenGL-style) E.g.:

```
a0  a2
a1  a3
```

8.46.2 Member Function Documentation

8.46.2.1 T* RawArray () [inline]

Provide access to internal contents.

Warning:
Use with care, this function is dangerous!

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

- gmlmatrix2.h

8.47 TMatrix3x3 Class Template Reference

Template class for 3x3 matrix.

`#include <gmlmatrix3.h>`

Public Member Functions

- T **Det** ()
- TMatrix3x3< T > **MultVectStr** (TVector3< T > &v1, TVector3< T > &v2)

Constructors

- TMatrix3x3 ()
  No initialization.
- TMatrix3x3 (T r)
  Initialization by diagonal element.
- TMatrix3x3 (const T *m)
  Initialization by array of T.
- TMatrix3x3 (T a00, T a01, T a02, T a10, T a11, T a12, T a20, T a21, T a22)
  Direct initialization by components (row order).

Access to values

- T * RawArray ()
  Provide access to internal contents.
- void GetValue (T *mp) const
  Copy this matrix mp *array (! column-major order).*
- const T * GetValue () const
  Provide const access to the buffer.
- void SetValue (const T *mp)
  Copy data from mp *array.*
- void SetValue (T r)
  Initialization by a diagonal component.
- T & operator() (int row, int col)
  access operator (e.g. m(2,1) = 20)
- const T & operator() (int row, int col) const
  const access operator (e.g. a = m(2,1))
- T & Elem (int row, int col)

Return (row, col) element.

- const T & Elem (int row, int col) const

  Return const (row, col) element.

Row and column access

- void SetScale (T s)

  Set scale components of this matrix to the same value s.

- void SetScale (const TVector3< T > &s)

  Set scale components of this matrix to values from vector s.

- void SetTranslation (const TVector2< T > &t)

  Set translation part of this matrix to values of vector t.

- void GetTranslation (TVector2< T > &t) const

  Set vector t *to translation part of this matrix.*

- TVector2< T > GetTranslation () const

  Return translation part of this matrix.

- void SetRotation (const TMatrix3x3< T > &mat)

  Set rotation part of the matrix (a 2x2 upper left corner).

- void GetRotation (TMatrix3x3< T > &mat) const

  Set mat to rotation part of this matrix.

- TMatrix3x3< T > GetRotation () const

  Return rotation part of this matrix (a 2x2 upper left corner).

- void SetRow (int r, const TVector3< T > &t)

  Set row r *of this matrix to vector* t.

- void SetColumn (int c, const TVector3< T > &t)

  Set column c *of this matrix to vector* t.

- void GetRow (int r, TVector3< T > &t) const

  Set vector t *equal to row* r *of this matrix.*

- TVector3< T > GetRow (int r) const

  Return row r.

- void GetColumn (int c, TVector3< T > &t) const

  Set vector t *equal to column* c *of this matrix.*

- TVector3< T > GetColumn (int c) const

  Return column c.

Matrix-2-Matrix operations

- TMatrix3x3< T > & MultRight (const TMatrix3x3< T > &b)
  Multiplication to matrix b *(right).*
- TMatrix3x3< T > & MultLeft (const TMatrix3x3< T > &b)
  Multiplication to matrix b *(left).*
- TMatrix3x3< T > & operator *= (const TMatrix3x3< T > &mat)
  *this *= mat*
- TMatrix3x3< T > & operator+= (const TMatrix3x3< T > &mat)
  this += mat
- TMatrix3x3< T > & **operator/=** (const double d)
- TMatrix3x3< T > **operator** *= (const double d)
- TMatrix3x3< T > **operator/** (const double d)
- TMatrix3x3< T > **operator=** (const TMatrix3x3< T > &mat)

Matrix-2-Vector operations

- void MultMatrixVec (const TVector2< T > &src, TVector2< T > &dst) const
  *dst = M * src*
- void MultMatrixVec (TVector2< T > &src_and_dst) const
  *src_and_dst = M * src_and_dst*
- void MultVecMatrix (const TVector2< T > &src, TVector2< T > &dst) const
  *dst = src * M*
- void MultVecMatrix (TVector2< T > &src_and_dst) const
  *src_and_dst = src_and_dst * M*
- void MultMatrixVec (const TVector3< T > &src, TVector3< T > &dst) const
  *dst = M * src*
- void MultMatrixVec (TVector3< T > &src_and_dst) const
  *src_and_dst = M * src_and_dst*
- void MultVecMatrix (const TVector3< T > &src, TVector3< T > &dst) const
  *dst = src * M*
- void MultVecMatrix (TVector3< T > &src_and_dst) const
  *src_and_dst = src_and_dst * M*
- void MultMatrixDir (const TVector3< T > &src, TVector3< T > &dst) const
  *dst = M * src (only rotation and scale part, no translation)*
- void MultMatrixDir (TVector3< T > &src_and_dst) const
  *src_and_dst = M * src_and_dst (only rotation and scale part, no translation)*

- void MultDirMatrix (const TVector2< T > &src, TVector2< T > &dst) const
  *dst = src * M (only rotation and scale part, no translation)*
- void MultDirMatrix (TVector2< T > &src_and_dst) const
  *src_and_dst = src_and_dst * M (only rotation and scale part, no translation)*

Miscellaneous

Multiplication by scalar r (diagonal)

- void MakeIdentity ()
  Make E-matrix from this one.
- TMatrix3x3< T > Inverse () const
  Return an inverse matrix.
- TMatrix3x3< T > Transpose () const
  Return transposed matrix.

