OpenCTM

1.0.3

Generated by Doxygen 1.12.0

1 OpenCTM API Reference 1
1.1 Introduction . . . 1
1.2 Usage . . . 1
1.3 Example usage . . . 1
1.3.1 Loading a CTM file . . . 1
1.3.2 Creating a CTM file . . . 2

2 Hierarchical Index 3
2.1 Class Hierarchy . . . 3

3 Class Index 5
3.1 Class List . . . 5

4 File Index 7
4.1 File List . . . 7

5 Class Documentation 9
5.1 ctm_error Class Reference . . . 9
5.1.1 Detailed Description . . . 9
5.1.2 Constructor & Destructor Documentation . . . 9
5.1.2.1 ctm_error() . . . 9
5.1.3 Member Function Documentation . . . 10
5.1.3.1 error_code() . . . 10
5.1.3.2 what() . . . 10
5.2 CTMexporter Class Reference . . . 10
5.2.1 Detailed Description . . . 11
5.2.2 Constructor & Destructor Documentation . . . 11
5.2.2.1 CTMexporter() [1/2] . . . 11
5.2.2.2 ~CTMexporter() . . . 11
5.2.2.3 CTMexporter() [2/2] . . . 11
5.2.3 Member Function Documentation . . . 11
5.2.3.1 AddAttribMap() . . . 11
5.2.3.2 AddUVMap() . . . 12
5.2.3.3 AttribPrecision() . . . 12
5.2.3.4 CompressionLevel() . . . 12
5.2.3.5 CompressionMethod() . . . 12
5.2.3.6 DefineMesh() . . . 12
5.2.3.7 FileComment() . . . 12
5.2.3.8 NormalPrecision() . . . 13
5.2.3.9 operator=() . . . 13
5.2.3.10 Save() . . . 13
5.2.3.11 SaveCustom() . . . 13
5.2.3.12 UVCoordPrecision() . . . 13
5.2.3.13 VertexPrecision() . . . 13

5.2.3.14 VertexPrecisionRel() . . . 13
5.3 CTMimporter Class Reference . . . 14
5.3.1 Detailed Description . . . 14
5.3.2 Constructor & Destructor Documentation . . . 15
5.3.2.1 CTMimporter() [1/2] . . . 15
5.3.2.2 ~CTMimporter() . . . 15
5.3.2.3 CTMimporter() [2/2] . . . 15
5.3.3 Member Function Documentation . . . 15
5.3.3.1 GetAttribMapFloat() . . . 15
5.3.3.2 GetAttribMapString() . . . 15
5.3.3.3 GetFloat() . . . 15
5.3.3.4 GetFloatArray() . . . 16
5.3.3.5 GetInteger() . . . 16
5.3.3.6 GetIntegerArray() . . . 16
5.3.3.7 GetNamedAttribMap() . . . 16
5.3.3.8 GetNamedUVMap() . . . 16
5.3.3.9 GetString() . . . 16
5.3.3.10 GetUVMapFloat() . . . 16
5.3.3.11 GetUVMapString() . . . 17
5.3.3.12 Load() . . . 17
5.3.3.13 LoadCustom() . . . 17
5.3.3.14 operator=() . . . 17

6 File Documentation **19**
6.1 openctm.h File Reference . . . 19
6.1.1 Macro Definition Documentation . . . 22
6.1.1.1 CTM_API_VERSION . . . 22
6.1.1.2 CTM_FALSE . . . 22
6.1.1.3 CTM_TRUE . . . 22
6.1.2 Typedef Documentation . . . 22
6.1.2.1 CTMcontext . . . 22
6.1.2.2 CTMfloat . . . 23
6.1.2.3 CTMint . . . 23
6.1.2.4 CTMreadfn . . . 23
6.1.2.5 CTMuint . . . 23
6.1.2.6 CTMwritefn . . . 23
6.1.3 Enumeration Type Documentation . . . 24
6.1.3.1 CTMenum . . . 24
6.1.4 Function Documentation . . . 25
6.1.4.1 ctmAddAttribMap() . . . 25
6.1.4.2 ctmAddUVMap() . . . 26
6.1.4.3 ctmAttribPrecision() . . . 26

6.1.4.4 ctmCompressionLevel() . . . 27
6.1.4.5 ctmCompressionMethod() . . . 27
6.1.4.6 ctmDefineMesh() . . . 27
6.1.4.7 ctmErrorString() . . . 28
6.1.4.8 ctmFileComment() . . . 28
6.1.4.9 ctmFreeContext() . . . 28
6.1.4.10 ctmGetAttribMapFloat() . . . 29
6.1.4.11 ctmGetAttribMapString() . . . 29
6.1.4.12 ctmGetError() . . . 30
6.1.4.13 ctmGetFloat() . . . 30
6.1.4.14 ctmGetFloatArray() . . . 31
6.1.4.15 ctmGetInteger() . . . 31
6.1.4.16 ctmGetIntegerArray() . . . 31
6.1.4.17 ctmGetNamedAttribMap() . . . 32
6.1.4.18 ctmGetNamedUVMap() . . . 32
6.1.4.19 ctmGetString() . . . 33
6.1.4.20 ctmGetUVMapFloat() . . . 33
6.1.4.21 ctmGetUVMapString() . . . 34
6.1.4.22 ctmLoad() . . . 34
6.1.4.23 ctmLoadCustom() . . . 35
6.1.4.24 ctmNewContext() . . . 35
6.1.4.25 ctmNormalPrecision() . . . 35
6.1.4.26 ctmSave() . . . 36
6.1.4.27 ctmSaveCustom() . . . 36
6.1.4.28 ctmUVCoordPrecision() . . . 36
6.1.4.29 ctmVertexPrecision() . . . 37
6.1.4.30 ctmVertexPrecisionRel() . . . 37
6.2 openctm.h . . . 38
6.3 openctmpp.h File Reference . . . 41
6.4 openctmpp.h . . . 41

Index **45**

Chapter 1

OpenCTM API Reference

1.1 Introduction

OpenCTM is an open file format for storing compressed triangle meshes. In order to easily read and write OpenCTM files (usually suffixed .ctm) an API (Application Program Interface) is provided that can easily be used from most modern programming languages.

The OpenCTM functionality itself is written in highly portable standard C (C99).

1.2 Usage

For information about how to use the OpenCTM API, see openctm.h.

For information about the C++ wrapper classes, see CTMimporter and CTMexporter.

1.3 Example usage

1.3.1 Loading a CTM file

Here is a simple example of loading a CTM file:

```
CTMcontext context;
CTMuint    vertCount, triCount, * indices;
CTMfloat   * vertices;

// Create a new context
context = ctmNewContext(CTM_IMPORT);

// Load the OpenCTM file
ctmLoad(context, "mymesh.ctm");
if(ctmGetError(context) == CTM_NONE)
{
  // Access the mesh data
  vertCount = ctmGetInteger(context, CTM_VERTEX_COUNT);
  vertices = ctmGetFloatArray(context, CTM_VERTICES);
  triCount = ctmGetInteger(context, CTM_TRIANGLE_COUNT);
  indices = ctmGetIntegerArray(context, CTM_INDICES);

  // Deal with the mesh (e.g. transcode it to our internal representation)
  // ...
}

// Free the context
ctmFreeContext(context);
```

1.3.2 Creating a CTM file

Here is a simple example of creating a CTM file:

```
CTMcontext context;
CTMuint    vertCount, triCount, * indices;
CTMfloat   * vertices;

// Create our mesh in memory
vertCount = 100;
triCount = 120;
vertices = (CTMfloat *) malloc(3 * sizeof(CTMfloat) * vertCount);
indices = (CTMuint *) malloc(3 * sizeof(CTMuint) * triCount);
// ...

// Create a new context
context = ctmNewContext(CTM_EXPORT);

// Define our mesh representation to OpenCTM (store references to it in
// the context)
ctmDefineMesh(context, vertices, vertCount, indices, triCount, NULL);

// Save the OpenCTM file
ctmSave(context, "mymesh.ctm");

// Free the context
ctmFreeContext(context);

// Free our mesh
free(indices);
free(vertices);
```

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

CTMexporter 10
CTMimporter 14
std::exception
    ctm_error 9

Chapter 3

Class Index

3.1 Class List

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

ctm_error
: OpenCTM exception 9

CTMexporter
: OpenCTM exporter class 10

CTMimporter
: OpenCTM importer class 14

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

openctm.h 19
openctmpp.h 41

Chapter 5

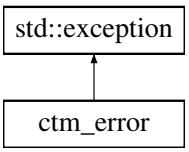
Class Documentation

5.1 ctm_error Class Reference

OpenCTM exception.

`#include <openctmpp.h>`

Inheritance diagram for ctm_error:



Public Member Functions

- ctm_error (CTMenum aError)
- virtual const char * what () const throw ()
- CTMenum error_code () const throw ()

5.1.1 Detailed Description

OpenCTM exception.

When an error occurs, a `ctm_error` exception is thrown. Its what() function returns the name of the OpenCTM error code (for instance "CTM_INVALID_OPERATION").

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ctm_error()

```
ctm_error::ctm_error (
            CTMenum aError)  [inline], [explicit]
```

5.1.3 Member Function Documentation

5.1.3.1 error_code()

```
CTMenum ctm_error::error_code () const throw ( )   [inline]
```

5.1.3.2 what()

```
virtual const char * ctm_error::what () const throw ( )   [inline], [virtual]
```

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

- openctmpp.h

5.2 CTMexporter Class Reference

OpenCTM exporter class.

`#include <openctmpp.h>`

Public Member Functions

- CTMexporter ()
  Constructor.
- ~CTMexporter ()
  Destructor.
- void CompressionMethod (CTMenum aMethod)
  Wrapper for ctmCompressionMethod()
- void CompressionLevel (CTMuint aLevel)
  Wrapper for ctmCompressionLevel()
- void VertexPrecision (CTMfloat aPrecision)
  Wrapper for ctmVertexPrecision()
- void VertexPrecisionRel (CTMfloat aRelPrecision)
  Wrapper for ctmVertexPrecisionRel()
- void NormalPrecision (CTMfloat aPrecision)
  Wrapper for ctmNormalPrecision()
- void UVCoordPrecision (CTMenum aUVMap, CTMfloat aPrecision)
  Wrapper for ctmUVCoordPrecision()
- void AttribPrecision (CTMenum aAttribMap, CTMfloat aPrecision)
  Wrapper for ctmAttribPrecision()
- void FileComment (const char *aFileComment)
  Wrapper for ctmFileComment()
- void DefineMesh (const CTMfloat *aVertices, CTMuint aVertexCount, const CTMuint *aIndices, CTMuint aTriangleCount, const CTMfloat *aNormals)
  Wrapper for ctmDefineMesh()
- CTMenum AddUVMap (const CTMfloat *aUVCoords, const char *aName, const char *aFileName)
  Wrapper for ctmAddUVMap()
- CTMenum AddAttribMap (const CTMfloat *aAttribValues, const char *aName)
  Wrapper for ctmAddAttribMap()
- void Save (const char *aFileName)
  Wrapper for ctmSave()
- void SaveCustom (CTMwritefn aWriteFn, void *aUserData)
  Wrapper for ctmSaveCustom()
- CTMexporter (const CTMexporter &v)
- CTMexporter & operator= (const CTMexporter &v)

5.2.1 Detailed Description

OpenCTM exporter class.

