
# OgreCollada v0.1a Manual

Van Aarde Krynauw

August 21, 2008

# Contents

# Chapter 1

# OgreCollada timeline

## 1.1 A brief history

OgreCollada was originally developed by Gregory Junker in 2007, utilizing the FCollada library[1] which supported loading of static scenery, materials, shaders, lights, cameras and possibly some more things. In 2008 Gregory added support for loading skeletons, mesh skinning and rudimentary scene node animation support. OgreCollada was proposed as a Google Summer of Code (GSoC) project in 2008 by Van Aarde Krynauw with Ogre3D being the mentoring organisation. The primary goal of the GSoC term would be to implement stable support for skeletal animation. During GSoC development scene node animation was rewritten, mesh skinning had to be slightly altered and an initial implementation of skeletal animation was done. This manual was written as part of the GSoC 2008 project.

## 1.2 What's new

### 1.2.1 v0.1a

Some features that are included in OgreCollada v0.1a:

- Light transformation animation
- Camera transformation animation
- Scene node animation
- Skeletal animation
- ColladaViewer that plays scene node and skeletal animations and open/imports[2] COLLADA documents.
- OgreCollada can be compiled for 32- and 64-bit Microsoft Windows (Visual C++) and Linux (GNU GCC) operating systems.

## 1.3 Roadmap

Features that are planned for future releases[3].

### 1.3.1 v0.2

- ColladaViewer redesigned and rewritten using GTK+.
- Command line converter utility.

[1] The FCollada library was developed by Feeling Software (http://www.feelingsoftware.com) but is now an open source project that resides at SourceForge as ColladaMaya (http://www.sf.net/projects/colladamaya).

[2] Refer to section 2.3.1

[3] DISCLAIMER: Roadmap features may be changed or removed without prior notification thereof.

### 1.3.2 v0.x

- Improve skeletal animation.
- Physics support.
- Spline paths.
- Allow user to easily access external reference URLs.
- Vertex animation (Pose and morph).
- Extend OgreCollada's importflags to support animation data.
- Convert COLLADA data to Ogre resources and write them back into a COLLADA document using <extra> tags as external references so that the COLLADA document can be used as a scene graph format.
- Name collection mechanism to allow inputting of a text file that will match resource names and rename them to something else, as specified in the name collection.

# Chapter 2

# OgreCollada utilities

Here you will find a description of the various parts / utilities that is included in the OgreCollada library.

## 2.1 FCollada

FCollada is the library used by OgreCollada to load the COLLADA 1.4.1 documents into intermediate data structures.

## 2.2 OgreCollada

OgreCollada retrieves the COLLADA document data from FCollada and loads it into an Ogre scene.

### 2.2.1 Using OgreCollada

Refer to chapter 4.

## 2.3 ColladaViewer

The ColladaViewer loads the COLLADA document into an Ogre instance and allows the user to pan / rotate the default camera, play/stop animations.

### 2.3.1 Using the viewer

| Input | Action |
|---|---|
| Left mouse button | Pan camera |
| Right mouse button | Rotate camera |
| Scroll middle mouse button | Zoom camera |

To play a scene node animation select the animation from the *Node Animation* menu. To stop the animation, deselect the animation in the menu. To play a skeletal animation, open the *Entity* menu and select the animation for the appropriate entity.

Note that the ColladaViewer makes use of some Ogre3D core media files located at **OgreCollada/media**. A linux binary of the MeshMagick [1] utility still resides in the directory which was used to scale the axes.mesh file.

#### Opening and importing

The ColladaViewer provides two different methods of loading COLLADA documents. The first, *Open*, loads a COLLADA document into a clean Ogre scene. The second, *Import*, loads COLLADA documents into the existing Ogre instance without clearing it beforehand.

[1] MeshMagick can be found in the ogre addons directory.

### Export

The ColladaViewer has a basic exporting functionality that saves the *all* Ogre meshes, materials and skeletons (that are currently in the scene) to the specified output directory. Note that the Ogre's material serializer does not copy textures when exporting a material, so this has to be done manually.

Note that LexiView will not be able to load the meshes directly after export because the material scripts, exported by OgreCollada, contains the default Ogre materials as well which clashes with the materials in LexiView's Ogre instance. After removing the offending materials from the exported material script the exported meshes should be loadable.

