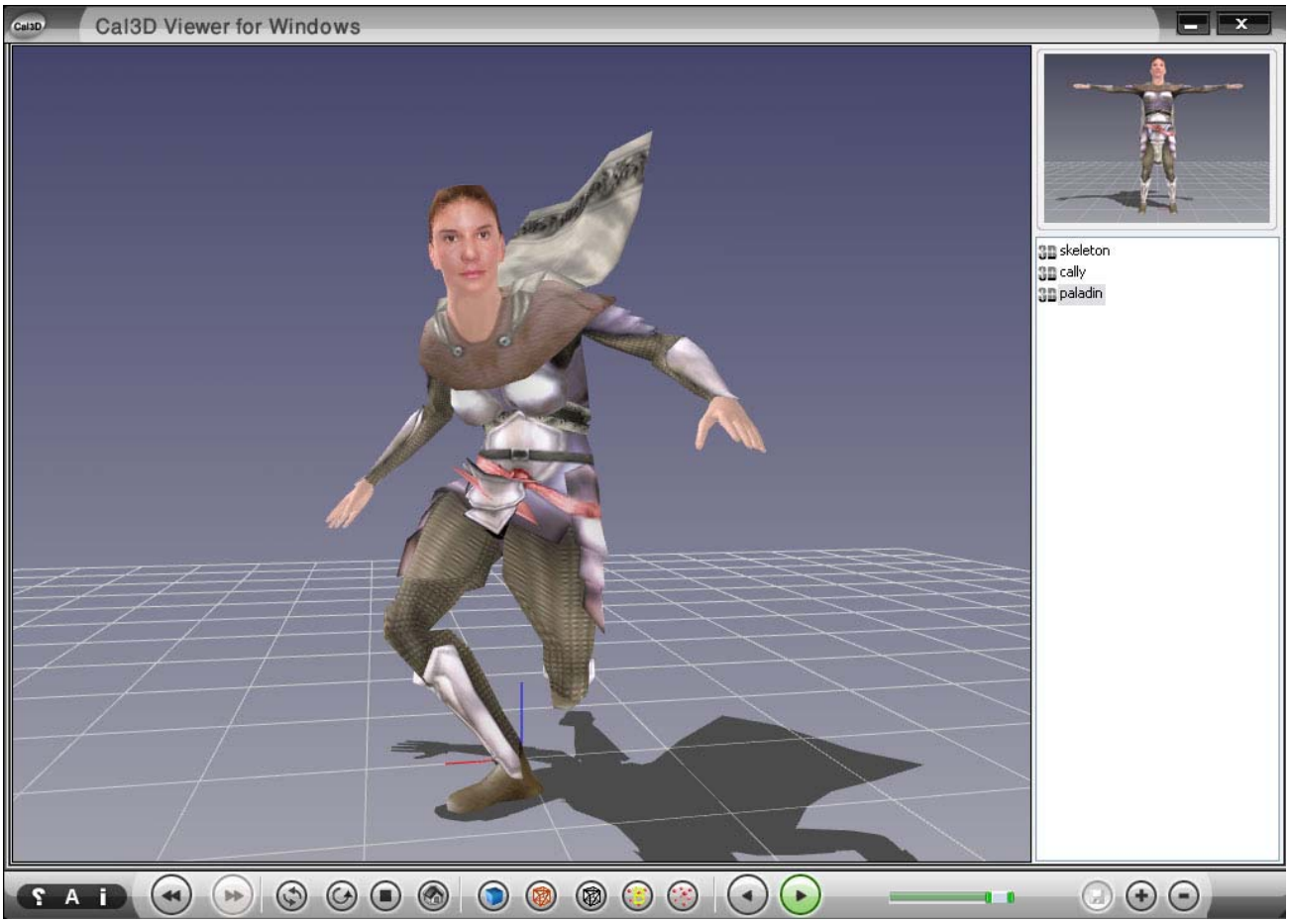
Cal3DViewer for Windows

version 1.0.1

The user's manual



Programming, testing and documentation
Ehsan Kamrani
opentechno@hotmail.com

8/22/2008

1. Introduction

This section is a brief introduction about the Cal3D and important aspects of this program.

1.1. Cal3D Overview

Cal3D is a skeletal based 3D character animation library written in C++ in a way that is both platform-independent and graphics API-independent. It was originally designed to be used in a 3D client for Worldforge, but evolved into a stand-alone product which can be used in many different kinds of projects.