This is a C++ wrapper class for an OpenCTM export context. Usage example:

```
void MySaveFile(CTMuint aVertCount, CTMuint aTriCount, CTMfloat * aVertices,
  CTMuint * aIndices, const char * aFileName)
{
  // Create a new OpenCTM exporter object
  CTMexporter ctm;

  // Define our mesh representation to OpenCTM (store references to it in
  // the context)
  ctm.DefineMesh(aVertices, aVertCount, aIndices, aTriCount, NULL);

  // Save the OpenCTM file
  ctm.Save(aFileName);
}
```

5.2.2 Constructor & Destructor Documentation

5.2.2.1 CTMexporter() [1/2]

```
CTMexporter::CTMexporter ()  [inline]
```

Constructor.

5.2.2.2 ~CTMexporter()

```
CTMexporter::~CTMexporter ()  [inline]
```

Destructor.

5.2.2.3 CTMexporter() [2/2]

```
CTMexporter::CTMexporter (
            const CTMexporter & v)
```

5.2.3 Member Function Documentation

5.2.3.1 AddAttribMap()

```
CTMenum CTMexporter::AddAttribMap (
            const CTMfloat * aAttribValues,
            const char * aName)  [inline]
```

Wrapper for ctmAddAttribMap()

5.2.3.2 AddUVMap()

```
CTMenum CTMexporter::AddUVMap (
            const CTMfloat * aUVCoords,
            const char * aName,
            const char * aFileName)  [inline]
```

Wrapper for ctmAddUVMap()

5.2.3.3 AttribPrecision()

```
void CTMexporter::AttribPrecision (
            CTMenum aAttribMap,
            CTMfloat aPrecision)  [inline]
```

Wrapper for ctmAttribPrecision()

5.2.3.4 CompressionLevel()

```
void CTMexporter::CompressionLevel (
            CTMuint aLevel)  [inline]
```

Wrapper for ctmCompressionLevel()

5.2.3.5 CompressionMethod()

```
void CTMexporter::CompressionMethod (
            CTMenum aMethod)  [inline]
```

Wrapper for ctmCompressionMethod()

5.2.3.6 DefineMesh()

```
void CTMexporter::DefineMesh (
            const CTMfloat * aVertices,
            CTMuint aVertexCount,
            const CTMuint * aIndices,
            CTMuint aTriangleCount,
            const CTMfloat * aNormals)  [inline]
```

Wrapper for ctmDefineMesh()

5.2.3.7 FileComment()

```
void CTMexporter::FileComment (
            const char * aFileComment)  [inline]
```

Wrapper for ctmFileComment()

5.2.3.8 NormalPrecision()

```
void CTMexporter::NormalPrecision (
            CTMfloat aPrecision)  [inline]
```

Wrapper for ctmNormalPrecision()

5.2.3.9 operator=()

```
CTMexporter & CTMexporter::operator= (
            const CTMexporter & v)
```

5.2.3.10 Save()

```
void CTMexporter::Save (
            const char * aFileName)  [inline]
```

Wrapper for ctmSave()

5.2.3.11 SaveCustom()

```
void CTMexporter::SaveCustom (
            CTMwritefn aWriteFn,
            void * aUserData)  [inline]
```

Wrapper for ctmSaveCustom()

5.2.3.12 UVCoordPrecision()

```
void CTMexporter::UVCoordPrecision (
            CTMenum aUVMap,
            CTMfloat aPrecision)  [inline]
```

Wrapper for ctmUVCoordPrecision()

5.2.3.13 VertexPrecision()

```
void CTMexporter::VertexPrecision (
            CTMfloat aPrecision)  [inline]
```

Wrapper for ctmVertexPrecision()

5.2.3.14 VertexPrecisionRel()

```
void CTMexporter::VertexPrecisionRel (
            CTMfloat aRelPrecision)  [inline]
```

Wrapper for ctmVertexPrecisionRel()

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

- openctmpp.h

5.3 CTMimporter Class Reference

OpenCTM importer class.

`#include <openctmpp.h>`

Public Member Functions

- CTMimporter ()
  Constructor.
- ~CTMimporter ()
  Destructor.
- CTMuint GetInteger (CTMenum aProperty)
  Wrapper for ctmGetInteger()
- CTMfloat GetFloat (CTMenum aProperty)
  Wrapper for ctmGetFloat()
- const CTMuint * GetIntegerArray (CTMenum aProperty)
  Wrapper for ctmGetIntegerArray()
- const CTMfloat * GetFloatArray (CTMenum aProperty)
  Wrapper for ctmGetFloatArray()
- CTMenum GetNamedUVMap (const char *aName)
  Wrapper for ctmGetNamedUVMap()
- const char * GetUVMapString (CTMenum aUVMap, CTMenum aProperty)
  Wrapper for ctmGetUVMapString()
- CTMfloat GetUVMapFloat (CTMenum aUVMap, CTMenum aProperty)
  Wrapper for ctmGetUVMapFloat()
- CTMenum GetNamedAttribMap (const char *aName)
  Wrapper for ctmGetNamedAttribMap()
- const char * GetAttribMapString (CTMenum aAttribMap, CTMenum aProperty)
  Wrapper for ctmGetAttribMapString()
- CTMfloat GetAttribMapFloat (CTMenum aAttribMap, CTMenum aProperty)
  Wrapper for ctmGetAttribMapFloat()
- const char * GetString (CTMenum aProperty)
  Wrapper for ctmGetString()
- void Load (const char *aFileName)
  Wrapper for ctmLoad()
- void LoadCustom (CTMreadfn aReadFn, void *aUserData)
  Wrapper for ctmLoadCustom()
- CTMimporter (const CTMimporter &v)
- CTMimporter & operator= (const CTMimporter &v)

5.3.1 Detailed Description

OpenCTM importer class.

This is a C++ wrapper class for an OpenCTM import context. Usage example:

```
// Create a new OpenCTM importer object
CTMimporter ctm;

// Load the OpenCTM file
ctm.Load("mymesh.ctm");

// Access the mesh data
vertCount = ctm.GetInteger(CTM_VERTEX_COUNT);
vertices = ctm.GetFloatArray(CTM_VERTICES);
triCount = ctm.GetInteger(CTM_TRIANGLE_COUNT);
indices = ctm.GetIntegerArray(CTM_INDICES);

// Deal with the mesh (e.g. transcode it to our internal representation)
// ...
```

5.3.2 Constructor & Destructor Documentation

5.3.2.1 CTMimporter() [1/2]

```
CTMimporter::CTMimporter ()  [inline]
```

Constructor.

5.3.2.2 ~CTMimporter()

```
CTMimporter::~CTMimporter ()  [inline]
```

Destructor.

5.3.2.3 CTMimporter() [2/2]

```
CTMimporter::CTMimporter (
            const CTMimporter & v)
```

5.3.3 Member Function Documentation

5.3.3.1 GetAttribMapFloat()

```
CTMfloat CTMimporter::GetAttribMapFloat (
            CTMenum aAttribMap,
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetAttribMapFloat()

5.3.3.2 GetAttribMapString()

```
const char * CTMimporter::GetAttribMapString (
            CTMenum aAttribMap,
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetAttribMapString()

5.3.3.3 GetFloat()

```
CTMfloat CTMimporter::GetFloat (
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetFloat()

5.3.3.4 GetFloatArray()

```
const CTMfloat * CTMimporter::GetFloatArray (
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetFloatArray()

5.3.3.5 GetInteger()

```
CTMuint CTMimporter::GetInteger (
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetInteger()

5.3.3.6 GetIntegerArray()

```
const CTMuint * CTMimporter::GetIntegerArray (
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetIntegerArray()

5.3.3.7 GetNamedAttribMap()

```
CTMenum CTMimporter::GetNamedAttribMap (
            const char * aName)  [inline]
```

Wrapper for ctmGetNamedAttribMap()

5.3.3.8 GetNamedUVMap()

```
CTMenum CTMimporter::GetNamedUVMap (
            const char * aName)  [inline]
```

Wrapper for ctmGetNamedUVMap()

5.3.3.9 GetString()

```
const char * CTMimporter::GetString (
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetString()

5.3.3.10 GetUVMapFloat()

```
CTMfloat CTMimporter::GetUVMapFloat (
            CTMenum aUVMap,
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetUVMapFloat()

5.3.3.11 GetUVMapString()

```
const char * CTMimporter::GetUVMapString (
            CTMenum aUVMap,
            CTMenum aProperty)  [inline]
```

Wrapper for ctmGetUVMapString()

5.3.3.12 Load()

```
void CTMimporter::Load (
            const char * aFileName)  [inline]
```

Wrapper for ctmLoad()

5.3.3.13 LoadCustom()

```
void CTMimporter::LoadCustom (
            CTMreadfn aReadFn,
            void * aUserData)  [inline]
```

Wrapper for ctmLoadCustom()

5.3.3.14 operator=()

```
CTMimporter & CTMimporter::operator= (
            const CTMimporter & v)
```

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

- openctmpp.h

Chapter 6

File Documentation

6.1 openctm.h File Reference

```
#include <stdint.h>
```

Macros

- #define CTM_API_VERSION 0x00000100

  OpenCTM API version (1.0).
- #define CTM_TRUE 1

  Boolean TRUE.
- #define CTM_FALSE 0

  Boolean FALSE.

Typedefs

- typedef float CTMfloat

  Single precision floating point type (IEEE 754 32 bits wide).
- typedef int32_t CTMint

  Signed integer (32 bits wide).
- typedef uint32_t CTMuint

  Unsigned integer (32 bits wide).
- typedef void * CTMcontext

  OpenCTM context handle.
- typedef CTMuint(CTMCALL * CTMreadfn) (void *aBuf, CTMuint aCount, void *aUserData)

  Stream read() function pointer.
- typedef CTMuint(CTMCALL * CTMwritefn) (const void *aBuf, CTMuint aCount, void *aUserData)

  Stream write() function pointer.

