

Games Programming 2

Coursework

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
| **Student ID:** | S1908212 |
| **Student Name:** | Damyan Furenesa |

*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*: D. FURENESA

Contents

[1 Header Files 3](#_Toc93007451)

[1.1 MainGame.h 3](#_Toc93007452)

[1.2 Display.h 3](#_Toc93007453)

[1.3 Mesh.h 4](#_Toc93007454)

[1.4 Camera.h 5](#_Toc93007455)

[1.5 transform.h 6](#_Toc93007456)

[1.6 BGAudio.h 6](#_Toc93007457)

[2 Script Files 7](#_Toc93007458)

[2.1 MainGame.cpp 7](#_Toc93007459)

[2.2 Main.cpp 8](#_Toc93007460)

[2.3 Display.cpp 8](#_Toc93007461)

[2.4 Mesh.cpp 9](#_Toc93007462)

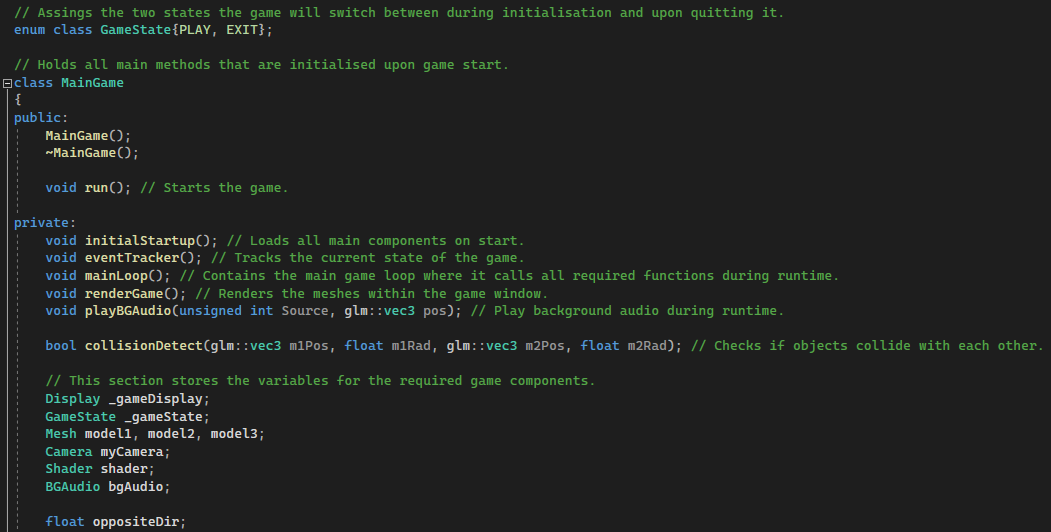
[2.5 BGAudio.cpp 10](#_Toc93007463)

[3 GitHub Repo 10](#_Toc93007464)

# Header Files

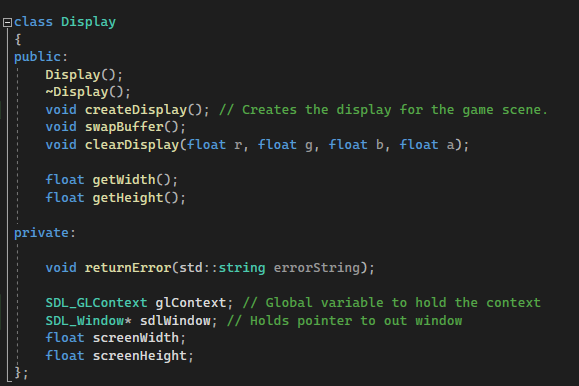
## MainGame.h

This header contains the required functions and variables for the main script to handle and execute. The variables store data for the game’s display window, the current game state, all three objects, game camera, shader and audio, with the float variable named “oppositeDir” used for the objects rotation and movement.