Cal3D's essentials can be boiled down to 2 parts: the C++ library and the exporter. The exporter is what you would use to take your characters (built in a 3D modeling package) and create the Cal3D-format files that the library knows how to load. The exporters are actually plug-ins for 3D modeling packages. This allows 3D artists to use the modeling tools that they're already comfortable with.

The C++ library is what you would actually use in your application, whether it's a game or a VR application. The library provides methods to load your exported files, build characters, run animations, and access the data necessary to render them with 3D graphics.

Cal3D models use ".csf"(skeleton file), .cmf (mesh file), .crf (material file) and .caf (animation file). Each model may have more than one mesh, material and animation, but only one skeleton. We usually write the name of these files in a text file to access to those names via this file.

Here are some more references about Cal3D:

https://gna.org/projects/cal3d/

http://home.gna.org/cal3d/

http://download.gna.org/cal3d/documentation/guide/

http://download.gna.org/cal3d/documentation/modeling/tutorial.html

1.2. Cal3DViewer Overview

Cal3DViewer is free software, focused into rendering Cal3D files in real time. This program is open source, easy-to-use and well-documented. Due to that, this project will be very useful for people who are interested to use Cal3D in their projects and/or test their exported models.

To develop this software, I have created an MFC application with the C++ language and OpenGL, Cal3D and Devil APIs.

- Programmers should notice that I haven't used Devil to load the Targa textures. Some textures of the Cal3D model aren't loaded correctly with Devil (It's not well known). That is why I decided to load the Targa files of the models with the tga class that is included in the "miniviewr_gl" example. However I haven't removed the function that can load the .tga files with Devil—look at the commonGL.h and CommonGL.cpp of the source code.

C++, OpenGL and Cal3D are cross-platform, but MFC projects run only on Windows OS. That is why this software isn't cross-platform and I have called it "Cal3DViewer for Windows", but the short name of this project is Cal3DViewer. Next time, I'll use wxWidgets or another API instead of MFC to make this software cross-platform.
In this project I have used "Cal3D 0.11" to solve the problems for a Cal3D viewer.

1.3. What Benefits Does It Have Over Other Cal3D viewers?

1.3.1. Better User Interface

CallyDemo and other applications released with Cal3D source code have educational purposes. Their intent is to make the code as simple as possible, so those programs aren't user friendly, they don't use dialog boxes to load the models, they have no database to save/load the names of the models on a list, they have no option to remove a model from the list and they use command lines to load the models which are not appropriate in commercial software.

Cal3DViewer has solved all these problems with a better interface. It also uses bitmap buttons, bitmap slider and/or bitmap dialog boxes which make the interface more interesting. The interface is easy to use, the user can load a model via a dialog, He/she can remove a name from a list with a button and he/she can save the names of the models with just pushing a button. You can find more about its interface at section 8.

