

Coursework

Games Programming 2

|  |  |
| --- | --- |
| **Module Name:** | Game Programming 2 |
| **Module Leader:** | Bryan Young |
|  |  |
| **Student ID:** | S1918159 |
| **Student Name:** | George Gordon Guthrie |

*I confirm that the code contained in this file (other than that provided or authorised) is all my own work and has not been submitted elsewhere in fulfilment of this or any other award*.

*Signature*. G.Guthrie

Github link: https://github.com/Georgeguthrie/Games-Programming-2

**Contents**

1. BGMusic.h…………………………………………………………………………………………………………………………….3

2. BGMusic.cpp…………………………………………………………………………………………………………………………3

3. Game\_Window.h………………………………………………………………………………………………………………….3

4. Game\_Window.cpp………………………………………………………………………………………………………………4

5. Main.cpp………………………………………………………………………………………………………………………………4

6. MainGame.h…………………………………………………………………………………………………………………………4

7. MainGame.cpp……………………………………………………………………………………………………………………..5

8. Obj\_loader.h/.cpp…………………………………………………………………………………………………………………6

9. Object\_Mesh.h……………………………………………………………………………………………………………………..6

10. Object\_Mesh.cpp………………………………………………………………………………………………………………..6

11. Object\_Shader.h……………………………………………………………………………………………………………….…7

12. Object\_Shader.cpp…………………………………………………………………………………………………………...…7

13. Object\_Texture.h………………………………………………………………………………………………………….……..8

14. Object\_Texture.cpp…………………………………………………………………………………………………….……….8

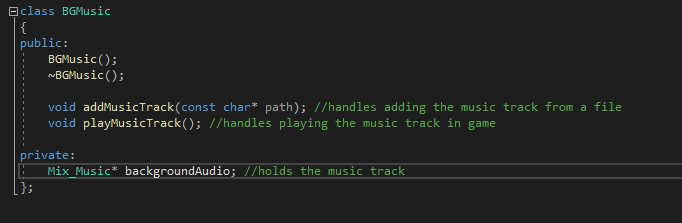
15. Object\_Transform.h………………………………………………………………………………………………….………...9

16. Stb\_image.h/.c……………………………………………………………………………………………………….…………...9

17. viewport.h………………………………………………………………………………….………………………..……………...9

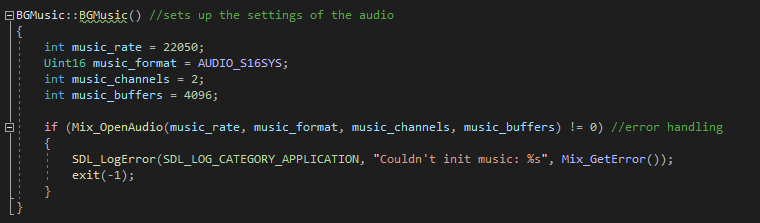
**1. BGMusic.h**

In this header file there is the constructor and destructor responsible for handling the audio of the project. Functions are declared for handling music being added from a file as well as played. There is also a variable named “backgroundAudio”



**2. BGMusic.cpp**

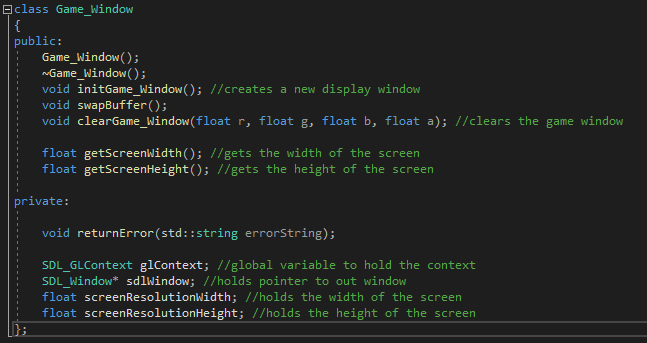
This script handles the code for the audio of the game. The settings for the audio are established when the script is called.



Two functions are declared below this that when called load the music track from a file and play the music track when called.