# Chapter 3

# Compiling OgreCollada

Note beforehand that OgreCollada can, at the time of writing, only be compiled as a static library. Shared library support will be added at a later stage.

## 3.1 Prerequisites

### 3.1.1 Build System

OgreCollada build scripts can be generated for multiple build environments by using the *premake* utility which can be downloaded at http://premake.sourceforge.net. Note that a *premake* binary executable for Windows has been included in the *premake* directory.

### 3.1.2 OgreCollada

OgreCollada is directly dependent on the following libraries:

- Ogre3D Engine v1.4.9 (http://www.ogre3d.org), and
- FCollada[1] (http://colladamaya.sf.net).

### 3.1.3 ColladaViewer

The ColladaViewer is dependent on the following libraries:

- OgreCollada (http://ogrecollada.sf.net),
- wxWidgets[2] v2.8.8 (http://www.wxwidgets.org), and
- OpenGL (http://www.opengl.org).

## 3.2 Obtaining the source code

OgreCollada is hosted at SourceForge (http://www.sf.net) using an SVN repository. The source code can be checked out with the following command:

```
svn co https://ogrecollada.svn.sourceforge.net/svnroot/ogrecollada ogrecollada
```

Provided that you have downloaded and extracted the OgreCollada source code you can now continue to the next sections to find operating system specific information on compiling OgreCollada.

[1] OgreCollada will not compile against the stock FCollada v3.05B. It requires the modified version of 3.05B which is distributed with OgreCollada.

[2] The ColladaViewer GUI implementation will change from wxWidgets to GTK in the near future.

## 3.3 Building on Linux

### 3.3.1 GCC compiler

#### Generate build scripts

To compile OgreCollada on a linux system you will be using Makefiles. You can use premake to generate Makefiles for a GNU system by executing the following *premake* command, from within the **premake** subdirectory:

**OgreCollada/premake $** premake --verbose --target gnu

The *--verbose* command line option generates verbose Makefiles. OgreCollada links, by default, against shared wxWidgets library. To generate Makefiles that link against static wxWidgets libraries you need to execute premake with the *--with-static-wx* command line option:

**OgreCollada/premake $** premake --with-static-wx --target gnu

#### Notes on Ogre3D, wxWidgets and OpenGL configuration

The CFLAGS for both Ogre3D and wxWidgets are determined dynamically (upon running *make*). Ogre3D's CFLAGS are obtained by using the *pkg-config* utility. If you need to specify these values manually, edit the **OgreCollada/premake/OgreCollada/premake.lua** file. Simply follow the instructions in the comments where the *pkg-config* command is invoked. Similarly, wxWidgets and OpenGL flags can be manually set in the **OgreCollada/premake/ColladaViewer/premake.lua** file. Remember to rebuild the Makefiles when you have altered the premake scripts!

#### Compiling

To build OgreCollada in the default *Release* configuration, you can simply run *make*. To compile the *Debug* configuration, you must run the *make* command as follows:

**OgreCollada/premake $** make CONFIG=Debug

Note that you can also specify explicitly to build as *Release* configuration.

#### Output

Output of executables and libraries will be in **OgreCollada/bin/[configuration]**. If you compiled the ColladaViewer, the executable is located at **OgreCollada/bin/[configuration]/ColladaViewer**.

## 3.4 Building on Microsoft Windows

### 3.4.1 Microsoft Visual Studio

#### Generate build scripts

Batch files have been included to easily generate solution files for Visual Studio. Two batch files are provided for each supported Visual Studio version and are located in the **OgreCollada/premake** directory. To generate solution files that link against a shared (DLL) wxWidgets library, run the appropriate batch file for your VS version, e.g., for Visual Studio 2008 run the file **vs2008.bat**. If you want to link against static wxWidgets libraries, run the batch file pertaining to your Visual Studio version that ends with *_static_wx*, e.g., **vs2008_static_wx.bat**. The solution file will be created in the **OgreCollada/premake** directory.

Windows users will have to manually set up the *include* and *library* paths to both Ogre and wxWidgets in the Visual Studio configuration. It is recommended to define these paths in your *global* Visual Studio configuration otherwise the settings will be lost next time the solution scripts are generated.