Enumerations

- enum CTMenum {
  CTM_NONE = 0x0000 ,
  CTM_INVALID_CONTEXT = 0x0001 ,
  CTM_INVALID_ARGUMENT = 0x0002 ,
  CTM_INVALID_OPERATION = 0x0003 ,
  CTM_INVALID_MESH = 0x0004 ,
  CTM_OUT_OF_MEMORY = 0x0005 ,
  CTM_FILE_ERROR = 0x0006 ,
  CTM_BAD_FORMAT = 0x0007 ,
  CTM_LZMA_ERROR = 0x0008 ,
  CTM_INTERNAL_ERROR = 0x0009 ,
  CTM_UNSUPPORTED_FORMAT_VERSION = 0x000A ,
  CTM_IMPORT = 0x0101 ,
  CTM_EXPORT = 0x0102 ,
  CTM_METHOD_RAW = 0x0201 ,
  CTM_METHOD_MG1 = 0x0202 ,
  CTM_METHOD_MG2 = 0x0203 ,
  CTM_VERTEX_COUNT = 0x0301 ,
  CTM_TRIANGLE_COUNT = 0x0302 ,
  CTM_HAS_NORMALS = 0x0303 ,
  CTM_UV_MAP_COUNT = 0x0304 ,
  CTM_ATTRIB_MAP_COUNT = 0x0305 ,
  CTM_VERTEX_PRECISION = 0x0306 ,
  CTM_NORMAL_PRECISION = 0x0307 ,
  CTM_COMPRESSION_METHOD = 0x0308 ,
  CTM_FILE_COMMENT = 0x0309 ,
  CTM_NAME = 0x0501 ,
  CTM_FILE_NAME = 0x0502 ,
  CTM_PRECISION = 0x0503 ,
  CTM_INDICES = 0x0601 ,
  CTM_VERTICES = 0x0602 ,
  CTM_NORMALS = 0x0603 ,
  CTM_UV_MAP_1 = 0x0700 ,
  CTM_UV_MAP_2 = 0x0701 ,
  CTM_UV_MAP_3 = 0x0702 ,
  CTM_UV_MAP_4 = 0x0703 ,
  CTM_UV_MAP_5 = 0x0704 ,
  CTM_UV_MAP_6 = 0x0705 ,
  CTM_UV_MAP_7 = 0x0706 ,
  CTM_UV_MAP_8 = 0x0707 ,
  CTM_ATTRIB_MAP_1 = 0x0800 ,
  CTM_ATTRIB_MAP_2 = 0x0801 ,
  CTM_ATTRIB_MAP_3 = 0x0802 ,
  CTM_ATTRIB_MAP_4 = 0x0803 ,
  CTM_ATTRIB_MAP_5 = 0x0804 ,
  CTM_ATTRIB_MAP_6 = 0x0805 ,
  CTM_ATTRIB_MAP_7 = 0x0806 ,
  CTM_ATTRIB_MAP_8 = 0x0807 }

  OpenCTM specific enumerators.

Functions

- CTMEXPORT CTMcontext CTMCALL ctmNewContext (CTMenum aMode)

  Create a new OpenCTM context.
- CTMEXPORT void CTMCALL ctmFreeContext (CTMcontext aContext)

Free an OpenCTM context.

- CTMEXPORT CTMenum CTMCALL ctmGetError (CTMcontext aContext)

  Returns the latest error.
- CTMEXPORT const char *CTMCALL ctmErrorString (CTMenum aError)

  Converts an OpenCTM error code to a zero-terminated string.
- CTMEXPORT CTMuint CTMCALL ctmGetInteger (CTMcontext aContext, CTMenum aProperty)

  Get information about an OpenCTM context.
- CTMEXPORT CTMfloat CTMCALL ctmGetFloat (CTMcontext aContext, CTMenum aProperty)

  Get information about an OpenCTM context.
- CTMEXPORT const CTMuint *CTMCALL ctmGetIntegerArray (CTMcontext aContext, CTMenum aProperty)

  Get an integer array from an OpenCTM context.
- CTMEXPORT const CTMfloat *CTMCALL ctmGetFloatArray (CTMcontext aContext, CTMenum aProperty)

  Get a floating point array from an OpenCTM context.
- CTMEXPORT CTMenum CTMCALL ctmGetNamedUVMap (CTMcontext aContext, const char *aName)

  Get a reference to the named UV map.
- CTMEXPORT const char *CTMCALL ctmGetUVMapString (CTMcontext aContext, CTMenum aUVMap, CTMenum aProperty)

  Get information about a UV map.
- CTMEXPORT CTMfloat CTMCALL ctmGetUVMapFloat (CTMcontext aContext, CTMenum aUVMap, CTMenum aProperty)

  Get information about a UV map.
- CTMEXPORT CTMenum CTMCALL ctmGetNamedAttribMap (CTMcontext aContext, const char *aName)

  Get a reference to the named vertex attribute map.
- CTMEXPORT const char *CTMCALL ctmGetAttribMapString (CTMcontext aContext, CTMenum aAttribMap, CTMenum aProperty)

  Get information about a vertex attribute map.
- CTMEXPORT CTMfloat CTMCALL ctmGetAttribMapFloat (CTMcontext aContext, CTMenum aAttribMap, CTMenum aProperty)

  Get information about a vertex attribute map.
- CTMEXPORT const char *CTMCALL ctmGetString (CTMcontext aContext, CTMenum aProperty)

  Get information about an OpenCTM context.
- CTMEXPORT void CTMCALL ctmCompressionMethod (CTMcontext aContext, CTMenum aMethod)

  Set which compression method to use for the given OpenCTM context.
- CTMEXPORT void CTMCALL ctmCompressionLevel (CTMcontext aContext, CTMuint aLevel)

  Set which LZMA compression level to use for the given OpenCTM context.
- CTMEXPORT void CTMCALL ctmVertexPrecision (CTMcontext aContext, CTMfloat aPrecision)

  Set the vertex coordinate precision (only used by the MG2 compression method).
- CTMEXPORT void CTMCALL ctmVertexPrecisionRel (CTMcontext aContext, CTMfloat aRelPrecision)

  Set the vertex coordinate precision, relative to the mesh dimensions (only used by the MG2 compression method).
- CTMEXPORT void CTMCALL ctmNormalPrecision (CTMcontext aContext, CTMfloat aPrecision)

  Set the normal precision (only used by the MG2 compression method).
- CTMEXPORT void CTMCALL ctmUVCoordPrecision (CTMcontext aContext, CTMenum aUVMap, CTMfloat aPrecision)

  Set the coordinate precision for the specified UV map (only used by the MG2 compression method).
- CTMEXPORT void CTMCALL ctmAttribPrecision (CTMcontext aContext, CTMenum aAttribMap, CTMfloat aPrecision)

  Set the attribute value precision for the specified attribute map (only used by the MG2 compression method).
- CTMEXPORT void CTMCALL ctmFileComment (CTMcontext aContext, const char *aFileComment)

  Set the file comment for the given OpenCTM context.
- CTMEXPORT void CTMCALL ctmDefineMesh (CTMcontext aContext, const CTMfloat *aVertices, CTMuint aVertexCount, const CTMuint *aIndices, CTMuint aTriangleCount, const CTMfloat *aNormals)

  Define a triangle mesh.

- CTMEXPORT CTMenum CTMCALL ctmAddUVMap (CTMcontext aContext, const CTMfloat *aUVCoords, const char *aName, const char *aFileName)

  Define a UV map.
- CTMEXPORT CTMenum CTMCALL ctmAddAttribMap (CTMcontext aContext, const CTMfloat *aAttrib↩ Values, const char *aName)

  Define a custom vertex attribute map.
- CTMEXPORT void CTMCALL ctmLoad (CTMcontext aContext, const char *aFileName)

  Load an OpenCTM format file into the context.
- CTMEXPORT void CTMCALL ctmLoadCustom (CTMcontext aContext, CTMreadfn aReadFn, void *aUser↩ Data)

  Load an OpenCTM format file using a custom stream read function.
- CTMEXPORT void CTMCALL ctmSave (CTMcontext aContext, const char *aFileName)

  Save an OpenCTM format file.
- CTMEXPORT void CTMCALL ctmSaveCustom (CTMcontext aContext, CTMwritefn aWriteFn, void *aUser↩ Data)

  Save an OpenCTM format file using a custom stream write function.

6.1.1 Macro Definition Documentation

6.1.1.1 CTM_API_VERSION

```
#define CTM_API_VERSION 0x00000100
```

OpenCTM API version (1.0).

6.1.1.2 CTM_FALSE

```
#define CTM_FALSE 0
```

Boolean FALSE.

6.1.1.3 CTM_TRUE

```
#define CTM_TRUE 1
```

Boolean TRUE.

6.1.2 Typedef Documentation

6.1.2.1 CTMcontext

```
typedef void* CTMcontext
```

OpenCTM context handle.

6.1.2.2 CTMfloat

```
typedef float CTMfloat
```

Single precision floating point type (IEEE 754 32 bits wide).

6.1.2.3 CTMint

```
typedef int32_t CTMint
```

Signed integer (32 bits wide).

6.1.2.4 CTMreadfn

```
typedef CTMuint(CTMCALL * CTMreadfn) (void *aBuf, CTMuint aCount, void *aUserData)
```

Stream read() function pointer.

Parameters

in	*aBuf*	Pointer to the memory buffer to which data should be read.
in	*aCount*	The number of bytes to read.
in	*aUserData*	The custom user data that was passed to the ctmLoadCustom() function.

Returns

The number of bytes actually read (if this is less than aCount, it indicates that an error occured or the end of file was reached before all bytes were read).

6.1.2.5 CTMuint

```
typedef uint32_t CTMuint
```

Unsigned integer (32 bits wide).

6.1.2.6 CTMwritefn

```
typedef CTMuint(CTMCALL * CTMwritefn) (const void *aBuf, CTMuint aCount, void *aUserData)
```

Stream write() function pointer.

Parameters

in	*aBuf*	Pointer to the memory buffer from which data should be written.
in	*aCount*	The number of bytes to write.
in	*aUserData*	The custom user data that was passed to the ctmSaveCustom() function.

Returns

The number of bytes actually written (if this is less than aCount, it indicates that an error occured).