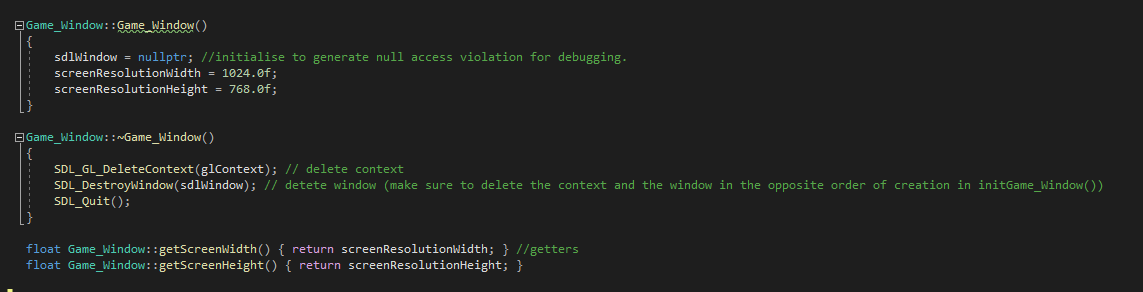
**3. Game\_Window.h**

In this header file there is the constructor and destructor as well as multiple functions responsible for creating the display used in the game. Variables are also declared for holding important elements of the screen such as the resolution of the screen.

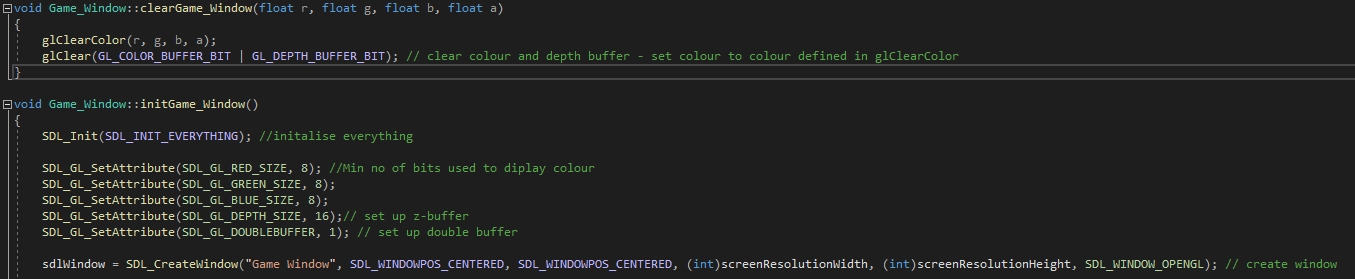


**4. Game\_Window.cpp**

This script handles the creation of the display of the game. The resolution of the window is declared at first and acquired by the code and error handling is established.

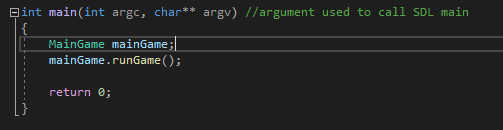


Next a function is declared for clearing the display and another function is created that will initialize everything. In this function all the attributes of the display relating to colour are declared and the display is created using the declared resolution and attributes.



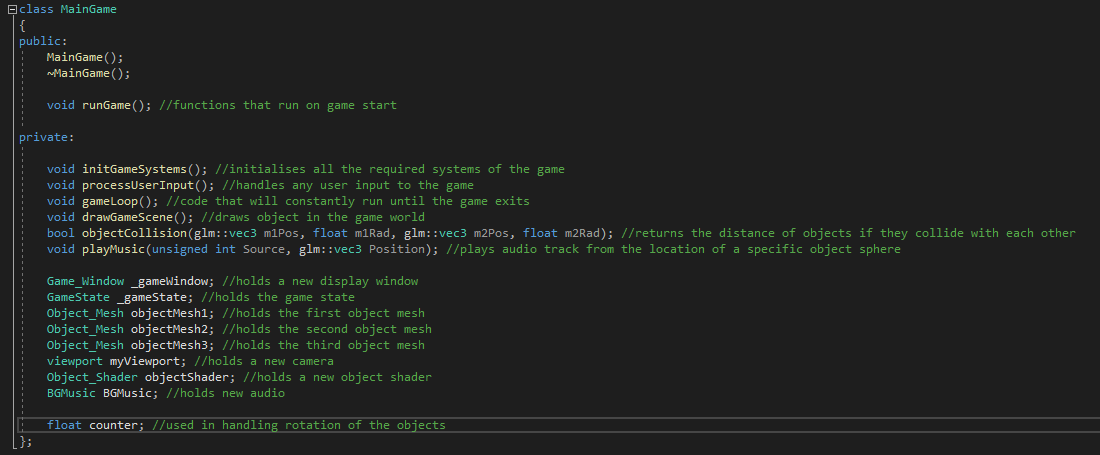
**5. Main.cpp**

This short script is called on startup of the program and exists purely to call the runGame function in the MainGame script.



