

OpenCTM

1.0.3

Generated by Doxygen 1.9.7

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 <code>ctm_error</code> Class Reference	9
5.1.1 Detailed Description	9
5.1.2 Constructor & Destructor Documentation	9
5.1.2.1 <code>ctm_error()</code>	9
5.1.3 Member Function Documentation	10
5.1.3.1 <code>error_code()</code>	10
5.1.3.2 <code>what()</code>	10
5.2 <code>CTMExporter</code> Class Reference	10
5.2.1 Detailed Description	11
5.2.2 Constructor & Destructor Documentation	11
5.2.2.1 <code>CTMExporter()</code> [1/2]	11
5.2.2.2 <code>~CTMExporter()</code>	11
5.2.2.3 <code>CTMExporter()</code> [2/2]	11
5.2.3 Member Function Documentation	11
5.2.3.1 <code>AddAttribMap()</code>	11
5.2.3.2 <code>AddUVMap()</code>	12
5.2.3.3 <code>AttribPrecision()</code>	12
5.2.3.4 <code>CompressionLevel()</code>	12
5.2.3.5 <code>CompressionMethod()</code>	12
5.2.3.6 <code>DefineMesh()</code>	12
5.2.3.7 <code>FileComment()</code>	12
5.2.3.8 <code>NormalPrecision()</code>	13
5.2.3.9 <code>operator=()</code>	13
5.2.3.10 <code>Save()</code>	13
5.2.3.11 <code>SaveCustom()</code>	13
5.2.3.12 <code>UVCoordPrecision()</code>	13
5.2.3.13 <code>VertexPrecision()</code>	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 <code>ctmCompressionLevel()</code>	27
6.1.4.5 <code>ctmCompressionMethod()</code>	27
6.1.4.6 <code>ctmDefineMesh()</code>	28
6.1.4.7 <code>ctmErrorString()</code>	28
6.1.4.8 <code>ctmFileComment()</code>	29
6.1.4.9 <code>ctmFreeContext()</code>	29
6.1.4.10 <code>ctmGetAttribMapFloat()</code>	29
6.1.4.11 <code>ctmGetAttribMapString()</code>	30
6.1.4.12 <code>ctmGetError()</code>	30
6.1.4.13 <code>ctmGetFloat()</code>	31
6.1.4.14 <code>ctmGetFloatArray()</code>	31
6.1.4.15 <code>ctmGetInteger()</code>	32
6.1.4.16 <code>ctmGetIntegerArray()</code>	32
6.1.4.17 <code>ctmGetNamedAttribMap()</code>	33
6.1.4.18 <code>ctmGetNamedUVMap()</code>	33
6.1.4.19 <code>ctmGetString()</code>	34
6.1.4.20 <code>ctmGetUVMapFloat()</code>	34
6.1.4.21 <code>ctmGetUVMapString()</code>	35
6.1.4.22 <code>ctmLoad()</code>	35
6.1.4.23 <code>ctmLoadCustom()</code>	35
6.1.4.24 <code>ctmNewContext()</code>	36
6.1.4.25 <code>ctmNormalPrecision()</code>	36
6.1.4.26 <code>ctmSave()</code>	37
6.1.4.27 <code>ctmSaveCustom()</code>	37
6.1.4.28 <code>ctmUVCoordPrecision()</code>	37
6.1.4.29 <code>ctmVertexPrecision()</code>	38
6.1.4.30 <code>ctmVertexPrecisionRel()</code>	38
6.2 <code>openctm.h</code>	39
6.3 <code>openctmpp.h</code> File Reference	42
6.4 <code>openctmpp.h</code>	42
Index	47

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	42

Chapter 5

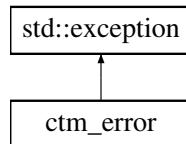
Class Documentation

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` *aAttribValues, 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 *aUserData)

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 *aUserData)