6.1.3 Enumeration Type Documentation

6.1.3.1 CTMenum

`enum CTMenum`

OpenCTM specific enumerators.

Note

For the information query functions, it is an error to query a value of the wrong type (e.g. to query a string value with the ctmGetInteger() function).

Enumerator

CTM_NONE	No error has occured (everything is OK). Also used as an error return value for functions that should return a CTMenum value.
CTM_INVALID_CONTEXT	The OpenCTM context was invalid (e.g. NULL).
CTM_INVALID_ARGUMENT	A function argument was invalid.
CTM_INVALID_OPERATION	The operation is not allowed.
CTM_INVALID_MESH	The mesh was invalid (e.g. no vertices).
CTM_OUT_OF_MEMORY	Not enough memory to proceed.
CTM_FILE_ERROR	File I/O error.
CTM_BAD_FORMAT	File format error (e.g. unrecognized format or corrupted file).
CTM_LZMA_ERROR	An error occured within the LZMA library.
CTM_INTERNAL_ERROR	An internal error occured (indicates a bug).
CTM_UNSUPPORTED_FORMAT_VERSION	Unsupported file format version.
CTM_IMPORT	The OpenCTM context will be used for importing data.
CTM_EXPORT	The OpenCTM context will be used for exporting data.
CTM_METHOD_RAW	Just store the raw data.
CTM_METHOD_MG1	Lossless compression (floating point).
CTM_METHOD_MG2	Lossless compression (fixed point).
CTM_VERTEX_COUNT	Number of vertices in the mesh (integer).
CTM_TRIANGLE_COUNT	Number of triangles in the mesh (integer).
CTM_HAS_NORMALS	CTM_TRUE if the mesh has normals (integer).
CTM_UV_MAP_COUNT	Number of UV coordinate sets (integer).
CTM_ATTRIB_MAP_COUNT	Number of custom attribute sets (integer).
CTM_VERTEX_PRECISION	Vertex precision - for MG2 (float).
CTM_NORMAL_PRECISION	Normal precision - for MG2 (float).
CTM_COMPRESSION_METHOD	Compression method (integer).
CTM_FILE_COMMENT	File comment (string).
CTM_NAME	Unique name (UV/attrib map string).
CTM_FILE_NAME	File name reference (UV map string).
CTM_PRECISION	Value precision (UV/attrib map float).
CTM_INDICES	Triangle indices (integer array).
CTM_VERTICES	Vertex point coordinates (float array).
CTM_NORMALS	Per vertex normals (float array).
CTM_UV_MAP_1	Per vertex UV map 1 (float array).
CTM_UV_MAP_2	Per vertex UV map 2 (float array).

Enumerator

CTM_UV_MAP_3	Per vertex UV map 3 (float array).
CTM_UV_MAP_4	Per vertex UV map 4 (float array).
CTM_UV_MAP_5	Per vertex UV map 5 (float array).
CTM_UV_MAP_6	Per vertex UV map 6 (float array).
CTM_UV_MAP_7	Per vertex UV map 7 (float array).
CTM_UV_MAP_8	Per vertex UV map 8 (float array).
CTM_ATTRIB_MAP_1	Per vertex attribute map 1 (float array).
CTM_ATTRIB_MAP_2	Per vertex attribute map 2 (float array).
CTM_ATTRIB_MAP_3	Per vertex attribute map 3 (float array).
CTM_ATTRIB_MAP_4	Per vertex attribute map 4 (float array).
CTM_ATTRIB_MAP_5	Per vertex attribute map 5 (float array).
CTM_ATTRIB_MAP_6	Per vertex attribute map 6 (float array).
CTM_ATTRIB_MAP_7	Per vertex attribute map 7 (float array).
CTM_ATTRIB_MAP_8	Per vertex attribute map 8 (float array).

6.1.4 Function Documentation

6.1.4.1 ctmAddAttribMap()

```
CTMEXPORT CTMenum CTMCALL ctmAddAttribMap (
            CTMcontext aContext,
            const CTMfloat * aAttribValues,
            const char * aName)
```

Define a custom vertex attribute map.

Custom vertex attributes can be used for defining special per-vertex attributes, such as color, weight, ambient occlusion factor, etc.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aAttribValues*	An array of attribute values. Each attribute value is made up by four consecutive floats, and there must be as many values as there are vertices in the mesh.
in	*aName*	A unique name for this attribute map (zero terminated UTF-8 string).

Returns

A attribute map index (CTM_ATTRIB_MAP_1 and higher). If the function failed, it will return the zero valued CTM_NONE (use ctmGetError() to determine the cause of the error).

Note

A triangle mesh must have been defined before calling this function, since the number of vertices is defined by the triangle mesh.

See also

ctmDefineMesh().

6.1.4.2 ctmAddUVMap()

```
CTMEXPORT CTMenum CTMCALL ctmAddUVMap (
            CTMcontext aContext,
            const CTMfloat * aUVCoords,
            const char * aName,
            const char * aFileName)
```

Define a UV map.

There can be several UV maps in a mesh. A UV map is typically used for 2D texture mapping.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aUVCoords*	An array of UV coordinates. Each UV coordinate is made up by two consecutive floats, and there must be as many coordinates as there are vertices in the mesh.
in	*aName*	A unique name for this UV map (zero terminated UTF-8 string).
in	*aFileName*	A reference to a image file (zero terminated UTF-8 string). If no file name reference exists, pass NULL.

Returns

A UV map index (CTM_UV_MAP_1 and higher). If the function failed, it will return the zero valued CTM_NONE (use ctmGetError() to determine the cause of the error).

Note

A triangle mesh must have been defined before calling this function, since the number of vertices is defined by the triangle mesh.

See also

ctmDefineMesh().

6.1.4.3 ctmAttribPrecision()

```
CTMEXPORT void CTMCALL ctmAttribPrecision (
            CTMcontext aContext,
            CTMenum aAttribMap,
            CTMfloat aPrecision)
```

Set the attribute value precision for the specified attribute map (only used by the MG2 compression method).

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aAttribMap*	An attribute map specifier for a defined attribute map (CTM_ATTRIB_MAP_1, ...).
in	*aPrecision*	Fixed point precision. For instance, if this value is 0.001, all attribute values will be rounded to three decimals. If the attributes represent integer values, set the precision to 1.0. The default attribute precision is 2^-8 ~= 0.0039.

See also

ctmAddAttribMap().

6.1.4.4 ctmCompressionLevel()

```
CTMEXPORT void CTMCALL ctmCompressionLevel (
            CTMcontext aContext,
            CTMuint aLevel)
```

Set which LZMA compression level to use for the given OpenCTM context.

The compression level can be between 0 (fastest) and 9 (best). The higher the compression level, the more memory is required for compression and decompression. The default compression level is 1.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aLevel*	Which compression level to use (0 to 9).

6.1.4.5 ctmCompressionMethod()

```
CTMEXPORT void CTMCALL ctmCompressionMethod (
            CTMcontext aContext,
            CTMenum aMethod)
```

Set which compression method to use for the given OpenCTM context.

The selected compression method will be used when calling the ctmSave() function.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aMethod*	Which compression method to use: CTM_METHOD_RAW, CTM_METHOD_MG1 or CTM_METHOD_MG2 (the default method is CTM_METHOD_MG1).

See also

CTM_METHOD_RAW, CTM_METHOD_MG1, CTM_METHOD_MG2

6.1.4.6 ctmDefineMesh()

```
CTMEXPORT void CTMCALL ctmDefineMesh (
            CTMcontext aContext,
            const CTMfloat * aVertices,
            CTMuint aVertexCount,
            const CTMuint * aIndices,
            CTMuint aTriangleCount,
            const CTMfloat * aNormals)
```

Define a triangle mesh.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aVertices*	An array of vertices (three consecutive floats make one vertex).
in	*aVertexCount*	The number of vertices in `aVertices` (and optionally `aTexCoords`).
in	*aIndices*	An array of vertex indices (three consecutive integers make one triangle).
in	*aTriangleCount*	The number of triangles in `aIndices` (there must be exactly 3 x `aTriangleCount` indices in `aIndices`).
in	*aNormals*	An array of per-vertex normals (or NULL if there are no normals). Each normal is made up by three consecutive floats, and there must be `aVertexCount` normals.

See also

ctmAddUVMap(), ctmAddAttribMap(), ctmSave(), ctmSaveCustom().

6.1.4.7 ctmErrorString()

```
CTMEXPORT const char *CTMCALL ctmErrorString (
            CTMenum aError)
```

Converts an OpenCTM error code to a zero-terminated string.

Parameters

in	*aError*	An OpenCTM error code, as returned by ctmGetError().

Returns

A zero terminated string that describes the error. For instance, if `aError` is CTM_INVALID_OPERATION, then the return value will be "CTM_INVALID_OPERATION".

See also

CTMenum

6.1.4.8 ctmFileComment()

```
CTMEXPORT void CTMCALL ctmFileComment (
            CTMcontext aContext,
            const char * aFileComment)
```

Set the file comment for the given OpenCTM context.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aFileComment*	The file comment (zero terminated UTF-8 string).

6.1.4.9 ctmFreeContext()

```
CTMEXPORT void CTMCALL ctmFreeContext (
            CTMcontext aContext)
```

Free an OpenCTM context.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().

See also

ctmNewContext()

6.1.4.10 ctmGetAttribMapFloat()

```
CTMEXPORT CTMfloat CTMCALL ctmGetAttribMapFloat (
            CTMcontext aContext,
            CTMenum aAttribMap,
            CTMenum aProperty)
```

Get information about a vertex attribute map.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aAttribMap*	Which vertex attribute map to query (CTM_ATTRIB_MAP_1 or higher).
in	*aProperty*	Which vertex attribute map property to return.

Returns

A floating point value, representing the vertex attribute map property given by `aProperty`.

See also

CTMenum

6.1.4.11 ctmGetAttribMapString()

```
CTMEXPORT const char *CTMCALL ctmGetAttribMapString (
            CTMcontext aContext,
            CTMenum aAttribMap,
            CTMenum aProperty)
```

Get information about a vertex attribute map.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aAttribMap*	Which vertex attribute map to query (CTM_ATTRIB_MAP_1 or higher).
in	*aProperty*	Which vertex attribute map property to return.

Returns

A string value, representing the vertex attribute map property given by `aProperty`.

Note

The string is only valid as long as the vertex attribute map within the OpenCTM context is valid. Trying to access an invalid string will result in undefined behaviour. Therefor it is recommended that the string is copied to a new variable if it is to be used other than directly after the call to ctmGetAttribMapString().

