

Coursework for

Games Programming

|  |  |
| --- | --- |
| **Module Name:** | Games Programming 2 |
| **Module Code:** | M3I626039 |
| **Module Leader:** | Bryan Young |
|  |  |
| **Student ID:** | S2222427 |
| **Student Name:** | Ektor Zoidis |

**By submitting this assignment, I agree with the following statements:**

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: *E. ZOIDIS*

Contents

[1 Classes & Structs 3](#_Toc124716912)

[1.1 MainGame 3](#_Toc124716913)

[1.2 Display 3](#_Toc124716914)

[1.3 Camera 4](#_Toc124716915)

[1.4 Mesh 4](#_Toc124716916)

[1.5 Transform 5](#_Toc124716917)

[1.6 Texture 6](#_Toc124716918)

[1.7 Shader 6](#_Toc124716919)

[1.8 SDLAudio 7](#_Toc124716920)

[1.9 Skybox 7](#_Toc124716921)

[2 References 8](#_Toc124716922)

# Classes & Structs

## MainGame

The MainGame.h file contains all the fields, properties and method declarations for the main game class. This class is responsible for handling the core functionality of the game such as initialising, running, processing and drawing the game components and holds the data necessary to do these functions.

The MainGame.cpp file contains all the definitions for the class methods where initSystems() initialises and loads the components; gameLoop() runs the core functionality while the game plays; processEvents() handles events such as key input or exiting the game; collision() handles collision between two objects; and drawGame() draws the components on the screen.



## Display

The Display.h file contains all the fields, properties and method declarations for the display class. This class is responsible for creating a window and displaying/drawing the game components on the screen.

The Display.cpp file contains all the definitions for the class methods where initDisplay() creates the window and its properties by the declared resolution; returnError() interrupts the process and outputs an error when something fails; clearDisplay() resets the depth buffer and color; and swapBuffer() swaps the rendering buffer.



## Camera

The Camera.h file contains all the fields, properties, method declarations and definitions for the camera struct. This struct handles functionality such as projecting the camera view to the display and moving/rotating it in the 3D space. initCamera() creates the camera with the given properties such as the initial position, FOV, aspect, near and far clips, and functions such as MoveForward(), Pitch(), RotateY() allow control over the cameras position, rotation and projection. It also includes getters and setters to get and set important information for the players view.



## Mesh

The Mesh.h file contains all the fields, properties and method declarations for the mesh class, vertex and sphere struct, and the definitions for the vertex and sphere struct. These handle the generation and display of models with the use of vertices and indices and the generation of spheres with a given radius that are used for collision detection.

The Mesh.cpp file contains all the definitions for the class methods where init() generates a model by the given vertices, indices and initialises it; initModel() generates and binds the vertex arrays, buffers and sets the attributes and data for the model; loadModel() takes an OBJ model, reads it and initialises it; draw() binds the vertex arrays and draws them; and updateSphereData() updates the model sphere position and radius.

Text

Description automatically generated

## Transform

The Transform.h file contains all the fields, properties and method declarations and definitions for the transform struct. This struct holds data such as the position, rotation and scale which can be used for any object. The initial values can be set by the constructor and can be read with the getters and modified with the setters. It can also read the data of a model with GetModel() and return a matrix of its position, rotation and scale.

Text

Description automatically generated

## Texture

The Texture.h file contains all the fields, properties and method declarations for the texture class. This class is responsible for binding and drawing textures on objects.

The Texture.cpp file contains all the definitions for the class methods where the constructor takes an image, generates the texture, binds it and sets its parameters; and Bind() binds the texture to an object.

Text

Description automatically generated

## Shader

The Shader.h file contains all the fields, properties and method declarations for the shader class. This class is responsible for creating, loading, initialising, activating and updating a shader.

The Shader.cpp file contains all the definitions for the class methods where init() loads the shader file, creates the shader, compiles it and links it; LoadShader() takes a shader file, structures it and returns it if valid; Activate() activates the shader program; Update() updates the shader data; Delete() deletes the shader program; and compileError() checks if there is a shader error and outputs it.

Text

Description automatically generated

## SDLAudio

The SDLAudio.h file contains all the fields, properties and method declarations for the SDLAudio class. This class is responsible for reading audio files such as SFX and tracks and playing them.

The SDLAudio.cpp file contains all the definitions for the class methods where the constructor sets up the audio data and outputs an error if failed; addSoundEffect() takes a WAV file, pushes it and outputs an error if failed; addAudioTrack() takes an audio file, loads it into a music object and outputs an error if failed; playSoundEffect() takes an index int, plays a sound effect with the given index and outputs an error if out of range; and playAudioTrack() plays the added tracks.

Text

Description automatically generated

## Skybox

The Skybox.h file contains all the fields, properties and method declarations for the skybox class. This class creates a cubemap, loads the textures of the skybox to it and displays it on the screen with the respect to the cameras perspective. It also uses a custom shader which uses a samplerCube for its fragment and a projection and view matrix for its vertex.

The Skybox.cpp file contains all the definitions for the class methods where init() initialises and loads the cubemap; initCubemap() creates the vertices and indices of the cubemap, generates the vertex array and buffers, binds them and sets the attributes and data; loadCubemap() loads the 6 textures of the skybox, generates them, sets the parameters and outputs an error if failed; and draw() binds the vertex array and textures and draws them.

Text

Description automatically generated

# References & Controls

Skybox/Cubemaps Video Tutorial

GORDAN, V., 2021. OpenGL Tutorial 19 - Cubemaps & Skyboxes [video]. Jun 18, 2021. [viewed 8 January 2023]. Available from: <https://www.youtube.com/watch?v=8sVvxeKI9Pk>

UFO Model

GRAPHFUN, 2019. UFO [3d model]. [viewed 11 January 2023]. Available from: https://sketchfab.com/3d-models/ufo-f7ac46de718a444384a73e953d49997c

Astronaut Model

CAROLINA P., 2016. astronaut [3d model]. [viewed 11 January 2023]. Available from: <https://sketchfab.com/3d-models/astronaut-47a23e3f2cfc49a7888b4d0d7e2ad1a2>

Camera controls:

W, A, S, D – Move forward, left, back, right

Q, E – Rotate left, right

Z, X – Pitch up, down

Scroll wheel – Zoom in, out