Static Public Member Functions

- const TMatrix3x3< T > & Identity ()
  Return predefined identity matrix.

Protected Attributes

- T **m** [9]

Friends

- TMatrix3x3< T > operator * (const TMatrix3x3< T > &m1, const TMatrix3x3< T > &m2)
  *Matrix multiplication (return m1 * m2).*
- bool operator== (const TMatrix3x3< T > &m1, const TMatrix3x3< T > &m2)
  Equality operator.
- bool operator!= (const TMatrix3x3< T > &m1, const TMatrix3x3< T > &m2)
  Unequality operator.

Related Functions

(Note that these are not member functions.)

- TMatrix3x3< T_TO > Conv (const TMatrix3x3< T_FROM > &u)

Convert TMatrix3x3<T_FROM> to TMatrix3x3<T_TO>.

- TMatrix3x3< float > ConvF (const TMatrix3x3< T > &u)
  Convert TMatrix3x3<T> to TMatrix3x3<float>.
- TMatrix3x3< double > ConvD (const TMatrix3x3< T > &u)
  Convert TMatrix3x3<T> to TMatrix3x3<double>.

8.47.1 Detailed Description

template<class T> class gml::TMatrix3x3< T >

Template class for 3x3 matrix.

Parameters:
T - template type of matrix elements

Warning:
Use it with some SIGNED integral type, or a real type Elements are stored in column-major order (OpenGL-style) E.g.:

```
a0  a3  a6
a1  a4  a7
a2  a5  a8
```

8.47.2 Member Function Documentation

8.47.2.1 T* RawArray () [inline]

Provide access to internal contents.

Warning:
Use with care, this function is dangerous!

8.47.2.2 void SetTranslation (const TVector2< T > & *t*) [inline]

Set translation part of this matrix to values of vector *t*.

Translation part is the 3-th column, so it is valid for use with column-vectors V = M * V; V = (v1 v2 v3 v4)T

See also:
MultMatrixVec

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

- gmlmatrix3.h

8.48 TMatrix4x4 Class Template Reference

Template class for 4x4 matrix.

`#include <gmlmatrix4.h>`

Public Member Functions

Constructors

- TMatrix4x4 ()
  No initialization.
- TMatrix4x4 (T r)
  Initialization by diagonal element.
- TMatrix4x4 (const T *m)
  Initialization by array of T.
- TMatrix4x4 (T a00, T a01, T a02, T a03, T a10, T a11, T a12, T a13, T a20, T a21, T a22, T a23, T a30, T a31, T a32, T a33)
  Direct initialization by components (row order).

Access to values

- T * RawArray ()
  Provide access to internal contents.
- void GetValue (T *mp) const
  Copy this matrix mp *array (! column-major order).*
- const T * GetValue () const
  Provide const access to the buffer.
- void SetValue (const T *mp)
  Copy data from mp *array.*
- void SetValue (T r)
  Initialization by a diagonal component.
- T & operator() (int row, int col)
  access operator (e.g. m(2,3) = 20)
- const T & operator() (int row, int col) const
  const access operator (e.g. a = m(2,3))
- T & Elem (int row, int col)
  Return (row, col) element.

- const T & Elem (int row, int col) const

  Return const (row, col) element.

Row and column access

- void SetScale (T s)

  Set scale components of this matrix to the same value s.

- void SetScale (const TVector3< T > &s)

  Set scale components of this matrix to values from vector s.

- void SetTranslation (const TVector3< T > &t)

  Set translation part of this matrix to values of vector t.

- void GetTranslation (TVector3< T > &t) const

  Set vector t *to translation part of this matrix.*

- TVector3< T > GetTranslation () const

  Return translation part of this matrix.

- void SetRotation (const TMatrix4x4< T > &mat)

  Set rotation part of the matrix (a 3x3 upper left corner).

- void GetRotation (TMatrix4x4< T > &mat) const

  Set mat to rotation part of this matrix.

- TMatrix4x4< T > GetRotation () const

  Return rotation part of this matrix (a 3x3 upper left corner).

- void SetRow (int r, const TVector4< T > &t)

  Set row r *of this matrix to vector* t.

- void SetColumn (int c, const TVector4< T > &t)

  Set column c *of this matrix to vector* t.

- void GetRow (int r, TVector4< T > &t) const

  Set vector t *equal to row* r *of this matrix.*

- TVector4< T > GetRow (int r) const

  Return row r.

- void GetColumn (int c, TVector4< T > &t) const

  Set vector t *equal to column* c *of this matrix.*

- TVector4< T > GetColumn (int c) const

  Return column c.