See also

CTMenum

6.1.4.12 ctmGetError()

```
CTMEXPORT CTMenum CTMCALL ctmGetError (
            CTMcontext aContext)
```

Returns the latest error.

Calling this function will return the last produced error code, or CTM_NO_ERROR (zero) if no error has occured since the last call to ctmGetError(). When this function is called, the internal error varibale will be reset to CTM_NONE.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().

Returns

An OpenCTM error code.

See also

CTMenum

6.1.4.13 ctmGetFloat()

```
CTMEXPORT CTMfloat CTMCALL ctmGetFloat (
            CTMcontext aContext,
            CTMenum aProperty)
```

Get information about an OpenCTM context.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aProperty*	Which property to return.

Returns

A floating point value, representing the OpenCTM context property given by `aProperty`.

See also

CTMenum

6.1.4.14 ctmGetFloatArray()

```
CTMEXPORT const CTMfloat *CTMCALL ctmGetFloatArray (
            CTMcontext aContext,
            CTMenum aProperty)
```

Get a floating point array from an OpenCTM context.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aProperty*	Which array to return.

Returns

A floating point array. If the requested array does not exist, or if `aProperty` does not indicate a float array, the function returns NULL.

Note

The array is only valid as long as the OpenCTM context is valid, or until the corresponding array changes within the OpenCTM context. Trying to access an invalid array will result in undefined behaviour. Therefor it is recommended that the array is copied to a new variable if it is to be used other than directly after the call to ctmGetFloatArray().

See also

CTMenum

6.1.4.15 ctmGetInteger()

```
CTMEXPORT CTMuint CTMCALL ctmGetInteger (
            CTMcontext aContext,
            CTMenum aProperty)
```

Get information about an OpenCTM context.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aProperty*	Which property to return.

Returns

An integer value, representing the OpenCTM context property given by `aProperty`.

See also

CTMenum

6.1.4.16 ctmGetIntegerArray()

```
CTMEXPORT const CTMuint *CTMCALL ctmGetIntegerArray (
            CTMcontext aContext,
            CTMenum aProperty)
```

Get an integer array from an OpenCTM context.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aProperty*	Which array to return.

Returns

An integer array. If the requested array does not exist, or if `aProperty` does not indicate an integer array, the function returns NULL.

Note

The array is only valid as long as the OpenCTM context is valid, or until the corresponding array changes within the OpenCTM context. Trying to access an invalid array will result in undefined behaviour. Therefor it is recommended that the array is copied to a new variable if it is to be used other than directly after the call to ctmGetIntegerArray().

See also

CTMenum

6.1.4.17 ctmGetNamedAttribMap()

```
CTMEXPORT CTMenum CTMCALL ctmGetNamedAttribMap (
            CTMcontext aContext,
            const char * aName)
```

Get a reference to the named vertex attribute map.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aName*	The name of the attribute map that should be returned.

Returns

A reference to an attribute map. If the attribute map was found, a value of CTM_ATTRIB_MAP_1 or higher is returned, otherwise CTM_NONE is returned.

6.1.4.18 ctmGetNamedUVMap()

```
CTMEXPORT CTMenum CTMCALL ctmGetNamedUVMap (
            CTMcontext aContext,
            const char * aName)
```

Get a reference to the named UV map.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aName*	The name of the UV map that should be returned.

Returns

A reference to a UV map. If the UV map was found, a value of CTM_UV_MAP_1 or higher is returned, otherwise CTM_NONE is returned.

6.1.4.19 ctmGetString()

```
CTMEXPORT const char *CTMCALL ctmGetString (
            CTMcontext aContext,
            CTMenum aProperty)
```

Get information about an OpenCTM context.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aProperty*	Which property to return.

Returns

A string value, representing the OpenCTM context property given by `aProperty`.

Note

The string is only valid as long as the OpenCTM context is valid, or until the corresponding string changes within the OpenCTM context (e.g. calling ctmFileComment() invalidates the CTM_FILE_COMMENT string). Trying to access an invalid string will result in undefined behaviour. Therefor it is recommended that the string is copied to a new variable if it is to be used other than directly after the call to ctmGetString().

See also

CTMenum

6.1.4.20 ctmGetUVMapFloat()

```
CTMEXPORT CTMfloat CTMCALL ctmGetUVMapFloat (
            CTMcontext aContext,
            CTMenum aUVMap,
            CTMenum aProperty)
```

Get information about a UV map.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aUVMap*	Which UV map to query (CTM_UV_MAP_1 or higher).
in	*aProperty*	Which UV map property to return.

Returns

A floating point value, representing the UV map property given by `aProperty`.

See also

CTMenum

6.1.4.21 ctmGetUVMapString()

```
CTMEXPORT const char *CTMCALL ctmGetUVMapString (
            CTMcontext aContext,
            CTMenum aUVMap,
            CTMenum aProperty)
```

Get information about a UV map.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aUVMap*	Which UV map to query (CTM_UV_MAP_1 or higher).
in	*aProperty*	Which UV map property to return.

Returns

A string value, representing the UV map property given by `aProperty`.

Note

The string is only valid as long as the UV map within the OpenCTM context is valid. Trying to access an invalid string will result in undefined behaviour. Therefor it is recommended that the string is copied to a new variable if it is to be used other than directly after the call to ctmGetUVMapString().

See also

CTMenum

6.1.4.22 ctmLoad()

```
CTMEXPORT void CTMCALL ctmLoad (
            CTMcontext aContext,
            const char * aFileName)
```

Load an OpenCTM format file into the context.

The mesh data can be retrieved with the various ctmGet functions.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aFileName*	The name of the file to be loaded.

6.1.4.23 ctmLoadCustom()

```
CTMEXPORT void CTMCALL ctmLoadCustom (
            CTMcontext aContext,
            CTMreadfn aReadFn,
            void * aUserData)
```

Load an OpenCTM format file using a custom stream read function.

The mesh data can be retrieved with the various ctmGet functions.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aReadFn*	Pointer to a custom stream read function.
in	*aUserData*	Custom user data, which can be a C language FILE handle, C++ istream object, or a custom object pointer of any type. The user data pointer will be passed to the custom stream read function.

See also

CTMreadfn.

6.1.4.24 ctmNewContext()

```
CTMEXPORT CTMcontext CTMCALL ctmNewContext (
            CTMenum aMode)
```

Create a new OpenCTM context.

The context is used for all subsequent OpenCTM function calls. Several contexts can coexist at the same time.

Parameters

in	*aMode*	An OpenCTM context mode. Set this to CTM_IMPORT if the context will be used for importing data, or set it to CTM_EXPORT if it will be used for exporting data.

Returns

An OpenCTM context handle (or NULL if no context could be created).

6.1.4.25 ctmNormalPrecision()

```
CTMEXPORT void CTMCALL ctmNormalPrecision (
            CTMcontext aContext,
            CTMfloat aPrecision)
```

Set the normal precision (only used by the MG2 compression method).

The normal is represented in spherical coordinates in the MG2 compression method, and the normal precision controls the angular and radial resolution.

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aPrecision*	Fixed point precision. For the angular information, this value represents the angular precision. For the radial information, this value is the linear resolution. For instance, 0.01 means that the circle is divided into 100 steps, and the normal magnitude is rounded to 2 decimals. The default normal precision is 2^-8 ~= 0.0039.

6.1.4.26 ctmSave()

```
CTMEXPORT void CTMCALL ctmSave (
            CTMcontext aContext,
            const char * aFileName)
```

Save an OpenCTM format file.

The mesh must have been defined by ctmDefineMesh().

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aFileName*	The name of the file to be saved.

6.1.4.27 ctmSaveCustom()

```
CTMEXPORT void CTMCALL ctmSaveCustom (
            CTMcontext aContext,
            CTMwritefn aWriteFn,
            void * aUserData)
```

Save an OpenCTM format file using a custom stream write function.

The mesh must have been defined by ctmDefineMesh().

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aWriteFn*	Pointer to a custom stream write function.
in	*aUserData*	Custom user data, which can be a C language FILE handle, C++ ostream object, or a custom object pointer of any type. The user data pointer will be passed to the custom stream write function.

See also

CTMwritefn.

6.1.4.28 ctmUVCoordPrecision()

```
CTMEXPORT void CTMCALL ctmUVCoordPrecision (
            CTMcontext aContext,
            CTMenum aUVMap,
            CTMfloat aPrecision)
```

Set the coordinate precision for the specified UV map (only used by the MG2 compression method).

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aUVMap*	A UV map specifier for a defined UV map (CTM_UV_MAP_1, ...).
in	*aPrecision*	Fixed point precision. For instance, if this value is 0.001, all UV coordinates will be rounded to three decimals. The default UV coordinate precision is 2^-12 ~= 0.00024.

See also

ctmAddUVMap().

6.1.4.29 ctmVertexPrecision()

```
CTMEXPORT void CTMCALL ctmVertexPrecision (
            CTMcontext aContext,
            CTMfloat aPrecision)
```

Set the vertex coordinate precision (only used by the MG2 compression method).

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aPrecision*	Fixed point precision. For instance, if this value is 0.001, all vertex coordinates will be rounded to three decimals. The default vertex coordinate precision is 2^-10 ~= 0.00098.

6.1.4.30 ctmVertexPrecisionRel()

```
CTMEXPORT void CTMCALL ctmVertexPrecisionRel (
            CTMcontext aContext,
            CTMfloat aRelPrecision)
```

Set the vertex coordinate precision, relative to the mesh dimensions (only used by the MG2 compression method).

Parameters

in	*aContext*	An OpenCTM context that has been created by ctmNewContext().
in	*aRelPrecision*	Relative precision. This factor is multiplied by the average triangle edge length in the mesh in order to obtain the final, fixed point precision. For instance, if aRelPrecision is 0.01, and the average edge length is 3.7, then the fixed point precision is set to 0.037.

Note

The mesh must have been defined using the ctmDefineMesh() function before calling this function.

See also

ctmVertexPrecision().