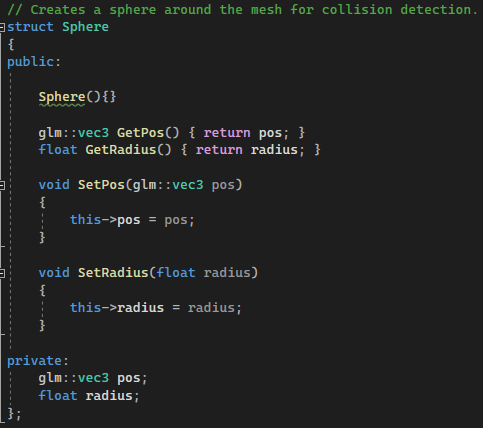
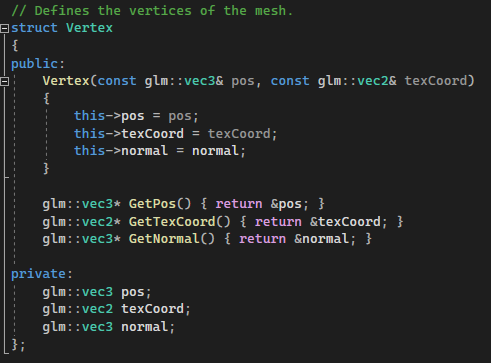
## Display.h

This header holds required variables for screen width and height and stores functions for the making of the game display window.



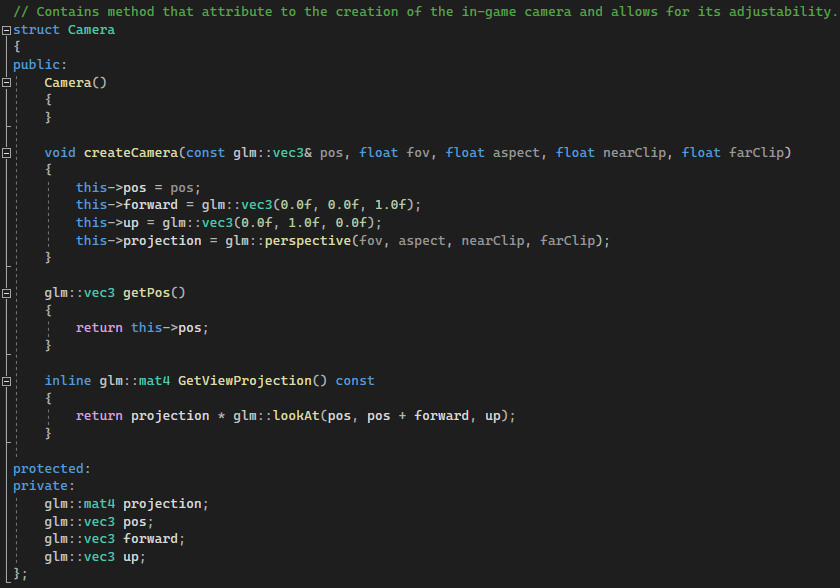
## Mesh.h

This header contains functions responsible for rendering objects into the scene. It also stores and manages variables that hold information for vertices and spheres that are later used for collision detection.



## Camera.h

This header contains functions for creating an adjustable camera within the game scene. Through the createCamera() method you are able to change the position of the camera by editing the vector3 values and adjust it based on your needs.

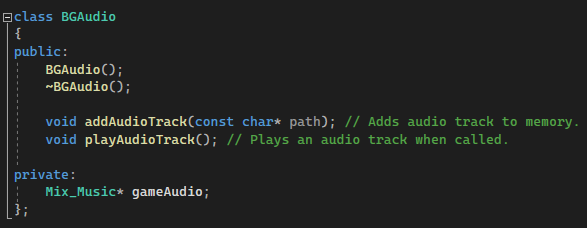


## transform.h

The header contains the Transform data structure that creates a new datatype responsible for the movement of the objects within the scene. It stores necessary getters and setters that receive the current position of the objects and can update that information based on our input to change their scale, position or rotation.

## BGAudio.h

The header contains functions that handle the game audio. addAudioTrack() lets us add an audio file to the queue, while the playAudioTrack() method is responsible for its playback.

