Stereoscopic Scene Viewer Project Documentation:

Project Members:

Jesse Fish

Kelvin Eweka

**Background:**

The purpose of this project is to provide a stereoscopic 3d scene-viewing environment that can be controlled via Matlab.

This project was written in Java. With the version of Matlab that we used we found it best to compile with Java 5.

We programmed the project using the Eclipse IDE. All of the code for the project can be found at the googlecode page https://stereosceneviewer.googlecode.com/

The codebase consists of two projects, both of which must be checked out using an svn client. We suggest using the Subversive plug in for Eclipse.

The two projects can be found at:

https://stereosceneviewer.googlecode.com/svn/3dSceneViewer/trunk/3dSceneViewer

and

https://stereosceneviewer.googlecode.com/svn/ObjectLoader/trunk/ObjectLoader

There is a self contained JAR file in the 3dSceneViewer project which the entire project can be run from. However, issues were found with adjusting lighting while running from the JAR file, which were not present when running in Eclipse. It is our suggestion that the project be run from Eclipse during operation.

The project hinges on JOGL, the openGL implementation in JAVA to be installed on the machine the project is running on. To install JOGL go to the JOGL website, https://jogl.dev.java.net/ , and download the current release build of the implementation. There are generally installation instructions that come with the download. Our suggestion is to install the JOGL files to every JDK (Java development kit) and JVM(Java Virtual Machine) on the computer. We encountered an issue when installing for the first time where we installed JOGL to the wrong virtual machine and our program failed to compile.

**Running the Program**

The program should be run as follows.

1. Run the SceneViewer class main method. You can do this either by double clicking on the JAR application or by running the class manually in Eclipse. This will open a blank window that does nothing. The SceneViewer uses a port to communicate with Matlab. If you run the program with the JAR or without command line arguments in Eclipse, then the default port is 6789. You can enter command line arguments for the port number for the application to use. You can also enter command line arguments for the window size. The 4 ways to run the program are:

()

(int width, int height)

(int port)

(int width, int height, int port)

2. Next in Matlab you need to add the Controller.class to the classpath. There are two demo Matlab scripts that show you how to do this. You can either add the JAR file to the classpath and import the Controller class through that, or add the com folder to the classpath and import the Controller class through the file system. Once the Controller is in the classpath, call the SceneViewer Matlab script/funciton and store its result in a variable. This function creates a Controller Java object. The script takes a port number as an argument. By default you should use 6789. If you changed the port that the SceneViewer uses then you need to change this port as well so they are the same.

3. Background on the controller. The controller immediately should connect with the already running empty scene when it is created. You can call functions from the controller object that you have created like a normal Java object. This is demonstrated in the demo.m and hand\_demo.m scripts. The controller works in 2 modes. One that buffers the commands that you call and does not send any until you call the flushCommands() method. And another which sends commands as you call them. You can turn buffering on and off by calling the setBufferCommands() method and passing it either true or false to turn the buffering on or off.

4. Load the XML file. Once the controller is created you need to send a command to load the scene file. Call the loadScene() method. You need to pass the string of the absolute path to the file into this method. By default command buffering will be enabled on the controller object so if the scene does not load after this command is called then call flushCommands() immediately after calling loadScene()

The scene should now be loaded and you can manipulate it using any of the commands built into the controller object. These are covered in detail later in this document.

**XML and model documentation**

The only model files that will work with this program are .obj aka Wavefront files with .mtl material files associated to them. These files can be made with most any 3d modeling software (Art of Illusion, Blender, Maya, 3dsMax). One issue we found was that if you use texture images with your material file the texture image needs to be in the same folder as the working directory you run the SceneViewer in. So if you are using the JAR it needs to be in the same folder as the JAR. If running from Eclipse it needs to be in the SceneViewer project directory in Eclipse.

The XML files are very simply defined. You can figure out most of how they work simply from looking at the example files. Here is an example file with documentation added to it. The green text is comments and are not part of the original XML file.

*This first line is simply a comment about the xml type*

<?xml version="1.0" encoding="UTF-8"?>

*We create a Scene object immediately in the file. The scene tag encloses all objects in the file.*

<Scene>

<Title>Demo Scene</Title> *The scene has a title with a string value which will show up in the title of the SceneViewer window*

<StereoRender>on</StereoRender> *This value sets the scene to render in stereo 3d or non-stereo 3d. If rendering in non-stereo 3d set the Camera IOD to 0.*

*<Object> This tag represents the start of an object*

<Name>pound1</Name> *This is the name of the object, when changing aspects of the scene you can access objects by name. Many objects in the file have a name tag, all names in a file must be unique within that file.*

<Parent>ROOT</Parent> *This is the parent of the object. This value should only be set to ROOT which is a reserved name defined in the program, or to the name of an object in the scene. The order of objects defined in the XML file does not matter, but this value must be the name of an object that exists if it is not ROOT.*

<Model>./data/board.obj</Model> *This is the path to the model file that this object should be rendered as. The material file will be found alongside it. This is a relative path from the JAR file/Eclipse project directory, but can also be an absolute path.*