Matrix-2-Matrix operations

- TMatrix4x4< T > & MultRight (const TMatrix4x4< T > &b)
  Multiplication to matrix b *(right).*
- TMatrix4x4< T > & MultLeft (const TMatrix4x4< T > &b)
  Multiplication to matrix b *(left).*
- TMatrix4x4< T > & operator *= (const TMatrix4x4< T > &mat)
  *this *= mat*
- TMatrix4x4< T > & operator+= (const TMatrix4x4< T > &mat)
  this += mat

Matrix-2-Vector operations

- void MultMatrixVec (const TVector3< T > &src, TVector3< T > &dst) const
  *dst = M * src*
- void MultMatrixVec (TVector3< T > &src_and_dst) const
  *src_and_dst = M * src_and_dst*
- void MultVecMatrix (const TVector3< T > &src, TVector3< T > &dst) const
  *dst = src * M*
- void MultVecMatrix (TVector3< T > &src_and_dst) const
  *src_and_dst = src_and_dst * M*
- void MultMatrixVec (const TVector4< T > &src, TVector4< T > &dst) const
  *dst = M * src*
- void MultMatrixVec (TVector4< T > &src_and_dst) const
  *src_and_dst = M * src_and_dst*
- void MultVecMatrix (const TVector4< T > &src, TVector4< T > &dst) const
  *dst = src * M*
- void MultVecMatrix (TVector4< T > &src_and_dst) const
  *src_and_dst = src_and_dst * M*
- void MultMatrixDir (const TVector3< T > &src, TVector3< T > &dst) const
  *dst = M * src (only rotation and scale part, no translation)*
- void MultMatrixDir (TVector3< T > &src_and_dst) const
  *src_and_dst = M * src_and_dst (only rotation and scale part, no translation)*
- void MultDirMatrix (const TVector3< T > &src, TVector3< T > &dst) const
  *dst = src * M (only rotation and scale part, no translation)*
- void MultDirMatrix (TVector3< T > &src_and_dst) const
  *src_and_dst = src_and_dst * M (only rotation and scale part, no translation)*

Matrix-2-Scalar operations

- TMatrix4x4< T > & operator *= (const T &r)

  Multiplication by scalar r (diagonal).

Miscellaneous

- void MakeIdentity ()

  Make E-matrix from this one.

- TMatrix4x4< T > Inverse () const

  Return an inverse matrix.

- TMatrix4x4< T > Transpose () const

  Return transposed matrix.

- void **Ortho** (double left, double right, double bottom, double top, double znear, double zfar)

Static Public Member Functions

- const TMatrix4x4< T > & Identity ()

  Return predefined identity matrix.

Protected Attributes

- T **m** [16]

Friends

- TMatrix4x4< T > operator * (const TMatrix4x4< T > &m1, const TMatrix4x4< T > &m2)

  *Matrix multiplication (return m1 * m2).*

- bool operator== (const TMatrix4x4< T > &m1, const TMatrix4x4< T > &m2)

  Equality operator.

- bool operator!= (const TMatrix4x4< T > &m1, const TMatrix4x4< T > &m2)

  Unequality operator.

Related Functions

(Note that these are not member functions.)

- TMatrix4x4< T_TO > Conv (const TMatrix4x4< T_FROM > &u)
  Convert TMatrix4x4<T_FROM> to TMatrix4x4<T_TO>.
- TMatrix4x4< float > ConvF (const TMatrix4x4< T > &u)
  Convert TMatrix4x4<T> to TMatrix4x4<float>.
- TMatrix4x4< double > ConvD (const TMatrix4x4< T > &u)
  Convert TMatrix4x4<T> to TMatrix4x4<double>.

8.48.1 Detailed Description

template<class T> class gml::TMatrix4x4< T >

Template class for 4x4 matrix.

Parameters:
T - template type of matrix elements

Warning:
Use it with some SIGNED integral type, or a real type Elements are stored in column-major order (OpenGL-style) E.g.:

```
a0  a4  a8  a12
a1  a5  a9  a13
a2  a6  a10 a14
a3  a7  a11 a15
```

8.48.2 Member Function Documentation

8.48.2.1 T* RawArray () [inline]

Provide access to internal contents.

Warning:
Use with care, this function is dangerous!

8.48.2.2 void SetTranslation (const TVector3< T > & *t*) [inline]

Set translation part of this matrix to values of vector *t*.

Translation part is the 4-th column, so it is valid for use with column-vectors V = M * V; V = (v1 v2 v3 v4)T

See also:
MultMatrixVec

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

- gmlmatrix4.h

8.49 TQuaternion Class Template Reference

quaternion template

`#include <gmlquat.h>`

Public Member Functions

- **TQuaternion** (const T v[4])
- **TQuaternion** (T q0, T q1, T q2, T q3)
- **TQuaternion** (const TMatrix4x4< T > &m)
- **TQuaternion** (const TVector3< T > &axis, T radians)
- **TQuaternion** (const TVector3< T > &rotateFrom, const TVector3< T > &rotateTo)
- **TQuaternion** (const TVector3< T > &from_look, const TVector3< T > &from_up, const TVector3< T > &to_look, const TVector3< T > &to_up)
- const T * **GetValue** () const
- void **GetValue** (T &q0, T &q1, T &q2, T &q3) const
- TQuaternion & **SetValue** (T q0, T q1, T q2, T q3)
- void **GetValue** (TVector3< T > &axis, T &radians) const
- void **GetValue** (TMatrix4x4< T > &m) const
- TQuaternion & **SetValue** (const T *qp)
- TQuaternion & **SetValue** (const TMatrix4x4< T > &m)
- TQuaternion & **SetValue** (const TVector3< T > &axis, T theta)
- TQuaternion & **SetValue** (const TVector3< T > &rotateFrom, const TVector3< T > &rotateTo)
- TQuaternion & **SetValue** (const TVector3< T > &from_look, const TVector3< T > &from_up, const TVector3< T > &to_look, const TVector3< T > &to_up)
- TQuaternion & **operator** *= (const TQuaternion &qr)
- void **Normalize** ()
- bool **Equals** (const TQuaternion &r, T tolerance) const
- TQuaternion & **Conjugate** ()
- TQuaternion & **Invert** ()
- TQuaternion **Inverse** () const
- void **MultVec** (const TVector3< T > &src, TVector3< T > &dst) const
- void **MultVec** (TVector3< T > &src_and_dst) const
- void **ScaleAngle** (T scaleFactor)
- T & **operator[ ]** (int i)
- const T & **operator[ ]** (int i) const