### Define custom include and library paths

This will most likely be necessary when compiling the ColladaViewer on the Microsoft Windows operating system. Open the **OgreCollada/premake/ColladaViewer/premake.lua** script and search for the string "--DEFINE YOUR CUSTOM INCLUDES AND LIBDIRS HERE" to see an example on how to accomplish this.

### Compiling

Click the compile button and hope for the best!

# Chapter 4

# Working with OgreCollada

Using OgreCollada is a rather straight-forward business. Here we will be looking at an overview of using the (simple) OgreCollada API. For a more complete reference, especially with regards to each function's parameters, refer to the OgreCollada API reference. The *ColladaViewer* application also serves as an implementation example for OgreCollada.

## 4.1 Basic Usage

### 4.1.1 Declaration

Access to the OgreCollada importer can be gained through a pointer to the **OgreCollada::ImpExp** class (defined in **OgreCollada/OgreCollada/include/OgreCollada.h**):

```
OgreCollada:ImpExp* pImporterExporter;
```

### 4.1.2 Creation

Creating the exporter is done through a global function in the OgreCollada namespace:

```
pImporterExporter = OgreCollada::CreateImpExp(pRootNode, pSceneMgr);
```

### 4.1.3 Resource notifications

If you are interested to be notified when new Ogre resources are created you can implement the *IResourceNotification* interface and register the object with OgreCollada.

```
//Implement the IResourceNotification interface
class ColladaViewer :  public OgreCollada::IResourceNotification, SomeOtherClasses
{
  void cameraCreated(Ogre::Camera* pCam, FCDCamera* pFColladaCamera);
  void sceneNodeCreated(Ogre::SceneNode* pNode, FCDSceneNode* pFColladaSceneNode);
  ⋮
};

//Register the IResourceNotification instance with the OgreCollada importer
pImporterExporter->setResourceNotificationListener( pColladaViewer );
```

This is typically useful for something like a command line converter where the command line converter can serialise the Ogre resources whenever a resource has been created.

### 4.1.4 Importing a COLLADA document

The COLLADA document is imported into a scene graph that was given to the OgreCollada importer upon it's creation (see section 4.1.2):

```
bool success = pImporterExporter->importCollada( filename, prefix, importflags );
```

The prefix string is appended to the name of every resource created in Ogre during this import. The reason for this is when multiple COLLADA documents are imported into Ogre it is very likely that there will be duplicate names of resources and Ogre requires most resources of the same type to have unique names. It is recommended to prefix each import with a unique string identifier, e.g., "import1", "import2", "import3", etc.

Something else to keep in mind is that you do not need to import your data into your active Ogre scene graph. You can keep a separate scene graph for importing documents, processing the results and then using the data that you want/need in your active scene graph.

Import flags can be specified to filter which Ogre resources must be loaded from the COLLADA document. The default behaviour is to load everything.

### 4.1.5 Destruction

Destroying the exporter is done through a global function in the OgreCollada namespace:

```
OgreCollada::DestroyImpExp(pImporterExporter);
```

## 4.2 Possible pitfalls

Even though OgreCollada imports COLLADA data into an Ogre instance some of the data may still need special care in order to work correctly. This section will discuss some of those potential pitfalls.

### 4.2.1 Sharing skeleton between multiple entities

Some COLLADA documents will contain skinned models, with skeletal animation, that consist of multiple meshes which may be imported into Ogre as separate entities (This depends on exactly how the model was constructed in the 3D animation package). One skeleton instance must be shared amongst all the entities so that the entities are all animated by activating an animation state from the single skeleton instance. See the ColladaViewer implementation for an example.

# Chapter 5

# Features

This chapter will list features (and their status in OgreCollada) that are required to be implemented by COLLADA exporters and importers from the COLLADA 1.4.1 specification. For a more complete description of all the features listed below, refer to the COLLADA 1.4.1 specification. Note that all the feature statuses listed below are only for *importing* COLLADA documents. Export feature lists will be added when OgreCollada's exporting features have been developed.