<PositionX>0</PositionX> *These are the position of the object.*

<PositionY>-0.5</PositionY>

<PositionZ>0</PositionZ>

<RotationX>0</RotationX> *These are the rotation of the object using a quaternion.*

<RotationY>0</RotationY>

<RotationZ>1</RotationZ>

<RotationAngle>30</RotationAngle>

<ScaleX>1</ScaleX> *These are the scaling values for the object*

<ScaleY>1</ScaleY>

<ScaleZ>1</ScaleZ>

</Object >

*any number of objects can be put in the scene tags of this file. The order does not matter. Here is a second object that is a child of the first.*

<Object>

<Name>pound2</Name>

<Parent>pound1</Parent>

<Model>./data/board.obj</Model>

<PositionX>0</PositionX>

<PositionY>0</PositionY>

<PositionZ>25</PositionZ>

<RotationX>0</RotationX>

<RotationY>1</RotationY>

<RotationZ>0</RotationZ>

<RotationAngle>0</RotationAngle>

<ScaleX>1</ScaleX>

<ScaleY>1</ScaleY>

<ScaleZ>1</ScaleZ>

</Object >

*This is the camera object. It defines all aspects of the camera we view the scene through*

<Camera>

<FieldOfView>45</FieldOfView>*This is the field of view for the camera*

<NearClippingPlane>.1</NearClippingPlane> *This is the near clipping plane of the camera*

<FarClippingPlane>500</FarClippingPlane> *This is the far clipping plane of the camera*

<IntraOcularDistance>1</IntraOcularDistance> *This is distance between the virtual eyes of the camera.*

<PositionX>0</PositionX>*These are the position values of the camera*

<PositionY>0</PositionY>

<PositionZ>-100</PositionZ>

<UpX>0</UpX> *These are the values of the up vector of the camera*

<UpY>1</UpY>

<UpZ>0</UpZ>

<TargetX>0</TargetX> *These are the position values of the target of the camera*

<TargetY>0</TargetY>

<TargetZ>0</TargetZ>

</Camera>

*Only one Camera object should be defined in the XML file*

<GlobalLight> *This is an object that represents the global lighting in the scene*

<ColorR>0.0</ColorR> *These values are the RGBA color values for the global light*

<ColorG>0.0</ColorG>

<ColorB>0.0</ColorB>

<ColorA>1</ColorA>

<ClearColorR>0.0</ClearColorR> *These values are the RGBA values for the background or clear color of the scene*

<ClearColorG>0.0</ClearColorG>

<ClearColorB>0.0</ClearColorB>

<ClearColorA>1</ClearColorA>

<GL\_LIGHT\_MODEL\_LOCAL\_VIEWER>off </GL\_LIGHT\_MODEL\_LOCAL\_VIEWER> *This is either and on or off value to determine if GL\_LIGHT\_MODEL\_LOCAL\_VIEWER should be enabled. The value must be case sensitive either on or off*

<GL\_LIGHT\_MODEL\_TWO\_SIDE>off</GL\_LIGHT\_MODEL\_TWO\_SIDE> *This is either and on or off value to determine if GL\_LIGHT\_MODEL\_TWO\_SIDE should be enabled. The value must be case sensitive either on or off*

</GlobalLight >

*Only one GlobalLight object should be defined in the XML file*

<LightSource> *This is a light source object*

<Name>light1</Name> *This is the name of the object such that it can be accessed and modified while the scene is running*

<Number>1</Number> *This is the openGL light number of the object. the range of this number is dependent on the computer system. My computer said the highest number was 3377 and the lowest was 1. These need to be unique between light sources.*

<AmbientColorR>0</AmbientColorR> *These are the color values for the light source*

<AmbientColorG>0</AmbientColorG>

<AmbientColorB>0</AmbientColorB>

<AmbientColorA>1</AmbientColorA>

<DiffuseColorR>1</DiffuseColorR>

<DiffuseColorG>1</DiffuseColorG>

<DiffuseColorB>1</DiffuseColorB>

<DiffuseColorA>1</DiffuseColorA>

<SpecularColorR>1</SpecularColorR>

<SpecularColorG>1</SpecularColorG>

<SpecularColorB>1</SpecularColorB>

<SpecularColorA>1</SpecularColorA>

<PositionX>0</PositionX> *These are the position values of the light source*

<PositionY>5</PositionY>

<PositionZ>1</PositionZ>

<DirectionX>0</DirectionX> *These are the direction values of the light source*

<DirectionY>0</DirectionY>

<DirectionZ>-1</DirectionZ>

<Spot\_Cutoff>180</Spot\_Cutoff> *This is the spot cutoff value for the light source. It should be a double number between 0 and 90 for spot lights, and has the reserved value of 180 for point lights.*