Static Public Member Functions

- TQuaternion **Slerp** (const TQuaternion &p, const TQuaternion &q, T alpha)
- TQuaternion **Identity** ()

Protected Member Functions

- void **CounterNormalize** ()

Protected Attributes

- unsigned char **counter**

Friends

- bool **operator==** (const TQuaternion &q1, const TQuaternion &q2)
- bool **operator!=** (const TQuaternion &q1, const TQuaternion &q2)
- TQuaternion **operator** * (const TQuaternion &q1, const TQuaternion &q2)

Related Functions

(Note that these are not member functions.)

- TQuaternion< T_TO > Conv (const TQuaternion< T_FROM > &u)
  Convert TQuaternion<T_FROM> to TQuaternion<T_TO>.
- TQuaternion< float > ConvF (const TQuaternion< T > &u)
  Convert TQuaternion<T> to TQuaternion<float>.
- TQuaternion< double > ConvD (const TQuaternion< T > &u)
  Convert TQuaternion<T> to TQuaternion<double>.

8.49.1 Detailed Description

template<class T> class gml::TQuaternion< T >

quaternion template

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

- gmlquat.h

8.50 TVector2 Class Template Reference

Template class for 2D vectors.

`#include <gmlvector2.h>`

Element access

- T & **operator[ ]** (int i)
- const T & **operator[ ]** (int i) const
- void SetValue (double x0, double y0)

  Set elements by 2 double components.

- void SetValue (double a)

  Set all the elements to the same T value.

- **GetDim** ()

Public Member Functions

Constructors

- TVector2 ()

  No initialization.

- TVector2 (double v)

  Initialization by scalar.

- TVector2 (double x0, double y0)

  Initialization by given components.

- **TVector2** (const TVector3< T > &u)

Conversions

- operator T * ()

  Convert a TVector to array of (2) elements.

- operator const T * () const

  Convert a TVector to array of (2) elements.

Binary operations

- TVector2< T > & **operator+=** (const TVector2< T > &u)
- TVector2< T > & **operator-=** (const TVector2< T > &u)
- TVector2< T > **operator+** (const TVector2< T > &u) const
- TVector2< T > **operator-** (const TVector2< T > &u) const
- TVector2< T > & **operator** *= (const double d)
- TVector2< T > & **operator/=** (const double d)

- TVector2< T > MultVec (const TVector2< T > &u) const
  Vector scaling (this vector is not modified).
- TVector2< T > DivVec (const TVector2< T > &u) const
  Vector-by vector division (this vector is not modified).
- TVector2< T > **operator** * (const double d) const
- TVector2< T > **operator/** (const double d) const
- double operator * (const TVector2< T > &rhs) const
  dot product
- double operator^ (const TVector2< T > &rhs) const
  cross product
- double DotProduct (const TVector2< T > &rhs) const
  dot product
- double CrossProduct (const TVector2< T > &rhs) const
  cross product

Unary operations

- void Negate ()
  Vector (self) negation.
- TVector2< T > **operator-** () const

Service methods

- void Clip (double vmin, double vmax)
  Clip elements of this TVector to constraints from vmin to vmax.
- TVector2< double > operator~ () const
  returns normalized vector (always double)
- TVector2< T > & Normalize ()
  Normalize this vector, return vector itself.
- bool Normalized () const
  Test whether the TVector is normalized.
- double SqrLength () const
  Squared length of vector.
- double Length () const
  Length of the vector.
- bool LessOrEqual (const TVector2< T > &u) const
  Return true if all components of this vector less of equal than u.

- double operator! () const

  Length of the vector.

- TVector2< T > NormalRight () const

  Returns normal N to this vector, such that DotProduct(∗this, N) = 0.

MFC, VCL, OWL - specific functions (defined only if proper defines are found)

- **TVector2** (const TPoint &p)
- TVector2< T > & **operator=** (const TPoint &p)
- **operator TPoint** ()
- **TVector2** (const CPoint &p)
- **TVector2** (const POINT &p)
- **TVector2** (const CSize &p)
- TVector2< T > & **operator=** (const CPoint &p)
- TVector2< T > & **operator=** (const POINT &p)
- TVector2< T > & **operator=** (const CSize &p)
- **operator CPoint** ()
- **operator CvPoint** ()
- **operator CvSize** ()
- **TVector2** (const CvPoint &p)
- TVector2 **operator=** (const CvPoint &p)

Static Public Member Functions

- const TVector2< T > & Cast (const T ∗u)

  Treat array of (2) elements as a TVector.

- TVector2< T > & Cast (T ∗u)

  Treat array of (2) elements as a TVector.

Related Functions

(Note that these are not member functions.)