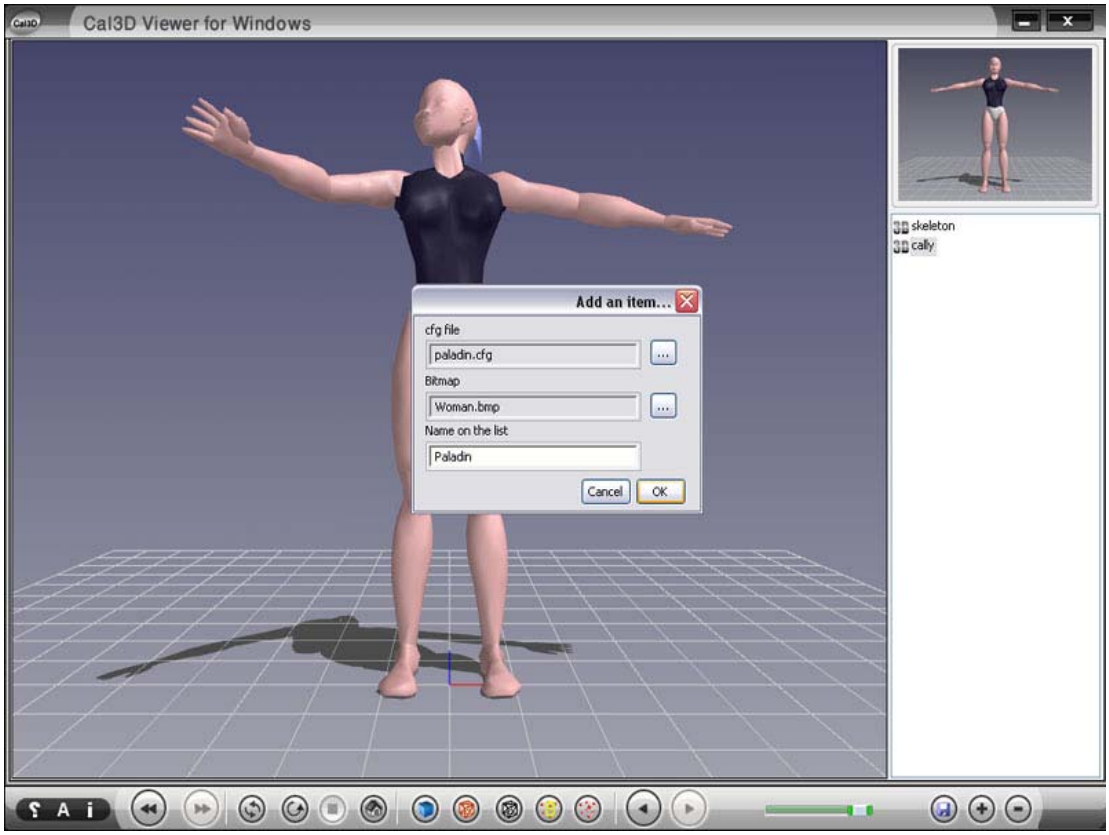


Figure 1. Cal3DViewer

1.3.2. More Information

The purpose of this viewer is not just loading the models. It's software to test the Cal3D models, so it should report some information about the models. For example, if it can't load some files of a model such as its textures, it should report an appropriate message with the name of the textures which this program failed to load.

This software also gives some useful information about your OpenGL implementation and your model such as you OpenGL vendor and number of faces, vertices and animations of the model, etc.

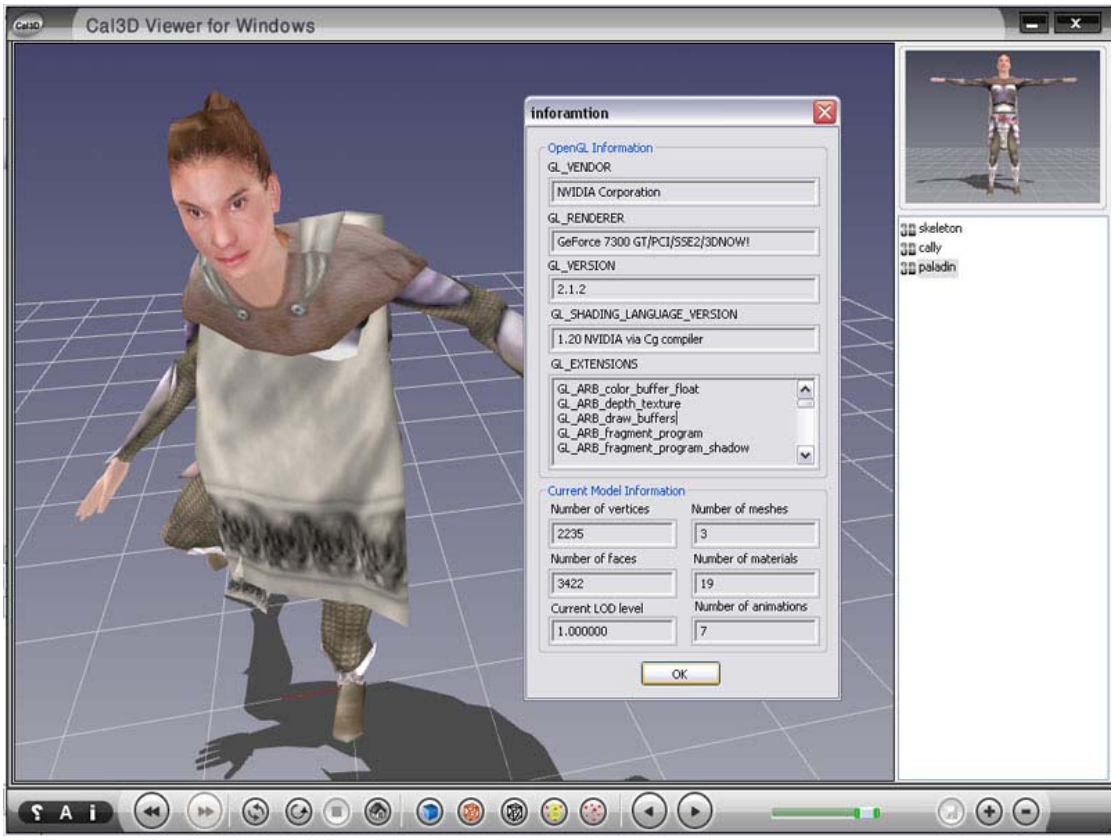


Figure 2. Information dialog

1.3.3. A Companion Document

Most of other cal3D viewers have not good documentations. A good document solves many problems for the programmers and end users and saves their time.