## 5.1 Feature status legend

| Symbol | Description |
|---|---|
| o | Not yet started |
| x | In progress |
| t | Testing phase (Stable feature with known bugs) |
| v | Finished (All known bugs have been eliminated) |
| i | Ignoring this feature (not applicable for OgreCollada) |
| ? | Status of this feature is unknown [1] |

## 5.2 Hierarchy and transforms

| Feature | Import Status |
|---|---|
| Translation | v |
| Scaling | v |
| Rotation | v |
| Parenting | v |
| Static object instantiation | ? |
| Animated object instantiation | ? |
| Skewing | o |
| Transparency / reflectivity | ? |
| Texture mapping method | ? |
| Transform with no geometry [2] | ? |

[1] This is most likely because the person that wrote the documentation does not know how functional the feature is or whether it even exists.

[2] It must be possible to transform something with no geometry, e.g., a locator / empty / NULL object.

## 5.3 Materials and textures

| Feature | Import Status |
|---|---|
| RGB Textures | ? |
| RGBA Textures | ? |
| Baked Procedural Texture Coordinates | ? |
| Common profile material | ? |
| Per-face material | ? |

## 5.4 Vertex attributes

| Feature | Import Status |
|---|---|
| Vertex texture coordinates | ? |
| Vertex normals | t |
| Vertex binormals | ? |
| Vertex tangents | ? |
| Vertex UV coordinates | t |
| Custom vertex attributes | ? |

## 5.5 Animation

This section does not reflect the animation specification in the COLLADA specification as the specification is missing a few things.

| Feature | Import Status |
|---|---|
| Skeletal structure bind pose | t |
| Skeletal animation | t |
| Skeletal animation with skinned mesh | t |
| Scene node transformation animation | t |
| Camera transformation animation | t |
| Light transformation animation | t |
| UV placement parameters | o |
| Light parameters | o |
| Camera parameters | o |
| Shader parameters | o |
| Global environment parameters | o |
| Mesh-construction parameters | o |
| Node parameters | o |
| User parameters | o |

## 5.6 Scene Data

| Feature | Import Status |
|---|---|
| Empty nodes | ? |
| Cameras | t |
| Spotlights | t |
| Directional lights | t |
| Point lights | t |
| Ambient lights | ? |

## 5.7 Geometric Primitives

| Feature | Import Status |
|---|---|
| Triangles | v |
| Triangle strips | o |
| Triangle fans | o |
| Polygons | o |
| Polylists | o |
| Points | o |
| Lines | o |
| Linestrips | o |

Polygons and polylists are not directly support by Ogre by OgreCollada may, in the future, triangulate this data such that it can be displayed by Ogre. Note that the COLLADA-Refinery[3] can be used for mesh triangulation.

[3] http://sourceforge.net/projects/colladarefinery/

# Chapter 6

# Bugs and Limitations

## 6.1 Known Bugs

- The texturing for a skinned mesh breaks.
- Lights aren't animated properly when parented to animated objects.
- Skeletal animation issues seems to occur when a skeleton is shared amongst multiple entities.
- There are issues when importing a COLLADA document that contains a skinned mesh as well as a mesh that is not bound to the skeleton. This bug is most likely a ColladaViewer bug.
- Most crashes or strange document loading behaviour experienced in the ColladaViewer is inherent to the viewer itself and not necessarily OgreCollada.
- Visual Studio reports memory leaks that will be addressed at a later stage.

## 6.2 OgreCollada Limitations

The following limitations are specific to OgreCollada and will most likely be improved upon in future releases.

- Exporting a COLLADA document requires manual manipulation of the FCollada document structure.
- OgreCollada currently only **correctly** imports triangulated meshes. Importing of any other type of mesh will lead to unexpected results.
- It often happens that the keyframes for node (can be either a scene node or bone) that are retrieved from COLLADA does not have translation and rotation data at the same intervals (in COLLADA the translation, rotation and scaling keyframes are completely separated and having one of them does not imply that you'll get the other). In Ogre, on the other hand, each keyframe requires a translation, rotation and scale. To bridge these different approaches a keyframe sampler had to be devised that can interpolate between keyframes. Currently the sampler only interpolates between translation keyframes. Rotation interpolation still needs to be implemented.

## 6.3 FCollada Limitations

The following limitations are inherent to the FCollada library and will probably not be addressed by OgreCollada developers.

- Sketchup COLLADA exports are not properly supported, although the files are COLLADA compliant. FCollada breaks some of the transforms.