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 0x000000100
```

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	<i>aBuf</i>	Pointer to the memory buffer to which data should be read.
in	<i>aCount</i>	The number of bytes to read.
in	<i>aUserData</i>	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 occurred 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	<i>aBuf</i>	Pointer to the memory buffer from which data should be written.
in	<i>aCount</i>	The number of bytes to write.
in	<i>aUserData</i>	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 occurred).

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

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

Enumerator

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).
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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aAttribValues</i>	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	<i>aName</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aUVCoords</i>	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	<i>aName</i>	A unique name for this UV map (zero terminated UTF-8 string).
in	<i>aFileName</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aAttribMap</i>	An attribute map specifier for a defined attribute map (CTM_ATTRIB_MAP_1, ...).
in	<i>aPrecision</i>	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 $2^{-8} \approx 0.0039$.

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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aLevel</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aMethod</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aVertices</i>	An array of vertices (three consecutive floats make one vertex).
in	<i>aVertexCount</i>	The number of vertices in <i>aVertices</i> (and optionally <i>aTexCoords</i>).
in	<i>aIndices</i>	An array of vertex indices (three consecutive integers make one triangle).
in	<i>aTriangleCount</i>	The number of triangles in <i>aIndices</i> (there must be exactly 3 x <i>aTriangleCount</i> indices in <i>aIndices</i>).
in	<i>aNormals</i>	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 <i>aVertexCount</i> 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	<i>aError</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aFileComment</i>	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	<i>aContext</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aAttribMap</i>	Which vertex attribute map to query (CTM_ATTRIB_MAP_1 or higher).
in	<i>aProperty</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aAttribMap</i>	Which vertex attribute map to query (CTM_ATTRIB_MAP_1 or higher).
in	<i>aProperty</i>	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 occurred since the last call to [ctmGetError\(\)](#). When this function is called, the internal error varibale will be reset to CTM_NONE.

Parameters

in	<i>aContext</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aProperty</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aProperty</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aProperty</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aProperty</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aName</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aName</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aProperty</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aUVMap</i>	Which UV map to query (CTM_UV_MAP_1 or higher).
in	<i>aProperty</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aUVMap</i>	Which UV map to query (CTM_UV_MAP_1 or higher).
in	<i>aProperty</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aFileName</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aReadFn</i>	Pointer to a custom stream read function.
in	<i>aUserData</i>	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	<i>aMode</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aPrecision</i>	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 $2^{-8} \approx 0.0039$.

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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aFileName</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aWriteFn</i>	Pointer to a custom stream write function.
in	<i>aUserData</i>	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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aUVMap</i>	A UV map specifier for a defined UV map (CTM_UV_MAP_1, ...).
in	<i>aPrecision</i>	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 $2^{-12} \approx 0.00024$.

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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aPrecision</i>	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 $2^{-10} \approx 0.00098$.

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	<i>aContext</i>	An OpenCTM context that has been created by ctmNewContext() .
in	<i>aRelPrecision</i>	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 <i>aRelPrecision</i> 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
00031 #ifdef __cplusplus
00032     extern "C" {
00033 #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)) &&
00137         !defined(MIDL_PASS)
00138         #define CTMCALL __stdcall
00139     #else
00140         #define CTMCALL
00141     #endif
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,
00195     CTM_INVALID_CONTEXT = 0x0001,
00196     CTM_INVALID_ARGUMENT = 0x0002,
00197     CTM_INVALID_OPERATION = 0x0003,
00198     CTM_INVALID_MESH    = 0x0004,
00199     CTM_OUT_OF_MEMORY   = 0x0005,
00200     CTM_FILE_ERROR      = 0x0006,
00201     CTM_BAD_FORMAT      = 0x0007,
00202     CTM_LZMA_ERROR      = 0x0008,
00203     CTM_INTERNAL_ERROR   = 0x0009,
00204     CTM_UNSUPPORTED_FORMAT_VERSION = 0x000A,
00205
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
00041 class ctm_error: public std::exception
00042 {