- TVector2< T > operator ∗ (const double d, const TVector2< T > &u)

  Multiplication of scalar d by TVector u.

- double SqrLength (const TVector2< T > &u)

  Squared length of given vector.

- double Length (const TVector2< T > &u)

  Length of the vector.

- double DotProd (const TVector2< T > &v1, const TVector2< T > &v2)

  Dot product.

- double DotProduct (const TVector2< T > &v1, const TVector2< T > &v2)
  Dot product.
- double CrossProd (const TVector2< T > &v1, const TVector2< T > &v2)
  Cross product.
- double CrossProduct (const TVector2< T > &v1, const TVector2< T > &v2)
  Cross product.
- double Cos (const TVector2< T > &a, const TVector2< T > &b)
  cos between two vectors
- double Sin (const TVector2< T > &a, const TVector2< T > &b)
  sin between two vectors
- TVector2< T_TO > Conv (const TVector2< T_FROM > &u)
  Convert TVector2<T_FROM> to TVector2<T_TO>.
- TVector2< float > ConvF (const TVector2< T > &u)
  Convert TVector2<T> to TVector2<float>.
- TVector2< double > ConvD (const TVector2< T > &u)
  Convert TVector2<T> to TVector2<double>.
- TVector2< int > ConvI (const TVector2< T > &u)
  Convert TVector2<T> to TVector2<int>.
- TVector2< short > ConvS (const TVector2< T > &u)
  Convert TVector2<T> to TVector2<int>.

8.50.1 Detailed Description

template<class T> class gml::TVector2< T >

Template class for 2D vectors.

Parameters:
T - template type of TVector elements

Warning:
Use it with some SIGNED integral type, or a real type

8.50.2 Member Function Documentation

8.50.2.1 TVector2<T> NormalRight () const [inline]

Returns normal N to this vector, such that DotProduct(∗this, N) = 0.

Parameters:

def - vector returned if the length of this vector is near zero

Note:

Projection of the resulting vector to the X axis will have the same sign that projection of this vector. If x = 0, then normal will be equal (1,0)

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

- gmlvector2.h

8.51 TVector3 Class Template Reference

Template class for 3D geometric vectors.

`#include <gmlvector3.h>`

Element access

- T & **operator[ ]** (int i)
- const T & **operator[ ]** (int i) const
- void SetValue (double x0, double y0, double z0)

  Set elements by 3 double components.

- void SetValue (double a)

  Set all the elements to the same T value.

- **GetDim** ()

Public Member Functions

Constructors

- TVector3 ()

  No initialization.

- TVector3 (double v)

  Initialization by scalar.

- TVector3 (double x0, double y0, double z0)

  Initialization by given components.

- **TVector3** (T *data)
- **TVector3** (const TVector3< T > &data)

Conversions

- operator T * ()

  Convert a TVector to array of (3) elements.

- operator const T * () const

  Convert a TVector to array of (3) elements.

Binary operations

- TVector3< T > & **operator+=** (const TVector3< T > &u)
- TVector3< T > & **operator-=** (const TVector3< T > &u)
- TVector3< T > **operator+** (const TVector3< T > &u) const
- TVector3< T > **operator-** (const TVector3< T > &u) const
- TVector3< T > & **operator** *= (const double d)

- TVector3< T > MultVec (const TVector3< T > &u) const
  Vector scaling (this vector is not modified).
- TVector3< T > DivVec (const TVector3< T > &u) const
  Vector-by vector division (this vector is not modified).
- TVector3< T > & **operator/=** (const double d)
- TVector3< T > **operator** * (const double d) const
- TVector3< T > **operator/** (const double d) const
- TVector3< T > **operator/** (const TVector3< T > v) const
- double operator * (const TVector3< T > &rhs) const
  dot product
- TVector3< T > operator^ (const TVector3< T > &rhs) const
  cross product
- double DotProduct (const TVector3< T > &rhs) const
  dot product
- TVector3< T > CrossProduct (const TVector3< T > &rhs) const
  cross product

Unary operations

- void Negate ()
  Vector (self) negation.
- TVector3< T > operator- () const
  Unary negation operator.
- TVector3< double > operator~ () const
  returns normalized vector (always double)
- TVector3< T > & Normalize ()
  Normalize this vector, return vector itself.
- double Length () const
  Length of the vector.
- double operator! () const
  Length of the vector.

Service methods

Clip elements of this TVector to constraints from vmin to vmax

- void **Clip** (double vmin, double vmax)
- bool Normalized () const
  Test whether the TVector is normalized.

- double SqrLength () const

  Squared length of vector.

- bool LessOrEqual (const TVector3< T > &u) const

  Return true if all components of this vector less of equal than u.

- T MaxValue () const

  Return C-norm of the vector (maximal value among coords).

MFC, VCL, OWL - specific functions (defined only if proper defines are found)

- **TVector3** (const TPoint &p)
- TVector3< T > & **operator=** (const TPoint &p)
- **operator TPoint** ()
- **TVector3** (const CPoint &p)
- **TVector3** (const CSize &p)
- TVector3< T > & **operator=** (const CPoint &p)
- TVector3< T > & **operator=** (const CSize &p)
- **operator CPoint** ()
- **operator CvPoint** ()
- **TVector3** (const CvPoint &p)
- TVector3 **operator=** (const CvPoint &p)

Static Public Member Functions

- const TVector3< T > & Cast (const T *u)

  Treat array of (3) elements as a TVector.