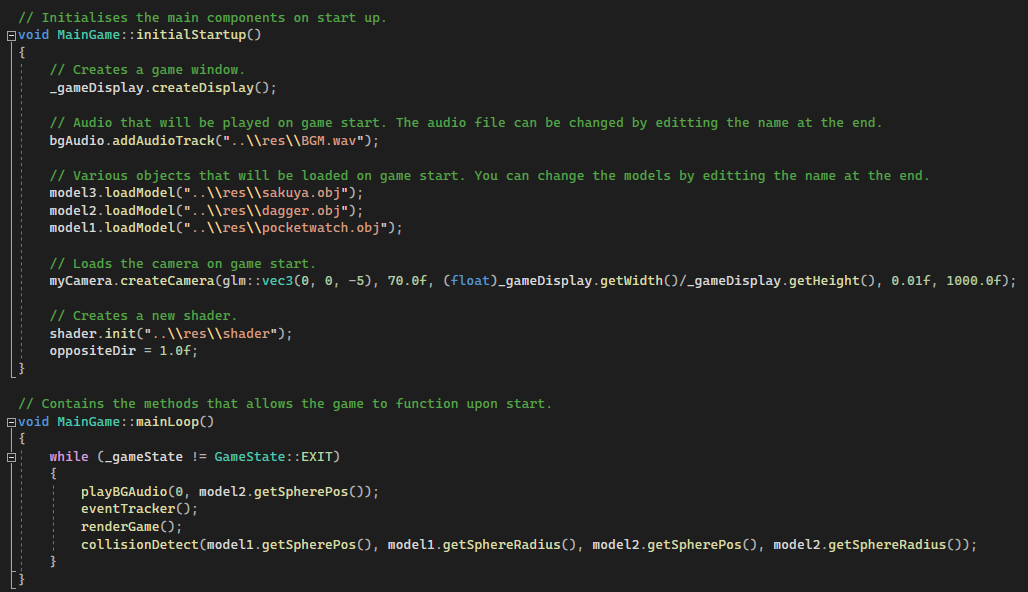


# Script Files

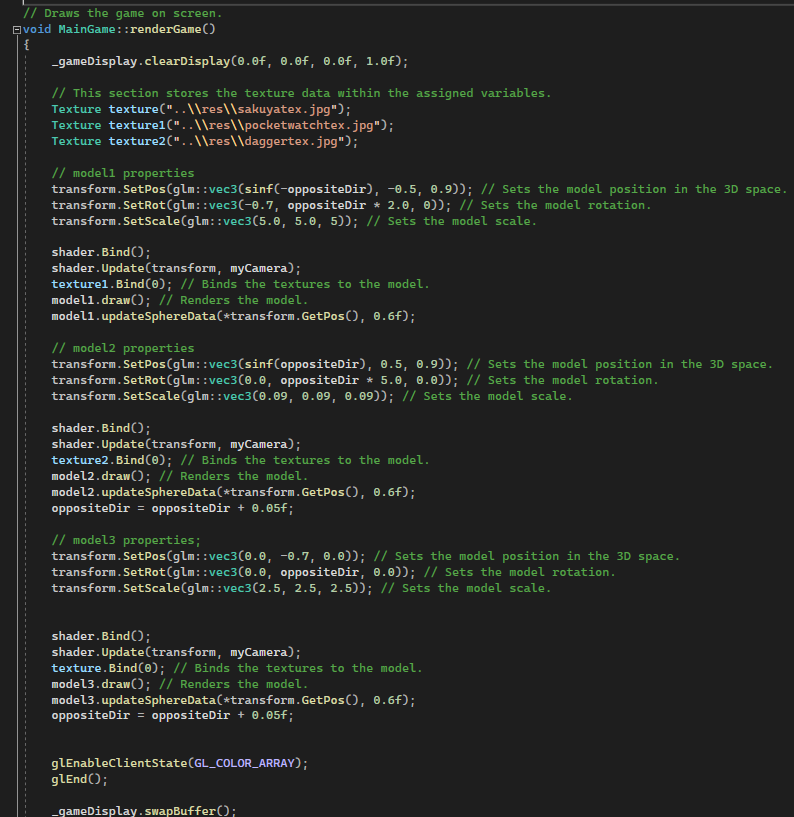
## MainGame.cpp

The MainGame script contains the core functions of the project. It begins by setting up the game for initialisation and then transitioning to the main game loop. On start up (**initialStartup()** method), main components and game window are being created. Audio files are being stored at the start and objects are loaded into the scene. The adjustable camera is also loaded in allowing us to view the scene.

