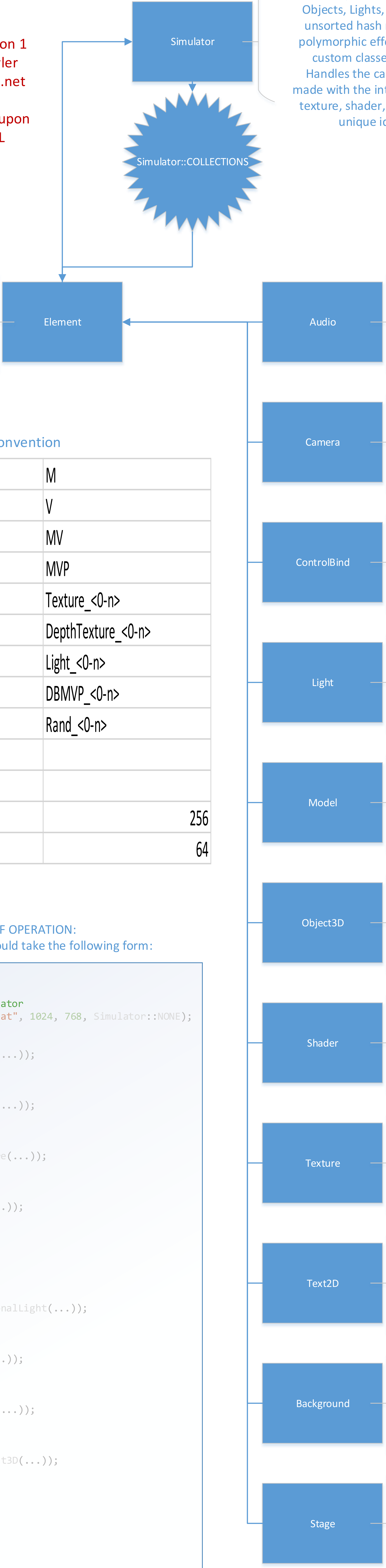
openGL Simulator Version 1
Created By: Jared Fowler
yrolg4tseuq@sbcglobal.net

Mini game-engine built upon
O-O C++ and openGL



Simulator handles the creation/deletion of the GLFW window, and manages the creation/removal of several collections of different element pointers, { Audio, Cameras, Models, Shaders, Textures, Objects, Lights, Controls, Text2D } Each collection is held in an unsorted hash map for O(1) lookup. Class Element takes on a polymorphic effect, allowing the future user to make their own custom classes which can then be added to the collection. Handles the calling of update/draw functions. Simulator was made with the intent of having only a single copy of every model, texture, shader, etc. Within each collection, the elements have unique id's which other elements can reference.

Base class. Contains an unique id per collection and a pointer to the Simulator Class

Uses the irrklang freesource library. The class simply acts as an API.

I've provided 3 basic camera types. Camera will be static and positioned based upon inputs. VectorCam will create a vector between 2 objects and move with those objects. ChaseCam and FollowCam are similar, except ChaseCam will turn to look at what the object is looking at.

Allows the user to provide custom inputs. The input can be a combination of a key and modifier, such as shift, alt, etc.. This is linked with a callback function. System reserved inputs are F1-F12.

Light is a bit more trick b/c it deals with shadow rendering. There are 4 types of lights provided. Static, directional, spot, and point. These classes set up the proper framebuffers/ texture to handle shadow rendering. It is still up to the user to implement the shader correctly.

Referring to the collection of vertices that make up the model. Reads a format I created called .IBO. You'll need to use the side program I've created to convert your .obj's to this format!

Contains the basic transformations that could take place. Rotation, Translation, Scale. Handles the update and drawing of a model (both are virtual functions)

Handles the reading of shader files and binding its various class members to the various possible uniforms located in the shader. This is done by naming protocol, which can be seen to the left.

Loads a .DDS format texture into the GPU and saves a handle to that texture. GIMP has a nice plug-in to export .DDS vs 5 format.

This class is actually not part of the Simulator Collections. It is a member of Simulator, however, but manages its own collection internally. Allows the user to add 2D text onto the screen. Relies upon the proper shader and texture being loaded.

Not a collection. An implementation of a very simple skybox which takes in 6 texture names.

Not a collection. Pointer to an Object3D that could act as a stage.

Shader Protocol Naming Convention

Model Matrix	M
View Matrix	V
Model-View Matrix	MV
Model-View-Projection Matrix	MVP
Textures (n = 64)	Texture_<0-n>
Depth Textures (n = 64)	DepthTexture_<0-n>
Lights (n = 64)	Light_<0-n>
Depth Bias MVP (n = 64)	DBMVP_<0-n>
Random Float (n = 64)	Rand_<0-n>
MAX FILE NAME LENGTH	256
MAX ID LENGTH	64

ORDER OF OPERATION:
Your main function should take the following form:

```
int main(void){

   //Instantiate the Simulator
   Simulator s("CSUN CubeSat", 1024, 768, Simulator::NONE);

   //Add Shaders
   s.addShader(new Shader(...));
   //...etc

   //Add Depth Shaders
   s.addShader(new Shader(...));
   //...etc

   //Add Textures
   s.addTexture(new Texture(...));
   //...etc

   //Add Models
   s.addModel(new Model(...));
   //...etc

   //Add Text2Ds
   s.addText2D(...);
   //...etc

   //Add Lights
   s.addLight(new DirectionalLight(...));
   //...etc

   //Add Audio
   s.addAudio(new Audio(...));
   //...etc

   //Add Cameras
   s.addCamera(new Camera(...));
   //...etc

   //Add instances
   s.addInstance(new Object3D(...));
   //...etc

   //Set Background
   s.setBackground(...);

   //Set Stage
   s.setStage(...);

   //Start Audio

   //Set Camera

   //ETC...

   //Start Simulation
   s.runSimulation();

}
```

```
Simulator Reserved KEYS :
F1->Free Mouse
F2->Free View(Allows you to use the mouse and arrow keys to
move around)
F3->Next Camera
F4->Hide / Show Text - 2D
F5->Mute / UnMute Audio //Not implimented yet :-( TODO
F6->Max UPS / FPS
F7->On / Off Shadow Rendering
F11->Show / Hide FPS
F12->Show / Hide About
```

I've worked very hard to produce all of this. The reader is welcome to study the code for educational purposes, but any use of the code should properly reference its creator, Jared Fowler. Thanks.