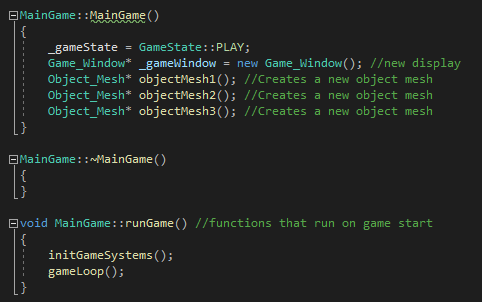
**6. MainGame.h**

In this header file there is the constructor and destructor as well as multiple functions responsible for handling the main functions of the game. Multiple variables are also declared for handling the various components of the game such as the three objects, the camera and the music.

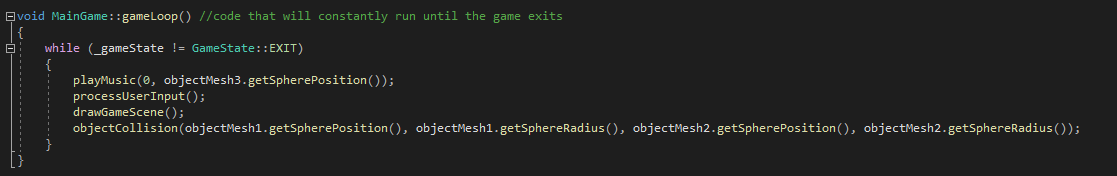


**7. MainGame.cpp**

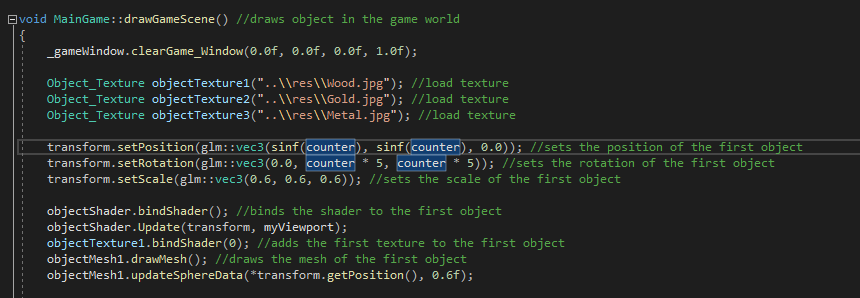
This script handles the main functionality of the game. Firstly, three meshes and a new display are created, the game systems are initialized, and the game loop begins.



The initGameSystems function will create a new game window and load the various resource files into the project such as the music track, shaders and object models. The gameLoop function handles all the code that needs to be run constantly during the project such as the music being played and the collision of the objects being registered.



The most important function that is called in the gameLoop is the drawGameScene function that handles the position, rotation and scale of each of the models as well as binding the shaders and textures onto each model. A counter variable is used to handle the rotation of the objects in the scene.

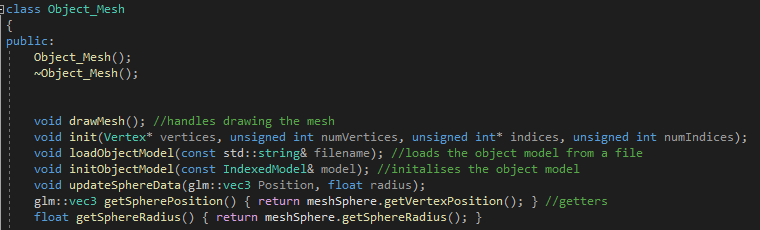


**8. OBJ\_Loader.h/.cpp**

These files allow the loading of .obj file type 3d models from a file for use in the project. These scripts were provided during a practical lab