- TVector3< T > & Cast (T *u)

  Treat array of (3) elements as a TVector.

Related Functions

(Note that these are not member functions.)

- TVector3< T > operator * (const double d, const TVector3< T > &u)

  Multiplication of scalar d by TVector u.

- double SqrLength (const TVector3< T > &u)

  Squared length of given vector.

- double Length (const TVector3< T > &u)

  Length of the vector.

- double DotProduct (const TVector3< T > &v1, const TVector3< T > &v2)

  Dot product.

- TVector3< T > CrossProduct (const TVector3< T > &v1, const TVector3< T > &v2)
  Cross product.
- double Cos (const TVector3< T > &a, const TVector3< T > &b)
  cos between two vectors
- double Sin (const TVector3< T > &a, const TVector3< T > &b)
  sin between two vectors
- TVector3< T_TO > Conv (const TVector3< T_FROM > &u)
  Convert TVector3<T_FROM> to TVector3<T_TO>.
- TVector3< float > ConvF (const TVector3< T > &u)
  Convert TVector3<T> to TVector3<float>.
- TVector3< double > ConvD (const TVector3< T > &u)
  Convert TVector3<T> to TVector3<double>.
- TVector3< int > ConvI (const TVector3< T > &u)
  Convert TVector3<T> to TVector3<int>.
- TVector3< short > ConvS (const TVector3< T > &u)
  Convert TVector3<T> to TVector3<int>.

8.51.1 Detailed Description

template<class T> class gml::TVector3< T >

Template class for 3D geometric vectors.

Parameters:
T - template type of TVector elements

Warning:
Use it with some SIGNED integral type, or a real type

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

- gmlvector3.h

8.52 TVector4 Class Template Reference

Template class for 4D geometric vectors.

`#include <gmlvector4.h>`

Public Member Functions

Constructors

- TVector4 ()
  No initialization.
- TVector4 (double v)
  Initialization by scalar.
- TVector4 (double x0, double y0, double z0, double w0)
  Initialization by given components.
- TVector4 (const TVector3< T > &v, double w0=1)
  Initialization by given 3d-vector (w become equal to 1).

Element access

- T & **operator[ ]** (int i)
- const T & **operator[ ]** (int i) const
- void SetValue (double x0, double y0, double z0, double w0)
  Set elements by 3 double components.
- void SetValue (double a)
  Set all the elements to the same T value.

Conversions

- operator T * ()
  Convert a TVector to array of (3) elements.
- operator const T * () const
  Convert a TVector to array of (3) elements.

Binary operations

- TVector4< T > & **operator+=** (const TVector4< T > &u)
- TVector4< T > & **operator-=** (const TVector4< T > &u)
- TVector4< T > **operator+** (const TVector4< T > &u) const
- TVector4< T > **operator-** (const TVector4< T > &u) const
- TVector4< T > & **operator** *= (const double d)

- TVector4< T > & **operator/=** (const double d)
- TVector4< T > MultVec (const TVector4< T > &u) const

  Vector scaling (this vector is not modified).

- TVector4< T > DivVec (const TVector4< T > &u) const

  Vector-by vector division (this vector is not modified).

- TVector4< T > **operator** * (const double d) const
- TVector4< T > **operator/** (const double d) const
- double operator * (const TVector4< T > &rhs) const

  dot product

- double DotProduct (const TVector4< T > &rhs) const

  dot product

Unary operations

- void Negate ()

  Vector (self) negation.

- TVector4< T > operator- () const

  Unary negation operator.

- TVector4< double > operator~ () const

  returns normalized vector (always double)

- TVector4< T > & Normalize ()

  Normalize this vector, return vector itself.

- double Length () const

  Length of the vector.

- double operator! () const

  Length of the vector.

Service methods

- void Clip (double vmin, double vmax)

  Clip elements of this TVector to constraints from vmin to vmax.

- bool Normalized () const

  Test whether the TVector is normalized.

- double SqrLength () const

  Squared length of vector.

- bool LessOrEqual (const TVector4< T > &u) const

  Return true if all components of this vector less of equal than u.

- bool LessOrEqual (const TVector3< T > &u) const

  Return true if all components of this vector less of equal than u.

- T MaxValue () const

  Return C-norm of the vector (maximal value among coords).

- TVector3< double > **Homogenize** ()

MFC, VCL, OWL - specific functions (defined only if proper defines are found)

- **TVector4** (const TPoint &p)
- TVector4< T > & **operator=** (const TPoint &p)
- **operator TPoint** ()
- **TVector4** (const CPoint &p)
- **TVector4** (const CSize &p)
- TVector4< T > & **operator=** (const CPoint &p)
- TVector4< T > & **operator=** (const CSize &p)
- **operator CPoint** ()
- **operator CvPoint** ()
- **TVector4** (const CvPoint &p)
- TVector4 **operator=** (const CvPoint &p)

Static Public Member Functions

- const TVector4< T > & Cast (const T *u)

  Treat array of (3) elements as a TVector.

- TVector4< T > & Cast (T *u)

  Treat array of (3) elements as a TVector.

Related Functions

(Note that these are not member functions.)

- TVector4< T > operator * (const double d, const TVector4< T > &u)

  Multiplication of scalar d by TVector u.

- double SqrLength (const TVector4< T > &u)

  Squared length of given vector.