6.2 openctm.h

Go to the documentation of this file.

```
00001 //-----------------------------------------------------------------------------
00002 // Product:     OpenCTM
00003 // File:        openctm.h
00004 // Description: OpenCTM API definition.
00005 //-----------------------------------------------------------------------------
00006 // Copyright (c) 2009-2010 Marcus Geelnard
00007 //
00008 // This software is provided 'as-is', without any express or implied
00009 // warranty. In no event will the authors be held liable for any damages
00010 // arising from the use of this software.
00011 //
00012 // Permission is granted to anyone to use this software for any purpose,
00013 // including commercial applications, and to alter it and redistribute it
00014 // freely, subject to the following restrictions:
00015 //
00016 //     1. The origin of this software must not be misrepresented; you must not
00017 //     claim that you wrote the original software. If you use this software
00018 //     in a product, an acknowledgment in the product documentation would be
00019 //     appreciated but is not required.
00020 //
00021 //     2. Altered source versions must be plainly marked as such, and must not
00022 //     be misrepresented as being the original software.
00023 //
00024 //     3. This notice may not be removed or altered from any source
00025 //     distribution.
00026 //-----------------------------------------------------------------------------
00027
00028 #ifndef __OPENCTM_H_
00029 #define __OPENCTM_H_
00030
00117 #ifdef __cplusplus
00118 extern "C" {
00119 #endif
00120
00121
00122 // Declare calling conventions etc.
00123 #if defined(WIN32) || defined(_WIN32)
00124   // Windows
00125   #if defined(OPENCTM_STATIC)
00126     #define CTMEXPORT
00127   #else
00128     #if defined(OPENCTM_BUILD)
00129       #define CTMEXPORT __declspec(dllexport)
00130     #else
00131       #define CTMEXPORT __declspec(dllimport)
00132     #endif
00133   #endif
00134   #if defined(__MINGW32__)
00135     #define CTMCALL __attribute__ ((__stdcall__))
00136   #elif (defined(_M_MRX000) || defined(_M_IX86) || defined(_M_ALPHA) || defined(_M_PPC)) &&
      !defined(MIDL_PASS)
00137     #define CTMCALL __stdcall
00138   #else
00139     #define CTMCALL
00140   #endif
00141 #else
00142   // Unix
00143   #if !defined(OPENCTM_STATIC) && !defined(OPENCTM_BUILD)
00144     #define CTMEXPORT extern
00145   #else
00146     #if defined(OPENCTM_BUILD) && defined(__GNUC__) && (__GNUC__ >= 4)
00147       #define CTMEXPORT __attribute__ ((visibility("default")))
00148     #else
00149       #define CTMEXPORT
00150     #endif
00151   #endif
00152   #define CTMCALL
00153 #endif
00154
00155
00156 // Get system specific type definitions for sized integers. We use the C99
00157 // standard stdint.h for this.
00158 #ifdef _MSC_VER
00159   // MS Visual Studio does not support C99
00160   typedef int int32_t;
00161   typedef unsigned int uint32_t;
00162 #else
00163   #include <stdint.h>
00164 #endif
00165
00166
00168 #define CTM_API_VERSION 0x00000100
```

```
00169
00171 #define CTM_TRUE 1
00172
00174 #define CTM_FALSE 0
00175
00177 typedef float CTMfloat;
00178
00180 typedef int32_t CTMint;
00181
00183 typedef uint32_t CTMuint;
00184
00186 typedef void * CTMcontext;
00187
00192 typedef enum {
00193   // Error codes (see ctmGetError())
00194   CTM_NONE              = 0x0000,
00198   CTM_INVALID_CONTEXT   = 0x0001,
00199   CTM_INVALID_ARGUMENT  = 0x0002,
00200   CTM_INVALID_OPERATION = 0x0003,
00201   CTM_INVALID_MESH      = 0x0004,
00202   CTM_OUT_OF_MEMORY     = 0x0005,
00203   CTM_FILE_ERROR        = 0x0006,
00204   CTM_BAD_FORMAT        = 0x0007,
00205   CTM_LZMA_ERROR        = 0x0008,
00206   CTM_INTERNAL_ERROR    = 0x0009,
00207   CTM_UNSUPPORTED_FORMAT_VERSION = 0x000A,
00208
00209   // OpenCTM context modes
00210   CTM_IMPORT            = 0x0101,
00211   CTM_EXPORT            = 0x0102,
00212
00213   // Compression methods
00214   CTM_METHOD_RAW        = 0x0201,
00215   CTM_METHOD_MG1        = 0x0202,
00216   CTM_METHOD_MG2        = 0x0203,
00217
00218   // Context queries
00219   CTM_VERTEX_COUNT      = 0x0301,
00220   CTM_TRIANGLE_COUNT    = 0x0302,
00221   CTM_HAS_NORMALS       = 0x0303,
00222   CTM_UV_MAP_COUNT      = 0x0304,
00223   CTM_ATTRIB_MAP_COUNT  = 0x0305,
00224   CTM_VERTEX_PRECISION  = 0x0306,
00225   CTM_NORMAL_PRECISION  = 0x0307,
00226   CTM_COMPRESSION_METHOD = 0x0308,
00227   CTM_FILE_COMMENT      = 0x0309,
00228
00229   // UV/attribute map queries
00230   CTM_NAME              = 0x0501,
00231   CTM_FILE_NAME         = 0x0502,
00232   CTM_PRECISION         = 0x0503,
00233
00234   // Array queries
00235   CTM_INDICES           = 0x0601,
00236   CTM_VERTICES          = 0x0602,
00237   CTM_NORMALS           = 0x0603,
00238   CTM_UV_MAP_1          = 0x0700,
00239   CTM_UV_MAP_2          = 0x0701,
00240   CTM_UV_MAP_3          = 0x0702,
00241   CTM_UV_MAP_4          = 0x0703,
00242   CTM_UV_MAP_5          = 0x0704,
00243   CTM_UV_MAP_6          = 0x0705,
00244   CTM_UV_MAP_7          = 0x0706,
00245   CTM_UV_MAP_8          = 0x0707,
00246   CTM_ATTRIB_MAP_1      = 0x0800,
00247   CTM_ATTRIB_MAP_2      = 0x0801,
00248   CTM_ATTRIB_MAP_3      = 0x0802,
00249   CTM_ATTRIB_MAP_4      = 0x0803,
00250   CTM_ATTRIB_MAP_5      = 0x0804,
00251   CTM_ATTRIB_MAP_6      = 0x0805,
00252   CTM_ATTRIB_MAP_7      = 0x0806,
00253   CTM_ATTRIB_MAP_8      = 0x0807
00254 } CTMenum;
00255
00264 typedef CTMuint (CTMCALL * CTMreadfn)(void * aBuf, CTMuint aCount, void * aUserData);
00265
00273 typedef CTMuint (CTMCALL * CTMwritefn)(const void * aBuf, CTMuint aCount, void * aUserData);
00274
00281 CTMEXPORT CTMcontext CTMCALL ctmNewContext(CTMenum aMode);
00282
00287 CTMEXPORT void CTMCALL ctmFreeContext(CTMcontext aContext);
00288
00297 CTMEXPORT CTMenum CTMCALL ctmGetError(CTMcontext aContext);
00298
00305 CTMEXPORT const char * CTMCALL ctmErrorString(CTMenum aError);
00306
00314 CTMEXPORT CTMuint CTMCALL ctmGetInteger(CTMcontext aContext, CTMenum aProperty);
```

```
00315
00323 CTMEXPORT CTMfloat CTMCALL ctmGetFloat(CTMcontext aContext, CTMenum aProperty);
00324
00339 CTMEXPORT const CTMuint * CTMCALL ctmGetIntegerArray(CTMcontext aContext,
00340   CTMenum aProperty);
00341
00356 CTMEXPORT const CTMfloat * CTMCALL ctmGetFloatArray(CTMcontext aContext,
00357   CTMenum aProperty);
00358
00366 CTMEXPORT CTMenum CTMCALL ctmGetNamedUVMap(CTMcontext aContext,
00367   const char * aName);
00368
00382 CTMEXPORT const char * CTMCALL ctmGetUVMapString(CTMcontext aContext,
00383   CTMenum aUVMap, CTMenum aProperty);
00384
00393 CTMEXPORT CTMfloat CTMCALL ctmGetUVMapFloat(CTMcontext aContext,
00394   CTMenum aUVMap, CTMenum aProperty);
00395
00403 CTMEXPORT CTMenum CTMCALL ctmGetNamedAttribMap(CTMcontext aContext,
00404   const char * aName);
00405
00420 CTMEXPORT const char * CTMCALL ctmGetAttribMapString(CTMcontext aContext,
00421   CTMenum aAttribMap, CTMenum aProperty);
00422
00432 CTMEXPORT CTMfloat CTMCALL ctmGetAttribMapFloat(CTMcontext aContext,
00433   CTMenum aAttribMap, CTMenum aProperty);
00434
00449 CTMEXPORT const char * CTMCALL ctmGetString(CTMcontext aContext,
00450   CTMenum aProperty);
00451
00461 CTMEXPORT void CTMCALL ctmCompressionMethod(CTMcontext aContext,
00462   CTMenum aMethod);
00463
00471 CTMEXPORT void CTMCALL ctmCompressionLevel(CTMcontext aContext,
00472   CTMuint aLevel);
00473
00481 CTMEXPORT void CTMCALL ctmVertexPrecision(CTMcontext aContext,
00482   CTMfloat aPrecision);
00483
00496 CTMEXPORT void CTMCALL ctmVertexPrecisionRel(CTMcontext aContext,
00497   CTMfloat aRelPrecision);
00498
00510 CTMEXPORT void CTMCALL ctmNormalPrecision(CTMcontext aContext,
00511   CTMfloat aPrecision);
00512
00523 CTMEXPORT void CTMCALL ctmUVCoordPrecision(CTMcontext aContext,
00524   CTMenum aUVMap, CTMfloat aPrecision);
00525
00537 CTMEXPORT void CTMCALL ctmAttribPrecision(CTMcontext aContext,
00538   CTMenum aAttribMap, CTMfloat aPrecision);
00539
00544 CTMEXPORT void CTMCALL ctmFileComment(CTMcontext aContext,
00545   const char * aFileComment);
00546
00562 CTMEXPORT void CTMCALL ctmDefineMesh(CTMcontext aContext,
00563   const CTMfloat * aVertices, CTMuint aVertexCount, const CTMuint * aIndices,
00564   CTMuint aTriangleCount, const CTMfloat * aNormals);
00565
00583 CTMEXPORT CTMenum CTMCALL ctmAddUVMap(CTMcontext aContext,
00584   const CTMfloat * aUVCoords, const char * aName, const char * aFileName);
00585
00602 CTMEXPORT CTMenum CTMCALL ctmAddAttribMap(CTMcontext aContext,
00603   const CTMfloat * aAttribValues, const char * aName);
00604
00610 CTMEXPORT void CTMCALL ctmLoad(CTMcontext aContext, const char * aFileName);
00611
00622 CTMEXPORT void CTMCALL ctmLoadCustom(CTMcontext aContext, CTMreadfn aReadFn,
00623   void * aUserData);
00624
00630 CTMEXPORT void CTMCALL ctmSave(CTMcontext aContext, const char * aFileName);
00631
00642 CTMEXPORT void CTMCALL ctmSaveCustom(CTMcontext aContext, CTMwritefn aWriteFn,
00643   void * aUserData);
00644
00645 #ifdef __cplusplus
00646 }
00647 #endif
00648
00649
00650 // C++ extensions to the API (to disable C++ extensions, define OPENCTM_NO_CPP)
00651 #if defined(__cplusplus) && !defined(OPENCTM_NO_CPP)
00652   #include "openctmpp.h"
00653 #endif
00654
00655 #endif // __OPENCTM_H_
```

6.3 openctmpp.h File Reference

```
#include "openctm.h"
#include <exception>
```

Classes

- class ctm_error

  OpenCTM exception.
- class CTMimporter

  OpenCTM importer class.
- class CTMexporter

  OpenCTM exporter class.