1.3.4. More Utility Functions and Classes

It took about one year that I developed a simple game engine to create simple games. I and my friends even created a small game for kids with this engine, but we never finished it as a commercial game, because of the lack of budget. To load and animate the characters of that game, I also used Cal3D.

I included some classes and functions of that engine—But not all of them. Although some of them are not used in this software (such as some functions in the file CommonGL.h), but they are really useful in OpenGL programs. Most of those files have documentations at the beginning of them, so they simplify the codes to understand. However some of those documentations are not up to date.

Figure 3. A simple game created with my game engine

2. Getting Started With Cal3DViewer and Installation

First of all, it is necessary to install the program, but it does not include any installer application. It is just a ZIP file, *Cal3DViewer.zip*. The file contains the following folders and files:

- Binary: a folder with all files and data necessary to run the program.
- User's Manual.pdf: you are reading this file right know.
- Readme.txt: some Information related to the current version.

Begin by extracting the included ZIP file in any folder, after that, go to that folder and enter in the *Binary* folder and double click “Cal3DViewer.exe”. The program will be executed. Note that it's the release version of the project. I have added 3 models to the database, so you will see them when you run the program (see the copyright section about the copyright of those models). You can remove the model names from the list and save the list or you can delete the file "database.dbf"- which is beside the executable file- to get an empty list.

3. Copyright

This program is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

I have copied some texts of the file" User Manual.pdf "of the ColladaLoader software in this user's manual. The author of that file and project is Ricardo Ruiz López and he has permitted me to copy some of the texts of his user's manual. You can find more about his project here:

http://sourceforge.net/project/showfiles.php?group_id=197616

Some texts of this user's manual are from the Cal3D FAQ. All the 3D models and their associated files are from the Cal3D project and you can download them from here:

http://download.gna.org/cal3d/sources/

2D bitmaps (except bitmap slider) of this project have been created by my friend, Shahriar Hosseni. 2D bitmaps are copyright © 2008, Shahriar Hosseini.

To create the bitmap slider of this project, I have used the CBitmapSlider project. You can download and read its comments here:

http://intel.tistory.com/2460558

I have used some functions and classes of the miniviewer_gl example from the Cal3D project, but I have changed and developed some of those functions and classes.

The author of Vector class is Vic Hollis. His codes are free as he has writen at the beginning of vector.h and vector.cpp files.

The entire source codes in which have been written by myself and this user's manual are copyright © 2008, Ehsan Kamrani.

4. System Requirements

These are the minimum system requirements necessary to run this program:
Windows 98 / Windows Me / Windows 2000 / Windows XP or Windows Vista (Windows NT and 95 are currently not supported).
800 MHz or faster Intel Pentium III or AMD Athlon processor.
256 MB and/or more RAM.
10 MB and/or more of free hard disk space
32 bits desktop display.
An OpenGL 1.1 compliant graphics card with at least 32 MB and these three extensions available:
GL_ARB_MULTISAMPLE,
GL_EXT_texture_filter_anisotropic
GL_SEPARATE_SPECULAR_COLOR

Your graphics card manual should have this information; in other case you can use an “Extensions Viewer” like for instance this one:
http://www.realtech-vr.com/glview/index.html

5. Support

This program is free software and it has no support of any class. However, if you have any questions you can send me an email to opentechno@hotmail.com

6. Features

- Meshes loading with vertex, normal and texture coordinate.
- Materials can be applied per face.
- Two kinds of transformations are supported: translation and rotation.
- Three directional lights are used.
- Information about the current model.
- Information about your OpenGL implementation.
- An orbital camera is used.
- Five different render modes.
- Easy-to-use user interface with bitmap buttons, dialogs, etc.
- Full-scene anti-aliasing.
- Mip-mapping, trilinear and anisotropic filtering**.**
- A shadow can be drawn for each loaded model.
- It is possible to activate or deactivate grid, shadow and axes drawing.
- It uses a data base to save the model names, addresses and their associated data, so next time you run this program it will load them if they exist.
- Support for Unicode strings

7. Using Cal3DViewer

It's a description that explains how to run the most important tasks of this program.