<Intensity>1</Intensity> *This is the intensity value for the light*

<ConstAtt>1</ConstAtt> *This is the constant attenuation factor for the light*

<LinearAtt>0</LinearAtt> *This is the linear attenuation factor for the light*

<QuadAtt>0</QuadAtt> *This is the quadratic attenuation factor for the light*

</LightSource >

*Any number of light sources can be added to the scene. here are two more without comments.*

<LightSource>

<Name>light2</Name>

<Number>2</Number>

<AmbientColorR>0</AmbientColorR>

<AmbientColorG>0</AmbientColorG>

<AmbientColorB>0</AmbientColorB>

<AmbientColorA>1</AmbientColorA>

<DiffuseColorR>1</DiffuseColorR>

<DiffuseColorG>1</DiffuseColorG>

<DiffuseColorB>1</DiffuseColorB>

<DiffuseColorA>1</DiffuseColorA>

<SpecularColorR>1</SpecularColorR>

<SpecularColorG>1</SpecularColorG>

<SpecularColorB>1</SpecularColorB>

<SpecularColorA>1</SpecularColorA>

<PositionX>3</PositionX>

<PositionY>0</PositionY>

<PositionZ>1</PositionZ>

<DirectionX>0</DirectionX>

<DirectionY>0</DirectionY>

<DirectionZ>-1</DirectionZ>

<Spot\_Cutoff>180</Spot\_Cutoff>

<Intensity>1</Intensity>

<ConstAtt>1</ConstAtt>

<LinearAtt>0</LinearAtt>

<QuadAtt>0</QuadAtt>

</LightSource >

<LightSource>

<Name>light3</Name>

<Number>3</Number>

<AmbientColorR>0</AmbientColorR>

<AmbientColorG>0</AmbientColorG>

<AmbientColorB>0</AmbientColorB>

<AmbientColorA>1</AmbientColorA>

<DiffuseColorR>1</DiffuseColorR>

<DiffuseColorG>1</DiffuseColorG>

<DiffuseColorB>1</DiffuseColorB>

<DiffuseColorA>1</DiffuseColorA>

<SpecularColorR>1</SpecularColorR>

<SpecularColorG>1</SpecularColorG>

<SpecularColorB>1</SpecularColorB>

<SpecularColorA>1</SpecularColorA>

<PositionX>-3</PositionX>

<PositionY>0</PositionY>

<PositionZ>1</PositionZ>

<DirectionX>0</DirectionX>

<DirectionY>0</DirectionY>

<DirectionZ>-1</DirectionZ>

<Spot\_Cutoff>180</Spot\_Cutoff>

<Intensity>1</Intensity>

<ConstAtt>1</ConstAtt>

<LinearAtt>0</LinearAtt>

<QuadAtt>0</QuadAtt>

</LightSource >

*This tag closes the scene and the file*

</Scene >

Now that you see what an XML scene file looks like, we suggest you keep a few simple scenes premade and build new scenes off of the old ones to save time and ensure that the scenes have correct capitalization and spelling.

**Controller Documentation**

The following are the commands that can be called from the controller object.

Most of the commands are self evident so documentation is thinner here

void close () throws IOException

void loadScene (String path) throws IOException

Loads the designated scene into the viewer

void quit () throws IOException

quits the application and closes the socket

void setObjectPosition (String name, double x, double y, double z) throws IOException

void setObjectRotation (String name, double x, double y, double z, double angle) throws IOException

void setObjectScale (String name, double x, double y, double z) throws IOException

void setObjectColorAmbient (String name, float r, float g, float b) throws IOException

void setObjectColorDiffuse (String name, float r, float g, float b) throws IOException

void setObjectColorSpecular (String name, float r, float g, float b) throws IOException

void setObjectTransparency (String name, float i) throws IOException

void setObjectDraw (String name, boolean draw) throws IOException

void setCameraPosition (double x, double y, double z) throws IOException

void setCameraTarget (double x, double y, double z) throws IOException

void setCameraUpVector (double x, double y, double z) throws IOException

void setCameraFovNF (double Fov, double N, double F) throws IOException

void setCameraIOD (double IOD) throws IOException

void setGlobalLightValues (float r, float g, float b, float a) throws IOException

void setClearColor (float r, float g, float b, float a) throws IOException

void setLightOptions (boolean local\_viewer, boolean two\_side) throws IOException

void setLightColorAmbient (String name, float r, float g, float b, float a) throws IOException

void setLightColorDiffuse (String name, float r, float g, float b, float a) throws IOException

void setLightColorSpecular (String name, float r, float g, float b, float a) throws IOException

void setLightOn (String name, boolean enable) throws IOException

void setLightValues (String name, float intensity, float constant\_attenuation\_constant, float linear\_attenuation\_constant, float quad\_attenuation\_constant) throws IOException

void setLightPosition (String name, double x, double y, double z) throws IOException

void flushCommands () throws IOException

boolean isBufferCommands ()

returns true or false as to whether the bufferCommands flag is enabled

void setBufferCommands (boolean bufferCommands)

clears all commands that are currently buffered and sets the buffer commands flag

More documentation can be found in the doxygen pages in the repository.