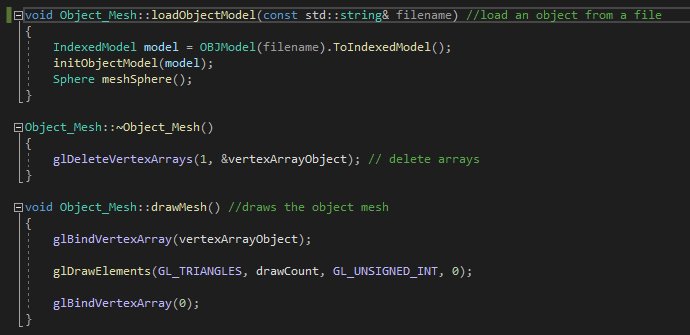
**9. Object\_Mesh.h**

In this header file there is the constructor and destructor that handle the creation the meshes of the objects as well as functions for loading the object models and drawing the meshes. Two structs also exist in this script for handling the various variables of the vertices and spheres



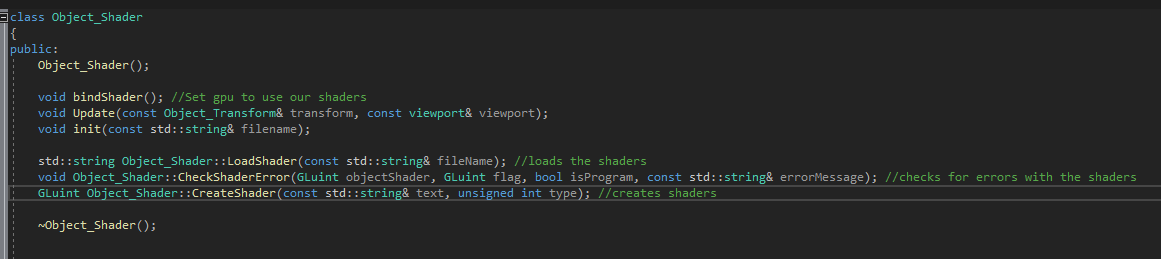
**10. Object\_Mesh.cpp**

This script handles the creation of the meshes of the objects. Various functions exist in this script for loading object models from files and drawing the meshes when initialized. This script also handles the data of the vertices and spheres of the objects for use in the MainGame script



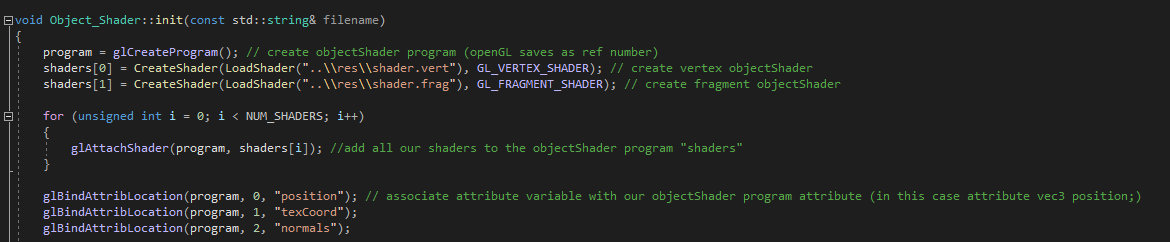
**11. Object\_Shader.h**

In this header file there is the constructor and destructor that handle the shaders of the objects. Functions are declared in here that will load and create the shaders as well as bind the GPU to use the shaders.

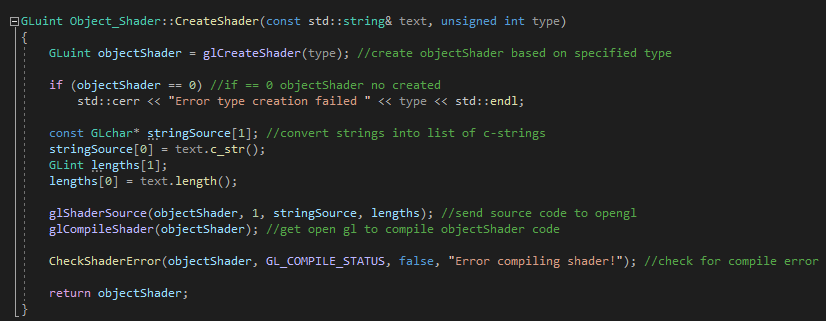


**12. Object\_Shader.cpp**

This script handles the creation of the shaders of the objects. When the script is initialized, the shaders will be loaded from files.