- double Length (const TVector4< T > &u)

  Length of the vector.

- double DotProduct (const TVector4< T > &v1, const TVector4< T > &v2)

  Dot product.

- TVector4< T_TO > Conv (const TVector4< T_FROM > &u)

Convert TVector4<T_FROM> to TVector4<T_TO>.

- TVector4< float > ConvF (const TVector4< T > &u)
  Convert TVector4<T> to TVector4<float>.
- TVector4< double > ConvD (const TVector4< T > &u)
  Convert TVector4<T> to TVector4<double>.
- TVector4< int > ConvI (const TVector4< T > &u)
  Convert TVector4<T> to TVector4<int>.
- TVector4< short > ConvS (const TVector4< T > &u)
  Convert TVector4<T> to TVector4<int>.

8.52.1 Detailed Description

template<class T> class gml::TVector4< T >

Template class for 4D geometric vectors.

Parameters:
T - template type of TVector elements

Warning:
Use it with some SIGNED integral type, or a real type

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

- gmlvector4.h

8.53 Viewport Class Reference

Class representing 3d viewport (a window with accociated camera).

`#include <gmlviewport.h>`

Public Member Functions

- **Viewport** (int x, int y, int xres, int yres, Camera *camera=0)
- void **GetResolution** (int &xres, int &yres) const
- void **SetResolution** (int xres, int yres)
- void **SetOrigin** (int x, int y)
- void **GetOrigin** (int &x, int &y) const
- const Matrix4x4d & **GetTransform** () const
- void **SetTransform** (const Matrix4x4d &view_transform)
- float **GetViewAngle** () const
- void **SetViewAngle** (float view_angle)
- const Camera * **GetCamera** () const
- void **SetCamera** (Camera *cam)
- Vector3d GetViewVector () const

  get view vector of camera (normalized)

- Vector3d GetUpVector () const

  get up vector of camera (normalized)

- Vector3d GetRightVector () const

  get up vector of camera (normalized)

- Vector3d GetObserver () const

  get camera position

- void **SetObserver** (const Vector3d &obs)
- Matrix4x4d **GetRotation** () const
- void **SetRotation** (const Matrix4x4d &rot)
- void **SetDistanceToTarget** (float dist)
- float **GetDistanceToTarget** () const

8.53.1 Detailed Description

Class representing 3d viewport (a window with accociated camera).

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

- gmlviewport.h

Chapter 9

Graphics and Media Lab CSL File Documentation

9.1 gmlbbox2.h File Reference

A class for 2D boundary box (BBox2) definition.

```
#include "gmlvector2.h"
#include <limits>
```

Namespaces

- namespace **gml**

9.1.1 Detailed Description

A class for 2D boundary box (BBox2) definition.

9.2 gmlbbox3.h File Reference

A class for 2D boundary box (BBox3) definition.

```
#include "gmlvector3.h"
#include <limits>
```

Namespaces

- namespace **gml**

9.2.1 Detailed Description

A class for 2D boundary box (BBox3) definition.

9.3 gmlbrowser.h File Reference

A class for 3d virtual camera.

```
#include "gmlviewport.h"
#include "../math/gmlvector2.h"
#include "../math/gmlbbox3.h"
```

Namespaces

- namespace **gml**

9.3.1 Detailed Description

A class for 3d virtual camera.

9.4 gmlbsphere3.h File Reference

defines template class BSphere33<T>

```
#include "gmlmath.h"
#include "gmlbbox3.h"
```

Namespaces

- namespace **gml**

9.4.1 Detailed Description

defines template class BSphere33<T>

9.5 gmlcallback.h File Reference

Definitions of callback helpers.

```
#include "gmlfunctioncallback.h"
#include "gmlmethodcallback.h"
```

Namespaces

- namespace **gml**

9.5.1 Detailed Description

Definitions of callback helpers.

9.6 gmlcamera.h File Reference

A class for 3d virtual camera.

```
#include "../math/gmlmatrix4.h"
```

Namespaces

- namespace **gml**

9.6.1 Detailed Description

A class for 3d virtual camera.

9.7 gmlcommon.h File Reference

General routines like ASSERT etc.

`#include <assert.h>`

Namespaces

- namespace **gml**

Defines

- #define ASSERT(expr)

  custom assertion (DEBUG mode only)

- #define **VERIFY**(f) ((void)(f))
- #define **NULL** 0

Typedefs

- typedef unsigned char **BYTE**
- typedef unsigned short **WORD**
- typedef unsigned long **DWORD**

9.7.1 Detailed Description

General routines like ASSERT etc.

9.8 gmlcommongl.h File Reference

OpenGL helpers.

Namespaces

- namespace **gml**

Defines

- #define **CHECK_GL** ASSERT(glGetError() == GL_NO_ERROR)

9.8.1 Detailed Description

OpenGL helpers.

9.9 gmlframecounter.h File Reference

Simple frame rate (FPS) counter.

Namespaces

- namespace **gml**

9.9.1 Detailed Description

Simple frame rate (FPS) counter.

9.10 gmlglrc.h File Reference

Class for operations with GL Win32 context.

Namespaces

- namespace **gml**

9.10.1 Detailed Description

Class for operations with GL Win32 context.

9.11 gmlmath.h File Reference

Common mathematic utility routines, gml::Math.