6.4 openctmpp.h

Go to the documentation of this file.

```
00001 //-----------------------------------------------------------------------------
00002 // Product:     OpenCTM
00003 // File:        openctmpp.h
00004 // Description: C++ wrapper for the OpenCTM API.
00005 //-----------------------------------------------------------------------------
00006 // Copyright (c) 2009-2010 Marcus Geelnard
00007 //
00008 // This software is provided 'as-is', without any express or implied
00009 // warranty. In no event will the authors be held liable for any damages
00010 // arising from the use of this software.
00011 //
00012 // Permission is granted to anyone to use this software for any purpose,
00013 // including commercial applications, and to alter it and redistribute it
00014 // freely, subject to the following restrictions:
00015 //
00016 //     1. The origin of this software must not be misrepresented; you must not
00017 //     claim that you wrote the original software. If you use this software
00018 //     in a product, an acknowledgment in the product documentation would be
00019 //     appreciated but is not required.
00020 //
00021 //     2. Altered source versions must be plainly marked as such, and must not
00022 //     be misrepresented as being the original software.
00023 //
00024 //     3. This notice may not be removed or altered from any source
00025 //     distribution.
00026 //-----------------------------------------------------------------------------
00027
00028 // To disable C++ extensions, define OPENCTM_NO_CPP
00029 #ifndef OPENCTM_NO_CPP
00030
00031 #ifndef __OPENCTMPP_H_
00032 #define __OPENCTMPP_H_
00033
00034 // Just in case (if this file was included from outside openctm.h)...
00035 #ifndef __OPENCTM_H_
00036 #include "openctm.h"
00037 #endif
00038
00039 #include <exception>
00040
00044 class ctm_error: public std::exception
00045 {
00046   private:
00047     CTMenum mErrorCode;
00048
00049   public:
00050     explicit ctm_error(CTMenum aError)
00051     {
00052       mErrorCode = aError;
00053     }
00054
00055     virtual const char* what() const throw()
```

```
00056     {
00057       return ctmErrorString(mErrorCode);
00058     }
00059
00060     CTMenum error_code() const throw()
00061     {
00062       return mErrorCode;
00063     }
00064 };
00065
00066
00086
00087 class CTMimporter {
00088   private:
00090     CTMcontext mContext;
00091
00094     void CheckError()
00095     {
00096       CTMenum err = ctmGetError(mContext);
00097       if(err != CTM_NONE)
00098         throw ctm_error(err);
00099     }
00100
00101   public:
00103     CTMimporter()
00104     {
00105       mContext = ctmNewContext(CTM_IMPORT);
00106     }
00107
00109     ~CTMimporter()
00110     {
00111       ctmFreeContext(mContext);
00112     }
00113
00115     CTMuint GetInteger(CTMenum aProperty)
00116     {
00117       CTMuint res = ctmGetInteger(mContext, aProperty);
00118       CheckError();
00119       return res;
00120     }
00121
00123     CTMfloat GetFloat(CTMenum aProperty)
00124     {
00125       CTMfloat res = ctmGetFloat(mContext, aProperty);
00126       CheckError();
00127       return res;
00128     }
00129
00131     const CTMuint * GetIntegerArray(CTMenum aProperty)
00132     {
00133       const CTMuint * res = ctmGetIntegerArray(mContext, aProperty);
00134       CheckError();
00135       return res;
00136     }
00137
00139     const CTMfloat * GetFloatArray(CTMenum aProperty)
00140     {
00141       const CTMfloat * res = ctmGetFloatArray(mContext, aProperty);
00142       CheckError();
00143       return res;
00144     }
00145
00147     CTMenum GetNamedUVMap(const char * aName)
00148     {
00149       CTMenum res = ctmGetNamedUVMap(mContext, aName);
00150       CheckError();
00151       return res;
00152     }
00153
00155     const char * GetUVMapString(CTMenum aUVMap, CTMenum aProperty)
00156     {
00157       const char * res = ctmGetUVMapString(mContext, aUVMap, aProperty);
00158       CheckError();
00159       return res;
00160     }
00161
00163     CTMfloat GetUVMapFloat(CTMenum aUVMap, CTMenum aProperty)
00164     {
00165       CTMfloat res = ctmGetUVMapFloat(mContext, aUVMap, aProperty);
00166       CheckError();
00167       return res;
00168     }
00169
00171     CTMenum GetNamedAttribMap(const char * aName)
00172     {
00173       CTMenum res = ctmGetNamedAttribMap(mContext, aName);
00174       CheckError();
```

```
00175       return res;
00176     }
00177
00179     const char * GetAttribMapString(CTMenum aAttribMap, CTMenum aProperty)
00180     {
00181       const char * res = ctmGetAttribMapString(mContext, aAttribMap, aProperty);
00182       CheckError();
00183       return res;
00184     }
00185
00187     CTMfloat GetAttribMapFloat(CTMenum aAttribMap, CTMenum aProperty)
00188     {
00189       CTMfloat res = ctmGetAttribMapFloat(mContext, aAttribMap, aProperty);
00190       CheckError();
00191       return res;
00192     }
00193
00195     const char * GetString(CTMenum aProperty)
00196     {
00197       const char * res = ctmGetString(mContext, aProperty);
00198       CheckError();
00199       return res;
00200     }
00201
00203     void Load(const char * aFileName)
00204     {
00205       ctmLoad(mContext, aFileName);
00206       CheckError();
00207     }
00208
00210     void LoadCustom(CTMreadfn aReadFn, void * aUserData)
00211     {
00212       ctmLoadCustom(mContext, aReadFn, aUserData);
00213       CheckError();
00214     }
00215
00216     // You can not copy nor assign from one CTMimporter object to another, since
00217     // the object contains hidden state. By declaring these dummy prototypes
00218     // without an implementation, you will at least get linker errors if you try
00219     // to copy or assign a CTMimporter object.
00220     CTMimporter(const CTMimporter& v);
00221     CTMimporter& operator=(const CTMimporter& v);
00222 };
00223
00224
00242
00243 class CTMexporter {
00244   private:
00246     CTMcontext mContext;
00247
00250     void CheckError()
00251     {
00252       CTMenum err = ctmGetError(mContext);
00253       if(err != CTM_NONE)
00254         throw ctm_error(err);
00255     }
00256
00257   public:
00259     CTMexporter()
00260     {
00261       mContext = ctmNewContext(CTM_EXPORT);
00262     }
00263
00265     ~CTMexporter()
00266     {
00267       ctmFreeContext(mContext);
00268     }
00269
00271     void CompressionMethod(CTMenum aMethod)
00272     {
00273       ctmCompressionMethod(mContext, aMethod);
00274       CheckError();
00275     }
00276
00278     void CompressionLevel(CTMuint aLevel)
00279     {
00280       ctmCompressionLevel(mContext, aLevel);
00281       CheckError();
00282     }
00283
00285     void VertexPrecision(CTMfloat aPrecision)
00286     {
00287       ctmVertexPrecision(mContext, aPrecision);
00288       CheckError();
00289     }
00290
00292     void VertexPrecisionRel(CTMfloat aRelPrecision)
```

```
00293     {
00294       ctmVertexPrecisionRel(mContext, aRelPrecision);
00295       CheckError();
00296     }
00297
00299     void NormalPrecision(CTMfloat aPrecision)
00300     {
00301       ctmNormalPrecision(mContext, aPrecision);
00302       CheckError();
00303     }
00304
00306     void UVCoordPrecision(CTMenum aUVMap, CTMfloat aPrecision)
00307     {
00308       ctmUVCoordPrecision(mContext, aUVMap, aPrecision);
00309       CheckError();
00310     }
00311
00313     void AttribPrecision(CTMenum aAttribMap, CTMfloat aPrecision)
00314     {
00315       ctmAttribPrecision(mContext, aAttribMap, aPrecision);
00316       CheckError();
00317     }
00318
00320     void FileComment(const char * aFileComment)
00321     {
00322       ctmFileComment(mContext, aFileComment);
00323       CheckError();
00324     }
00325
00327     void DefineMesh(const CTMfloat * aVertices, CTMuint aVertexCount,
00328       const CTMuint * aIndices, CTMuint aTriangleCount,
00329       const CTMfloat * aNormals)
00330     {
00331       ctmDefineMesh(mContext, aVertices, aVertexCount, aIndices, aTriangleCount,
00332                     aNormals);
00333       CheckError();
00334     }
00335
00337     CTMenum AddUVMap(const CTMfloat * aUVCoords, const char * aName,
00338       const char * aFileName)
00339     {
00340       CTMenum res = ctmAddUVMap(mContext, aUVCoords, aName, aFileName);
00341       CheckError();
00342       return res;
00343     }
00344
00346     CTMenum AddAttribMap(const CTMfloat * aAttribValues, const char * aName)
00347     {
00348       CTMenum res = ctmAddAttribMap(mContext, aAttribValues, aName);
00349       CheckError();
00350       return res;
00351     }
00352
00354     void Save(const char * aFileName)
00355     {
00356       ctmSave(mContext, aFileName);
00357       CheckError();
00358     }
00359
00361     void SaveCustom(CTMwritefn aWriteFn, void * aUserData)
00362     {
00363       ctmSaveCustom(mContext, aWriteFn, aUserData);
00364       CheckError();
00365     }
00366
00367     // You can not copy nor assign from one CTMexporter object to another, since
00368     // the object contains hidden state. By declaring these dummy prototypes
00369     // without an implementation, you will at least get linker errors if you try
00370     // to copy or assign a CTMexporter object.
00371     CTMexporter(const CTMexporter& v);
00372     CTMexporter& operator=(const CTMexporter& v);
00373 };
00374
00375 #endif // __OPENCTMPP_H_
00376
00377 #endif // OPENCTM_NO_CPP
```