The main game loop consists of all main functions being executed in order when the game state is set to “PLAY”. Background audio starts being played, the event tracker (used to monitor the current state of the game) is initialized and the meshes are being drawn on screen along with the collision detection.



Lastly, the renderGame() script contains functionality used to draw all meshes and camera on the screen. It also holds the various properties for each model that allows us to adjust their scale, position and rotation, including how fast they rotate or move around the scene. Within the script we also bind the shaders and textures to each model.

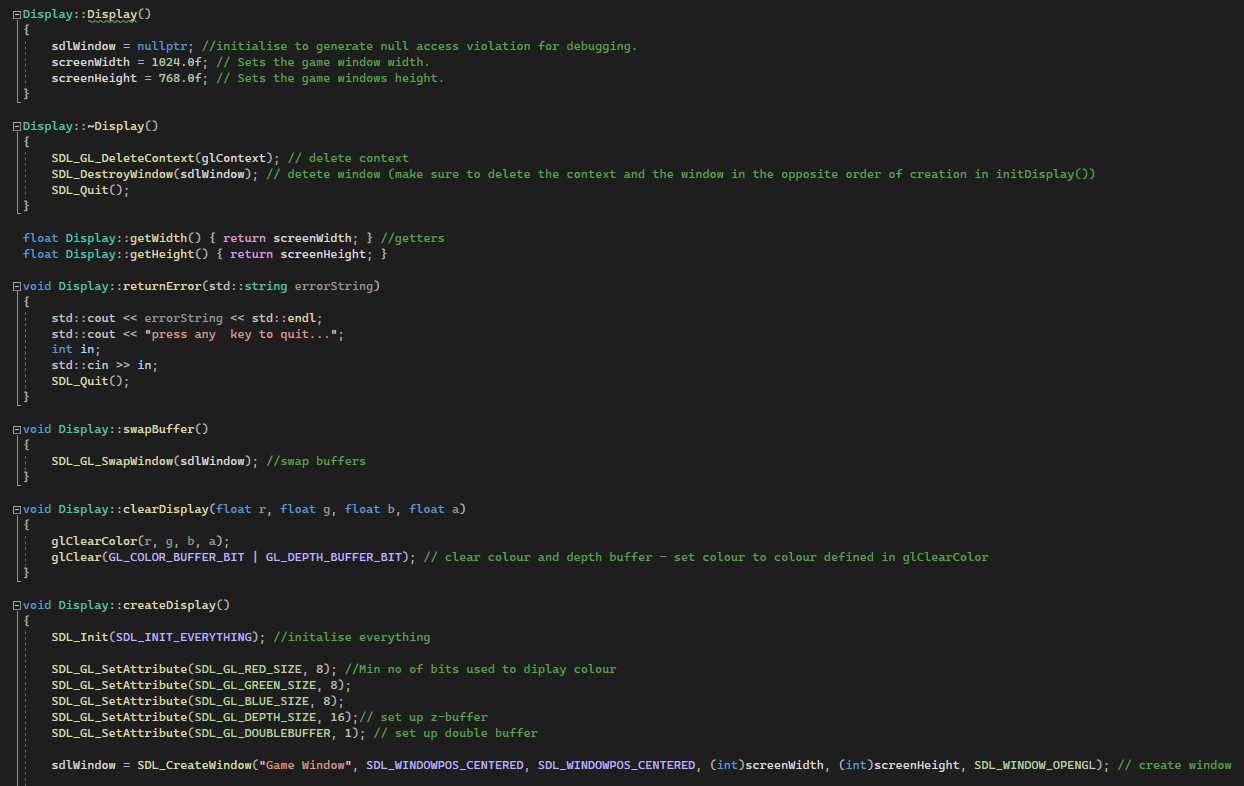


## Main.cpp

Loads all game functions on start up. Called only when the game is initialised.