`#include <math.h>`

Namespaces

- namespace **gml**

Defines

- #define **PI** 3.1415926535897

9.11.1 Detailed Description

Common mathematic utility routines, gml::Math.

9.12 gmlmatrix2.h File Reference

Definition of TMatrix2x2 template class and utility routines.

```
#include "gmlvector2.h"
```

Namespaces

- namespace **gml**

9.12.1 Detailed Description

Definition of TMatrix2x2 template class and utility routines.

9.13 gmlmatrix3.h File Reference

Definition of TMatrix3x3 template class and utility routines.

```
#include "gmlvector3.h"
#include "gmlvector2.h"
```

Namespaces

- namespace **gml**

9.13.1 Detailed Description

Definition of TMatrix3x3 template class and utility routines.

9.14 gmlmatrix4.h File Reference

Definition of TMatrix4x4 template class and utility routines.

```
#include "gmlvector3.h"
#include "gmlvector4.h"
```

Namespaces

- namespace **gml**

9.14.1 Detailed Description

Definition of TMatrix4x4 template class and utility routines.

9.15 gmlmesh.h File Reference

Definition of gml::Mesh class.

```
#include "../base/gmlsmartobject.h"
#include "../color/gmlcolor.h"
#include "../math/gmlvector2.h"
#include "../math/gmlvector3.h"
#include <vector>
```

Namespaces

- namespace **gml**

9.15.1 Detailed Description

Definition of gml::Mesh class.

9.16 gmlpathstring.h File Reference

defines gml::PathString

```
#include "../base/gmlcommon.h"
#include "../base/gmlstring.h"
#include <list>
```

Namespaces

- namespace **gml**
- namespace **std**

9.16.1 Detailed Description

defines gml::PathString

9.17 gmlquat.h File Reference

Definition of TQuaternion template class.

```
#include "gmlmatrix4.h"
```

Namespaces

- namespace **gml**

9.17.1 Detailed Description

Definition of TQuaternion template class.

9.18 gmlref.h File Reference

Definitions of smart reference.

Namespaces

- namespace **gml**

9.18.1 Detailed Description

Definitions of smart reference.

9.19 gmlsmartobject.h File Reference

Definitions of smart object (which supports smart references).

Namespaces

- namespace **gml**

9.19.1 Detailed Description

Definitions of smart object (which supports smart references).

9.20 gmlsocket.h File Reference

Windows sockets handling (Win32 API).

`#include <string>`

Namespaces

- namespace **gml**

9.20.1 Detailed Description

Windows sockets handling (Win32 API).

9.21 gmlstring.h File Reference

defines gml::String

```
#include <string>
#include <afxwin.h>
```

Namespaces

- namespace **gml**

9.21.1 Detailed Description

defines gml::String

9.22 gmlutils3d.h File Reference

3d projections etc (utilities)

```
#include "gmlviewport.h"
#include "../math/gmlbbox3.h"
```

Namespaces

- namespace **gml**

9.22.1 Detailed Description

3d projections etc (utilities)

9.23 gmlutilsgl.h File Reference

A set of OGL utitility functions.

```
#include "../math/gmlbbox3.h"
#include "../color/gmlcolor.h"
```

Namespaces

- namespace **gml**

9.23.1 Detailed Description

A set of OGL utitility functions.

9.24 gmlvector2.h File Reference

defines template gml::TVector2, gml::Math2 and typedefs with utility functions

```
#include "gmlmath.h"
#include "gmlvector3.h"
#include "../base/gmlcommon.h"
#include <ipl.h>
#include <cv.h>
#include <afxwin.h>
#include <windows.hpp>
```

Namespaces

- namespace **gml**

9.24.1 Detailed Description

defines template gml::TVector2, gml::Math2 and typedefs with utility functions

9.25 gmlvector3.h File Reference

defines template gml::TVector3, gml::Math3 and typedefs with utility functions

```
#include "gmlmath.h"
#include <afxwin.h>
#include <ipl.h>
#include <cv.h>
#include <windows.hpp>
```

Namespaces

- namespace **gml**

9.25.1 Detailed Description

defines template gml::TVector3, gml::Math3 and typedefs with utility functions

9.26 gmlvector4.h File Reference

defines template gml::TVector4, gml::Math3 and typedefs with utility functions

`#include "gmlvector3.h"`

Namespaces

- namespace **gml**

9.26.1 Detailed Description

defines template gml::TVector4, gml::Math3 and typedefs with utility functions

9.27 gmlviewport.h File Reference

A class for 3d viewport.

```
#include "gmlcamera.h"
#include "../math/gmlvector3.h"
```

Namespaces

- namespace **gml**

9.27.1 Detailed Description

A class for 3d viewport.

Chapter 10

Graphics and Media Lab CSL Page Documentation

10.1 Todo List

Class FileFinder : add directory structure handling

Class GLRC Add possibility to select pixel format (double/single buffer, stencil, etc)

10.2 Deprecated List

Member ColorRGBub use Color3 instead

Member ColorRGBw use Color3 instead

Member ColorRGBf use Color3 instead

Member ColorRGBAub use Color3 instead

Member ColorRGBAf use Color3 instead

Member DotProd(const TVector2< T > &v1, const TVector2< T > &v2) use use DotProduct instead

Member CrossProd(const TVector2< T > &v1, const TVector2< T > &v2) use CrossProduct instead