7.1. Viewing Models

To load and visualize a Cal3D model version 0.11, you should click the + button. It will open a dialog. There are three edit controls in this dialog. The first and second ones are read only controls and you need to use the buttons beside them to pop up a typical windows open file dialog and select the file name.

I have used relative addresses instead of absolute addresses. It's because this program saves the list and you may want to run your program in another system and don't use the same path as you used in your previous computer. You write the names of your Cal3D files in a text file with the .cfg extension. You need to put your .cfg file inside the folder data/models/your .cfg file name without the ".cfg" string/. For example if you have a model

with a .cfg file called cally.cfg, you have to put it inside the data/models/cally/, so when you click on the cally folder, you can see your .cfg file and all the files related to your cally model. You need to write this local address in your .cfg file with the “path” keyword. For example for our cally example, write the following line at the beginning of your .cfg file:
path = data/models/cally/
Some keywords must be used in your .cfg file to specify the file type and names correctly:

- *path* = specifies the relative path file in which you inserted your .cfg file.
- *scale* = It's used to adjust your model distance.Lower scales move the object to upper and nearer place.
- *skeleton* = Specifies the skeleton file.
- *mesh* = specifies the mesh file.
- *material* = specifies the material file.
- *animation* = specifies the animation file.
- # is used for comments and is not parsed by the program.

Here's our cally.cfg file:

```
#
# cal3d model configuration file
#model: callly
path = data/models/cally/
scale=1.0

skeleton=cally.csf

animation=cally_idle.caf
animation=cally_tornado_kick.caf
animation=cally_walk.caf
animation=cally_strut.caf
animation=cally_jog.caf
animation=cally_shoot_arrow.caf
animation=cally_wave.caf

mesh=cally_calf_left.cmf
mesh=cally_calf_right.cmf
mesh=cally_chest.cmf
mesh=cally_foot_left.cmf
mesh=cally_foot_right.cmf
mesh=cally_hand_left.cmf
mesh=cally_hand_right.cmf
mesh=cally_head.cmf
mesh=cally_lowerarm_left.cmf
mesh=cally_lowerarm_right.cmf
mesh=cally_neck.cmf
mesh=cally_pelvis.cmf
mesh=cally_ponytail.cmf
mesh=cally_thigh_left.cmf
mesh=cally_thigh_right.cmf
mesh=cally_upperarm_left.cmf
mesh=cally_upperarm_right.cmf

material=cally_skin.xrf
material=cally_ponytail.xrf
material=cally_chest.xrf
material=cally_pelvis.xrf
#end of cfg file
```

- Spaces and blank lines are ignored
- Keywords are case sensitive
- You must specify the file names correctly.If the program fails to load the file, it will report with an error. But if you don't specify the keyword correctly(For example if you write *Animation* instead of *animation*, the parser ignores that line and doesn't report an error.
- you must speciffy the *path* keyword before all other keywords
- you must specify your skeleton file before the animation, material and mesh file(s)
- The sequence in which you specify the mesh, animation and material files is not important.

You also need to put your bitmap files inside the data/bitmaps/ folder. After that you can select your bitmap and .cfg files and give the model a name and press the OK button. The program will try to load the model and if it fails to load some files, it will report it with appropriate messages.

- If you don't choose a bitmap file for a model, it will show the default bitmap (data/bitmaps/default.bmp) and if it doesn't find the default bitmap, it doesn't show the preview.
- Both cfg file and model name are required to load a model.

7.2. Camera

There is an orbital camera present. The orbital camera is controlled with the mouse and its buttons.

- To zoom in, move the mouse up while pressing the right button.
- To zoom out, move the mouse down while pressing the right button.
- To rotate the camera around the model, move the mouse in any direction while pressing the left button.
- To strafe the scene horizontally and vertically, move the mouse in the desired direction while pressing the middle button.

7.3. Lighting

Models are affected by three lights: Key light, fill light and back light. Key light generates a shadow. This type of lighting produces realistic results. These are directional lights and you can not change their directions (except you edit the source code and recompile it).