```

```
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
00067
00068 class CTMimporter {
00069     private:
00070         CTMcontext mContext;
00071
00072     void CheckError()
00073     {
00074         CTMenum err = ctmGetError(mContext);
00075         if(err != CTM_NONE)
00076             throw ctm_error(err);
00077     }
00078
00079     public:
00080         CTMimporter()
00081         {
00082             mContext = ctmNewContext(CTM_IMPORT);
00083         }
00084
00085         ~CTMimporter()
00086         {
00087             ctmFreeContext(mContext);
00088         }
00089
00090         CTMuint GetInteger(CTMenum aProperty)
00091         {
00092             CTMuint res = ctmGetInteger(mContext, aProperty);
00093             CheckError();
00094             return res;
00095         }
00096
00097         CTMfloat GetFloat(CTMenum aProperty)
00098         {
00099             CTMfloat res = ctmGetFloat(mContext, aProperty);
00100             CheckError();
00101             return res;
00102         }
00103
00104         const CTMuint * GetIntegerArray(CTMenum aProperty)
00105         {
00106             const CTMuint * res = ctmGetIntegerArray(mContext, aProperty);
00107             CheckError();
00108             return res;
00109         }
00110
00111         const CTMfloat * GetFloatArray(CTMenum aProperty)
00112         {
00113             const CTMfloat * res = ctmGetFloatArray(mContext, aProperty);
00114             CheckError();
00115             return res;
00116         }
00117
00118         CTMenum GetNamedUVMap(const char * aName)
00119         {
00120             CTMenum res = ctmGetNamedUVMap(mContext, aName);
00121             CheckError();
00122             return res;
00123         }
00124
00125         const char * GetUVMapString(CTMenum aUVMap, CTMenum aProperty)
00126         {
00127             const char * res = ctmGetUVMapString(mContext, aUVMap, aProperty);
00128             CheckError();
00129             return res;
00130         }
00131
00132         CTMfloat GetUVMapFloat(CTMenum aUVMap, CTMenum aProperty)
```

```

00164 {
00165     CTMfloat res = ctmGetUVMapFloat(mContext, aUVMap, aProperty);
00166     CheckError();
00167     return res;
00168 }
00169
00170 CTMenum GetNamedAttribMap(const char * aName)
00171 {
00172     CTMenum res = ctmGetNamedAttribMap(mContext, aName);
00173     CheckError();
00174     return res;
00175 }
00176
00177 const char * GetAttribMapString(CTMenum aAttribMap, CTMenum aProperty)
00178 {
00179     const char * res = ctmGetAttribMapString(mContext, aAttribMap, aProperty);
00180     CheckError();
00181     return res;
00182 }
00183
00184 CTMfloat GetAttribMapFloat(CTMenum aAttribMap, CTMenum aProperty)
00185 {
00186     CTMfloat res = ctmGetAttribMapFloat(mContext, aAttribMap, aProperty);
00187     CheckError();
00188     return res;
00189 }
00190
00191 const char * GetString(CTMenum aProperty)
00192 {
00193     const char * res = ctmGetString(mContext, aProperty);
00194     CheckError();
00195     return res;
00196 }
00197
00198 void Load(const char * aFileName)
00199 {
00200     ctmLoad(mContext, aFileName);
00201     CheckError();
00202 }
00203
00204 void LoadCustom(CTMreadfn aReadFn, void * aUserData)
00205 {
00206     ctmLoadCustom(mContext, aReadFn, aUserData);
00207     CheckError();
00208 }
00209
00210 // You can not copy nor assign from one CTMimporter object to another, since
00211 // the object contains hidden state. By declaring these dummy prototypes
00212 // without an implementation, you will at least get linker errors if you try
00213 // to copy or assign a CTMimporter object.
00214 CTMimporter(const CTMimporter& v);
00215 CTMimporter& operator=(const CTMimporter& v);
00216 };
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243 class CTMexporter {
00244     private:
00245         CTMcontext mContext;
00246
00247     void CheckError()
00248     {
00249         CTMenum err = ctmGetError(mContext);
00250         if(err != CTM_NONE)
00251             throw ctm_error(err);
00252     }
00253
00254
00255
00256
00257     public:
00258         CTMexporter()
00259         {
00260             mContext = ctmNewContext(CTM_EXPORT);
00261         }
00262
00263         ~CTMexporter()
00264         {