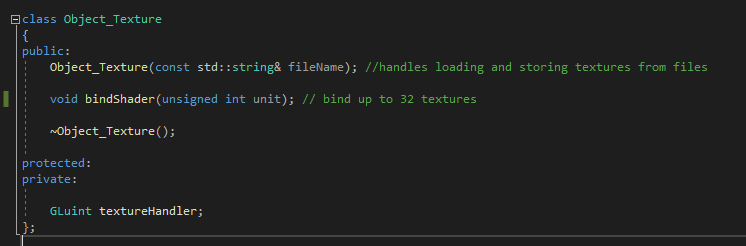


Functions then exist in this script for creating and binding the shaders to the GPU for use in the MainGame script.



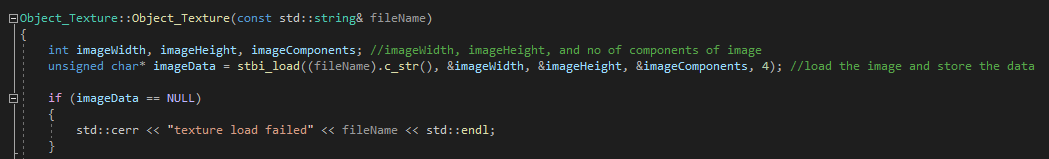
**13. Object\_Texture.h**

In this header file there is the constructor and destructor that handle the storing and loading of textures from the files. A function also exists for use in binding the textures using shaders.

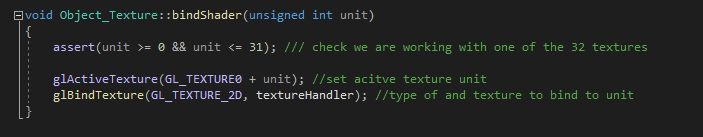


**14. Object\_Texture.cpp**

This script handles the textures used in the project. When initialized the script will load in a texture from a file and store the image data.

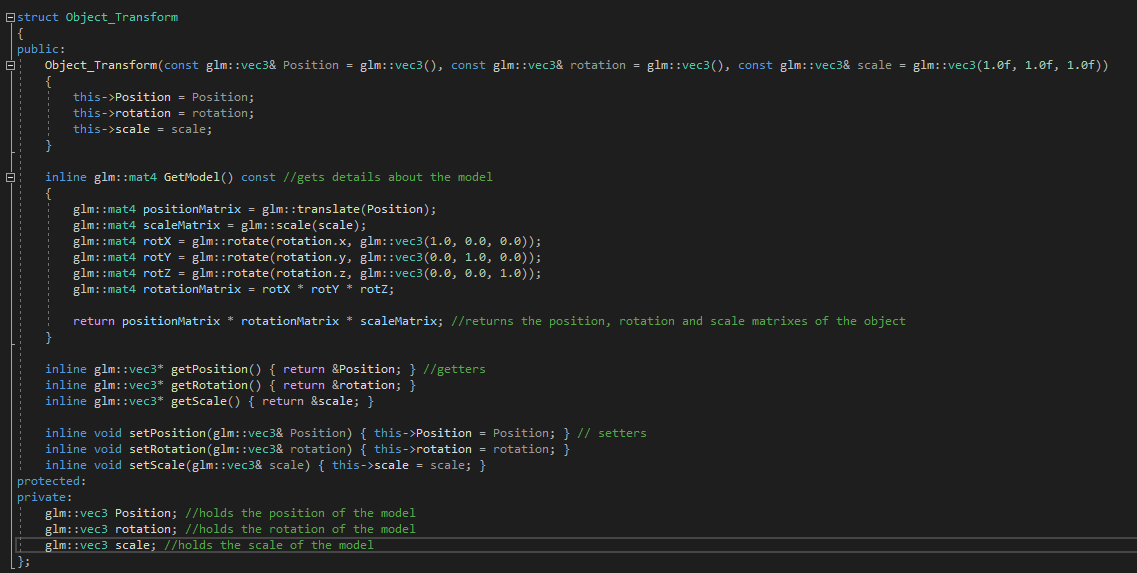


A function then exists for use in binding the loaded texture to a shader for use in the project.



**15. Object\_Transform.h**

This header file is responsible for handling the various transformation variables of the objects. The file will acquire the position, rotation and scale of each of the objects for use in the MainGame.cpp script. Variables exist here for holding the various matrixes of the objects



**16. stb\_image.h/c**

These files allow the loading of .jpg file type images from a file for use in the project as textures. These scripts were provided during a practical lab

**17. Viewport.h**

This header file is responsible for handling the camera of the project. The file when called will create a camera at a specific position. Some getters will acquire the position and view projection of the camera for use in the project.