Index

~CTMexporter
    CTMexporter, 11
~CTMimporter
    CTMimporter, 15

AddAttribMap
    CTMexporter, 11
AddUVMap
    CTMexporter, 11
AttribPrecision
    CTMexporter, 12

CompressionLevel
    CTMexporter, 12
CompressionMethod
    CTMexporter, 12
CTM_API_VERSION
    openctm.h, 22
CTM_ATTRIB_MAP_1
    openctm.h, 25
CTM_ATTRIB_MAP_2
    openctm.h, 25
CTM_ATTRIB_MAP_3
    openctm.h, 25
CTM_ATTRIB_MAP_4
    openctm.h, 25
CTM_ATTRIB_MAP_5
    openctm.h, 25
CTM_ATTRIB_MAP_6
    openctm.h, 25
CTM_ATTRIB_MAP_7
    openctm.h, 25
CTM_ATTRIB_MAP_8
    openctm.h, 25
CTM_ATTRIB_MAP_COUNT
    openctm.h, 24
CTM_BAD_FORMAT
    openctm.h, 24
CTM_COMPRESSION_METHOD
    openctm.h, 24
ctm_error, 9
    ctm_error, 9
    error_code, 10
    what, 10
CTM_EXPORT
    openctm.h, 24
CTM_FALSE
    openctm.h, 22
CTM_FILE_COMMENT
    openctm.h, 24
CTM_FILE_ERROR
    openctm.h, 24
CTM_FILE_NAME
    openctm.h, 24
CTM_HAS_NORMALS
    openctm.h, 24
CTM_IMPORT
    openctm.h, 24
CTM_INDICES
    openctm.h, 24
CTM_INTERNAL_ERROR
    openctm.h, 24
CTM_INVALID_ARGUMENT
    openctm.h, 24
CTM_INVALID_CONTEXT
    openctm.h, 24
CTM_INVALID_MESH
    openctm.h, 24
CTM_INVALID_OPERATION
    openctm.h, 24
CTM_LZMA_ERROR
    openctm.h, 24
CTM_METHOD_MG1
    openctm.h, 24
CTM_METHOD_MG2
    openctm.h, 24
CTM_METHOD_RAW
    openctm.h, 24
CTM_NAME
    openctm.h, 24
CTM_NONE
    openctm.h, 24
CTM_NORMAL_PRECISION
    openctm.h, 24
CTM_NORMALS
    openctm.h, 24
CTM_OUT_OF_MEMORY
    openctm.h, 24
CTM_PRECISION
    openctm.h, 24
CTM_TRIANGLE_COUNT
    openctm.h, 24
CTM_TRUE
    openctm.h, 22
CTM_UNSUPPORTED_FORMAT_VERSION
    openctm.h, 24
CTM_UV_MAP_1
    openctm.h, 24
CTM_UV_MAP_2

openctm.h, 24
CTM_UV_MAP_3
openctm.h, 25
CTM_UV_MAP_4
openctm.h, 25
CTM_UV_MAP_5
openctm.h, 25
CTM_UV_MAP_6
openctm.h, 25
CTM_UV_MAP_7
openctm.h, 25
CTM_UV_MAP_8
openctm.h, 25
CTM_UV_MAP_COUNT
openctm.h, 24
CTM_VERTEX_COUNT
openctm.h, 24
CTM_VERTEX_PRECISION
openctm.h, 24
CTM_VERTICES
openctm.h, 24
ctmAddAttribMap
openctm.h, 25
ctmAddUVMap
openctm.h, 25
ctmAttribPrecision
openctm.h, 26
ctmCompressionLevel
openctm.h, 26
ctmCompressionMethod
openctm.h, 27
CTMcontext
openctm.h, 22
ctmDefineMesh
openctm.h, 27
CTMenum
openctm.h, 24
ctmErrorString
openctm.h, 28
CTMexporter, 10
~CTMexporter, 11
AddAttribMap, 11
AddUVMap, 11
AttribPrecision, 12
CompressionLevel, 12
CompressionMethod, 12
CTMexporter, 11
DefineMesh, 12
FileComment, 12
NormalPrecision, 12
operator=, 13
Save, 13
SaveCustom, 13
UVCoordPrecision, 13
VertexPrecision, 13
VertexPrecisionRel, 13
ctmFileComment
openctm.h, 28
CTMfloat
openctm.h, 22
ctmFreeContext
openctm.h, 28
ctmGetAttribMapFloat
openctm.h, 29
ctmGetAttribMapString
openctm.h, 29
ctmGetError
openctm.h, 30
ctmGetFloat
openctm.h, 30
ctmGetFloatArray
openctm.h, 30
ctmGetInteger
openctm.h, 31
ctmGetIntegerArray
openctm.h, 31
ctmGetNamedAttribMap
openctm.h, 32
ctmGetNamedUVMap
openctm.h, 32
ctmGetString
openctm.h, 33
ctmGetUVMapFloat
openctm.h, 33
ctmGetUVMapString
openctm.h, 34
CTMimporter, 14
~CTMimporter, 15
CTMimporter, 15
GetAttribMapFloat, 15
GetAttribMapString, 15
GetFloat, 15
GetFloatArray, 15
GetInteger, 16
GetIntegerArray, 16
GetNamedAttribMap, 16
GetNamedUVMap, 16
GetString, 16
GetUVMapFloat, 16
GetUVMapString, 16
Load, 17
LoadCustom, 17
operator=, 17
CTMint
openctm.h, 23
ctmLoad
openctm.h, 34
ctmLoadCustom
openctm.h, 35
ctmNewContext
openctm.h, 35
ctmNormalPrecision
openctm.h, 35
CTMreadfn
openctm.h, 23
ctmSave

openctm.h, 36
ctmSaveCustom
openctm.h, 36
CTMuint
openctm.h, 23
ctmUVCoordPrecision
openctm.h, 36
ctmVertexPrecision
openctm.h, 37
ctmVertexPrecisionRel
openctm.h, 37
CTMwritefn
openctm.h, 23

DefineMesh
CTMexporter, 12

error_code
ctm_error, 10

FileComment
CTMexporter, 12

GetAttribMapFloat
CTMimporter, 15
GetAttribMapString
CTMimporter, 15
GetFloat
CTMimporter, 15
GetFloatArray
CTMimporter, 15
GetInteger
CTMimporter, 16
GetIntegerArray
CTMimporter, 16
GetNamedAttribMap
CTMimporter, 16
GetNamedUVMap
CTMimporter, 16
GetString
CTMimporter, 16
GetUVMapFloat
CTMimporter, 16
GetUVMapString
CTMimporter, 16

Load
CTMimporter, 17
LoadCustom
CTMimporter, 17

NormalPrecision
CTMexporter, 12

OpenCTM API Reference, 1
openctm.h, 19, 38
CTM_API_VERSION, 22
CTM_ATTRIB_MAP_1, 25
CTM_ATTRIB_MAP_2, 25
CTM_ATTRIB_MAP_3, 25
CTM_ATTRIB_MAP_4, 25
CTM_ATTRIB_MAP_5, 25
CTM_ATTRIB_MAP_6, 25
CTM_ATTRIB_MAP_7, 25
CTM_ATTRIB_MAP_8, 25
CTM_ATTRIB_MAP_COUNT, 24
CTM_BAD_FORMAT, 24
CTM_COMPRESSION_METHOD, 24
CTM_EXPORT, 24
CTM_FALSE, 22
CTM_FILE_COMMENT, 24
CTM_FILE_ERROR, 24
CTM_FILE_NAME, 24
CTM_HAS_NORMALS, 24
CTM_IMPORT, 24
CTM_INDICES, 24
CTM_INTERNAL_ERROR, 24
CTM_INVALID_ARGUMENT, 24
CTM_INVALID_CONTEXT, 24
CTM_INVALID_MESH, 24
CTM_INVALID_OPERATION, 24
CTM_LZMA_ERROR, 24
CTM_METHOD_MG1, 24
CTM_METHOD_MG2, 24
CTM_METHOD_RAW, 24
CTM_NAME, 24
CTM_NONE, 24
CTM_NORMAL_PRECISION, 24
CTM_NORMALS, 24
CTM_OUT_OF_MEMORY, 24
CTM_PRECISION, 24
CTM_TRIANGLE_COUNT, 24
CTM_TRUE, 22
CTM_UNSUPPORTED_FORMAT_VERSION, 24
CTM_UV_MAP_1, 24
CTM_UV_MAP_2, 24
CTM_UV_MAP_3, 25
CTM_UV_MAP_4, 25
CTM_UV_MAP_5, 25
CTM_UV_MAP_6, 25
CTM_UV_MAP_7, 25
CTM_UV_MAP_8, 25
CTM_UV_MAP_COUNT, 24
CTM_VERTEX_COUNT, 24
CTM_VERTEX_PRECISION, 24
CTM_VERTICES, 24
ctmAddAttribMap, 25
ctmAddUVMap, 25
ctmAttribPrecision, 26
ctmCompressionLevel, 26
ctmCompressionMethod, 27
CTMcontext, 22
ctmDefineMesh, 27
CTMenum, 24
ctmErrorString, 28
ctmFileComment, 28
CTMfloat, 22
ctmFreeContext, 28

ctmGetAttribMapFloat, 29
ctmGetAttribMapString, 29
ctmGetError, 30
ctmGetFloat, 30
ctmGetFloatArray, 30
ctmGetInteger, 31
ctmGetIntegerArray, 31
ctmGetNamedAttribMap, 32
ctmGetNamedUVMap, 32
ctmGetString, 33
ctmGetUVMapFloat, 33
ctmGetUVMapString, 34
CTMint, 23
ctmLoad, 34
ctmLoadCustom, 35
ctmNewContext, 35
ctmNormalPrecision, 35
CTMreadfn, 23
ctmSave, 36
ctmSaveCustom, 36
CTMuint, 23
ctmUVCoordPrecision, 36
ctmVertexPrecision, 37
ctmVertexPrecisionRel, 37
CTMwritefn, 23
openctmpp.h, 41
operator=
CTMexporter, 13
CTMimporter, 17

Save
CTMexporter, 13
SaveCustom
CTMexporter, 13

UVCoordPrecision
CTMexporter, 13

VertexPrecision
CTMexporter, 13
VertexPrecisionRel
CTMexporter, 13

what
ctm_error, 10