8. User Interface

This program uses bitmap dialog, bitmap buttons and bitmap slider. Adding the bitmaps to the application improves its appearance and makes it more interesting.
Here are useful buttons:

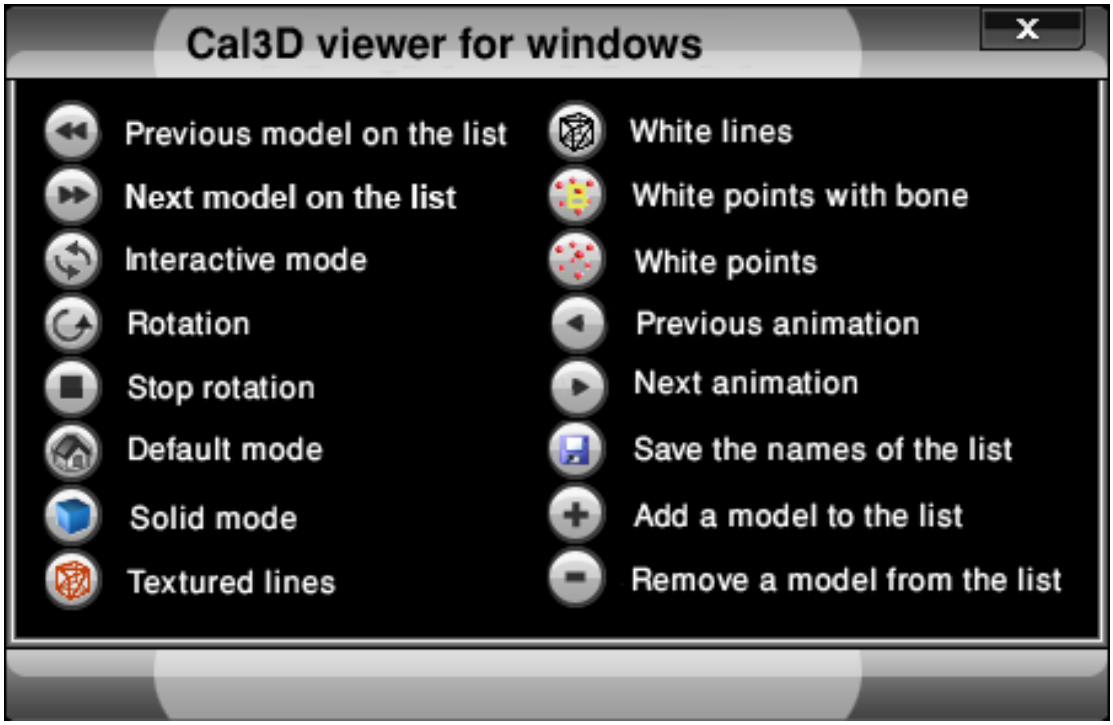


Figure 4. Help dialog

- One feature of this software have not an equivalent bitmap button, but it has a shortcut key. You can show or hide the grid, axes and shadow with the CTRL + G keyboard shortcut.
- Slider is used to change the LOD of the model. Note that changing the LOD crashes the program while selecting the paladin model. There's the same problem in miniviewer_gl example that has been released with cal3d source. It seems that this problem refers to cal3d, not my application. I have reported the bug to the cal3d moderators.

Grids and shadow are drawn in XY plane. I did it to show the default Cal3D models (cally, skeleton and paladin) correctly.

9. Keyboard Shortcuts

CTRL + C	Close the window
CTRL + M	Show or hide the window
CTRL + F1	Show the “About Cal3DViewer” dialog
CTRL + SHIFT + I	Show a dialog with information about the OpenGL implementation and current model
CTRL + H	Show the "Help" dialog
CTRL + A	Add a model to the list
CTRL + DELETE	Remove a model from the list
CTRL + S	Save the list name
CTRL + N	Show the next model on the list if it exists
CTRL + P	Show the previous model on the list if it exists
CTRL + SHIFT + N	Show the next animation of the current model if it exists
CTRL + SHIFT + P	Show the previous animation of the current model if it exists
CTRL + I	Activate interactive mode
CTRL + R	Start rotation
CTRL + SHIFT + S	Stop rotation
CTRL + F	Activate fill render mode
CTRL + T	Activate line render mode with textures or materials
CTRL + W	Activate white line render mode
CTRL + V	Activate white point(vertex) render mode
CTRL + B	Activate white point render mode and show the model's bone
CTRL + D	Activate default transformations and states(default mode)
CTRL + G	Show or hide the grid, axes and shadow

10. Acknowledgment

Special thanks to:

- Angus Dorbie
- Ricardo Ruiz López.
- behnam Safarzade

Thanks to Mr. Reza Babaei Pour for editing some texts of this document.

The information that programmers at the OpenGL and Gamedev forums gave me were really useful.