00265             ctmFreeContext(mContext);
00266         }
00267
00268         void CompressionMethod(CTMenum aMethod)
00269         {
00270             ctmCompressionMethod(mContext, aMethod);
00271             CheckError();
00272         }
00273
00274         void CompressionLevel(CTMuint aLevel)
00275         {
00276             ctmCompressionLevel(mContext, aLevel);
00277         }
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000
01001
01002
01003
01004
01005
01006
01007
01008
01009
01010
01011
01012
01013
01014
01015
01016
01017
01018
01019
01020
01021
01022
01023
01024
01025
01026
01027
01028
01029
01030
01031
01032
01033
01034
01035
01036
01037
01038
01039
01040
01041
01042
01043
01044
01045
01046
01047
01048
01049
01050
01051
01052
01053
01054
01055
01056
01057
01058
01059
01060
01061
01062
01063
01064
01065
01066
01067
01068
01069
01070
01071
01072
01073
01074
01075
01076
01077
01078
01079
01080
01081
01082
01083
01084
01085
01086
01087
01088
01089
01090
01091
01092
01093
01094
01095
01096
01097
01098
01099
01100
01101
01102
01103
01104
01105
01106
01107
01108
01109
01110
01111
01112
01113
01114
01115
01116
01117
01118
01119
01120
01121
01122
01123
01124
01125
01126
01127
01128
01129
01130
01131
01132
01133
01134
01135
01136
01137
01138
01139
01140
01141
01142
01143
01144
01145
01146
01147
01148
01149
01150
01151
01152
01153
01154
01155
01156
01157
01158
01159
01160
01161
01162
01163
01164
01165
01166
01167
01168
01169
01170
01171
01172
01173
01174
01175
01176
01177
01178
01179
01180
01181
01182
01183
01184
01185
01186
01187
01188
01189
01190
01191
01192
01193
01194
01195
01196
01197
01198
01199
01200
01201
01202
01203
01204
01205
01206
01207
01208
01209
01210
01211
01212
01213
01214
01215
01216
01217
01218
01219
01220
01221
01222
01223
01224
01225
01226
01227
01228
01229
01230
01231
01232
01233
01234
01235
01236
01237
01238
01239
01240
01241
01242
01243
01244
01245
01246
01247
01248
01249
01250
01251
01252
01253
01254
01255
01256
01257
01258
01259
01260
01261
01262
01263
01264
01265
01266
01267
01268
01269
01270
01271
01272
01273
01274
01275
01276
01277
01278
01279
01280
01281
01282
01283
01284
01285
01286
01287
01288
01289
01290
01291
01292
01293
01294
01295
01296
01297
01298
01299
01300
01301
01302
01303
01304
01305
01306
01307
01308
01309
01310
01311
01312
01313
01314
01315
01316
01317
01318
01319
01320
01321
01322
01323
01324
01325
01326
01327
01328
01329
01330
01331
01332
01333
01334
01335
01336
01337
01338
01339
01340
01341
01342
01343
01344
01345
01346
01347
01348
01349
01350
01351
01352
01353
01354
01355
01356
01357
01358
01359
01360
01361
01362
01363
01364
01365
01366
01367
01368
01369
01370
01371
01372
01373
01374
01375
01376
01377
01378
01379
01380
01381
01382
01383
01384
01385
01386
01387
01388
01389
01390
01391
01392
01393
01394
01395
01396
01397
01398
01399
01400
01401
01402
01403
01404
01405
01406
01407
01408
01409
01410
01411
01412
01413
01414
01415
01416
01417
01418
01419
01420
01421
01422
01423
01424
01425
01426
01427
01428
01429
01430
01431
01432
01433
01434
01435
01436
01437
01438
01439
01440
01441
01442
01443
01444
01445
01446
01447
01448
01449
01450
01451
01452
01453
01454
01455
01456
01457
01458
01459
01460
01461
01462
01463
01464
01465
01466
01467
01468
01469
01470
01471
01472
01473
01474
01475
01476
01477
01478
01479
01480
01481
01482
01483
01484
01485
01486
01487
01488
01489
01490
01491
01492
01493
01494
01495
01496
01497
01498
01499
01500
01501
01502
01503
01504
01505
01506
01507
01508
01509
01510
01511
01512
01513
01514
01515
01516
01517
01518
01519
01520
01521
01522
01523
01524
01525
01526
01527
01528
01529
01530
01531
01532
01533
01534
01535
01536
01537
01538
01539
01540
01541
01542
01543
01544
01545
01546
01547
01548
01549
01550
01551
01552
01553
01554
01555
01556
01557
01558
01559
01560
01561
01562
01563
01564
01565
01566
01567
01568
01569
01570
01571
01572
01573
01574
01575
01576
01577
01578
01579
01580
01581
01582
01583
01584
01585
01586
01587
01588
01589
01590
01591
01592
01593
01594
01595
01596
01597
01598
01599
01600
01601
01602
01603
01604
01605
01606
01607
01608
01609
01610
01611
01612
01613
01614
01615
01616
01617
01618
01619
01620
01621
01622
01623
01624
01625
01626
01627
01628
01629
01630
01631
01632
01633
01634
01635
01636
01637
01638
01639