## Display.cpp

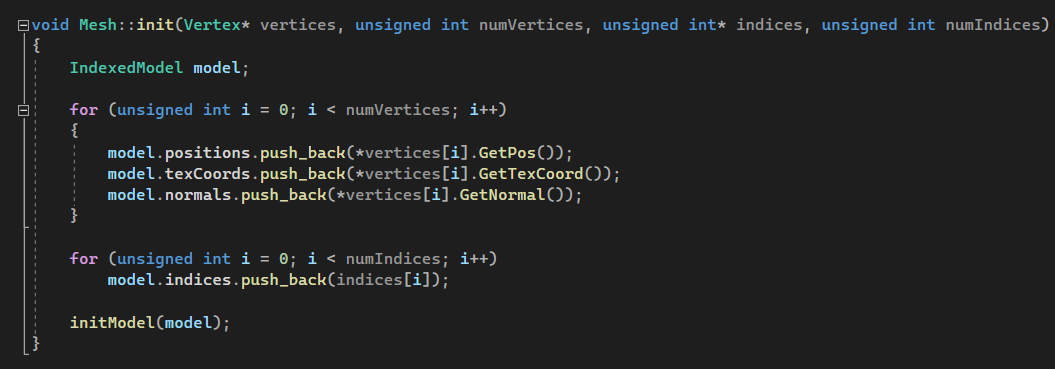
This script is in charge of creating the game's display. The window's resolution is first stated, then obtained by the code, and error handling is implemented.



Following that, a function for clearing the display is declared, followed by a function to initialise everything. All of the display's colour properties are defined in this function, and the display is built using the declared game window size and properties.

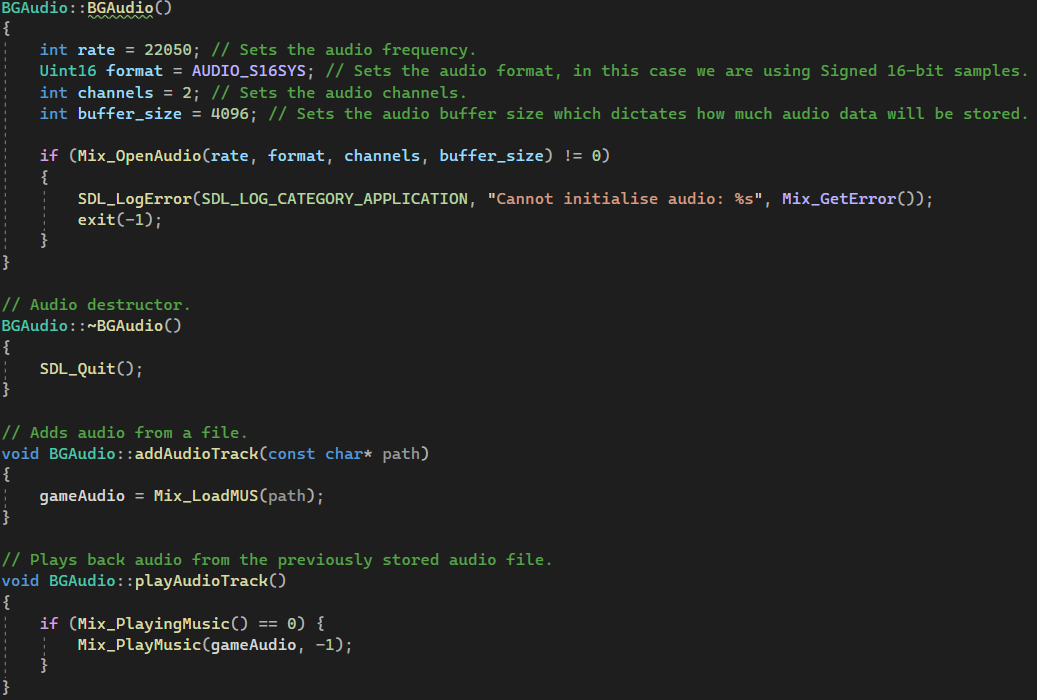
## Mesh.cpp

The Mesh script is responsible for generating meshes for the models when loaded from files in the project. It also holds methods that defines the vertices of the newly made mesh and generates a sphere used for collision detection.



## BGAudio.cpp

The BGAudio script is responsible for the handling of the in-game background audio using SDL Audio. It begins by initialising the required audio parameters for its frequency, audio format (in this case, Signed 16-bit samples), the audio channels and buffer.



We have two methods within the script. The first one loads an audio file from a file path and stores it in the gameAudio variable. The second one plays back the previously stored audio file when the method is called.

# GitHub Repo

<https://github.com/Damyan-dev/GP2-CW>