01640
01641
01642
01643
01644
01645
01646
01647
01648
01649
01650
01651
01652
01653
01654
01655
01656
01657
01658
01659
01660
01661
01662
01663
01664
01665
01666
01667
01668
01669
01670
01671
01672
01673
01674
01675
01676
01677
01678
01679
01680
01681
01682
01683
01684
01685
01686
01687
01688
01689
01690
01691
01692
01693
01694
01695
01696
01697
01698
01699
01700
01701
01702
01703
01704
01705
01706
01707
01708
01709
01710
01711
01712
01713
01714
01715
01716
01717
01718
01719
01720
01721
01722
01723
01724
01725
01726
01727
01728
01729
01730
01731
01732
01733
01734
01735
01736
01737
01738
01739
01740
01741
01742
01743
01744
01745
01746
01747
01748
01749
01750
01751
01752
01753
01754
01755
01756
01757
01758
01759
01760
01761
01762
01763
01764
01765
01766
01767
01768
01769
01770
01771
01772
01773
01774
01775
01776
01777
01778
01779
01780
01781
01782
01783
01784
01785
01786
01787
01788
01789
01790
01791
01792
01793
01794
01795
01796
01797
01798
01799
01800
01801
01802
01803
01804
01805
01806
01807
01808
01809
01810
01811
01812
01813
01814
01815
01816
01817
01818
01819
01820
01821
01822
01823
01824
01825
01826
01827
01828
01829
01830
01831
01832
01833
01834
01835
01836
01837
01838
01839
01840
01841
01842
01843
01844
01845
01846
01847
01848
01849
01850
01851
01852
01853
01854
01855
01856
01857
01858
01859
01860
01861
01862
01863
01864
01865
01866
01867
01868
01869
01870
01871
01872
01873
01874
01875
01876
01877
01878
01879
01880
01881
01882
01883
01884
01885
01886
01887
01888
01889
01890
01891
01892
01893
01894
01895
01896
01897
01898
01899
01900
01901
01902
01903
01904
01905
01906
01907
01908
01909
01910
01911
01912
01913
01914
01915
01916
01917
01918
01919
01920
01921
01922
01923
01924
01925
01926
01927
01928
01929
01930
01931
01932
01933
01934
01935
01936
01937
01938
01939
0
```

```
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, 25

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, 25

CTM_UV_MAP_2

openctm.h, 25
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, 25
ctmAddAttribMap
 openctm.h, 25
ctmAddUVMap
 openctm.h, 26
ctmAttribPrecision
 openctm.h, 26
ctmCompressionLevel
 openctm.h, 27
ctmCompressionMethod
 openctm.h, 27
CTMcontext
 openctm.h, 22
ctmDefineMesh
 openctm.h, 27
CTMenu
 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, 29
ctmGetAttribMapFloat
 openctm.h, 29
ctmGetAttribMapString
 openctm.h, 30
ctmGetError
 openctm.h, 30
ctmGetFloat
 openctm.h, 31
ctmGetFloatArray
 openctm.h, 31
ctmGetInteger
 openctm.h, 32
ctmGetIntegerArray
 openctm.h, 32
ctmGetNamedAttribMap
 openctm.h, 33
ctmGetNamedUVMap
 openctm.h, 33
ctmGetString
 openctm.h, 33
ctmGetUVMapFloat
 openctm.h, 34
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, 35
ctmLoadCustom
 openctm.h, 35
ctmNewContext
 openctm.h, 36
ctmNormalPrecision
 openctm.h, 36
CTMreadfn
 openctm.h, 23
ctmSave

openctm.h, 37
ctmSaveCustom
 openctm.h, 37
CTMuint
 openctm.h, 23
ctmUVCoordPrecision
 openctm.h, 37
ctmVertexPrecision
 openctm.h, 38
ctmVertexPrecisionRel
 openctm.h, 38
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, 39
 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, 25
CTM_OUT_OF_MEMORY, 24
CTM_PRECISION, 24
CTM_TRIANGLE_COUNT, 24
CTM_TRUE, 22
CTM_UNSUPPORTED_FORMAT_VERSION, 24
CTM_UV_MAP_1, 25
CTM_UV_MAP_2, 25
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, 25
ctmAddAttribMap, 25
ctmAddUVMap, 26
ctmAttribPrecision, 26
ctmCompressionLevel, 27
ctmCompressionMethod, 27
CTMcontext, 22
ctmDefineMesh, 27
CTMenum, 24
ctmErrorString, 28
ctmFileComment, 28
CTMfloat, 22
ctmFreeContext, 29

ctmGetAttribMapFloat, 29
ctmGetAttribMapString, 30
ctmGetError, 30
ctmGetFloat, 31
ctmGetFloatArray, 31
ctmGetInteger, 32
ctmGetIntegerArray, 32
ctmGetNamedAttribMap, 33
ctmGetNamedUVMap, 33
ctmGetString, 33
ctmGetUVMapFloat, 34
ctmGetUVMapString, 34
CTMint, 23
ctmLoad, 35
ctmLoadCustom, 35
ctmNewContext, 36
ctmNormalPrecision, 36
CTMreadfn, 23
ctmSave, 37
ctmSaveCustom, 37
CTMuint, 23
ctmUVCoordPrecision, 37
ctmVertexPrecision, 38
ctmVertexPrecisionRel, 38
CTMwritefn, 23
openctmpp.h, 42
operator=
 CTMexporter, 13
 CTMimporter, 17

Save
 CTMexporter, 13
SaveCustom
 CTMexporter, 13

UVCoordPrecision
 CTMexporter, 13

VertexPrecision
 CTMexporter, 13
VertexPrecisionRel
 CTMexporter, 13

what
 ctm_error, 10