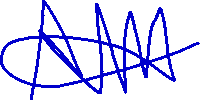
Anna Irene Margaritis S1905566  
Computer Games (Software Development), Games Programming 3 coursework documentation

*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*.



Contents

[1. Overview 3](#_Toc123492016)

[2. Camera Modes 4](#_Toc123492017)

[3. Minimap 6](#_Toc123492018)

[4. Collision and Combat 7](#_Toc123492019)

# Overview

The finished project can be found here: <https://github.com/voidirene/GP3-CWK>  
Video demo is available at this link: https://youtu.be/Qge5AF0uA\_U

The game’s controls are as follows:

* WASDQE to move
* F to fire
* T to toggle between different camera views
* L to lock/unlock free camera movement

When in free camera mode:

* Arrow keys to pan camera
* Mouse click & drag to rotate camera
* +, - to zoom camera in/out
* 1 to center camera on mesh
* 2 to point camera at mesh
* 3 to rotate camera around mesh clockwise
* 4 to rotate camera around mesh anti-clockwise

Asteroids appear on different points on the screen within a certain field every time the game is launched. The player can fire a laser bullet at the asteroids to destroy them, but also needs to carefully move around them to not destroy their spaceship. The game can be quit at any time using the ESC key.

# Camera Modes

The game features two different camera views and a free camera mode.

The first is a top-down or ‘eagle-view’ camera. This is the camera mode that is on by default when launching the game. When in this mode, the camera will move to chase the spaceship and always keep it in the centre of its view.

The second mode places the camera at a point behind the spaceship and points the camera towards it. This mode is useful for judging the depth that the spaceship is at, as asteroids can also spawn in the z-axis.

During either of these two camera views, the player may use the L key to ‘unlock’ the camera, allowing them to use the arrow keys and the mouse to freely move and rotate the camera. They may also use the + and – keys to zoom the camera in and out respectively. Available for them are also 1, 2, 3 and 4 keys, which will centre the camera on the mesh, point it at the mesh, and rotate around it clockwise and anticlockwise respectively.

   
*Figure 1: Code snippet showing the input handling functions for free camera movement*

Switching between camera views is done via a Boolean that changes value every time the player hits the toggle key (T). When rendering the scene, the program will get the camera’s mode and place the camera accordingly. Similarly, if the player chooses to unlock the camera, a different Boolean gets modified. It is important to note that the camera’s position will not be updated automatically if the cameraLock Boolean is false, to allow the player to move the camera themselves.

   
*Figure 2: Code snippet showing the two Booleans that determine which view the camera should use*

# Minimap

Another of the main features of the game is a small minimap on the top left corner of the screen. The minimap allows the player to easily orient themselves as not all asteroids may always be visible in the main view.

The minimap is rendered onto the screen using a second FBO, and an FBO grayscale shader is applied to it. The main display FBO is rendered onto a quad which is the size of the game window, while the minimap FBO is rendered onto a small quad on top of the former.

Due to the implementation of the minimap, the game’s rendering code has had its structure shifted a bit. The main function is called UpdateScreen(), and handles the rendering of the two FBOs onto the screen. Inside UpdateScreen(), DrawDisplay() and DrawMinimap() are also called, which handle the models, positions and shaders of the GameObjects.

   
*Figure 3: Code snippet of the UpdateScreen() function*

# Collision and Combat

To implement combat, collision detection is used to check whether any asteroid has collided with either the laser bullet or the spaceship. The GameLoop() will check for the size of the asteroids array and run the collision detection code for every asteroid.

  
*Figure 4: Code snippet of the game loop function*

For every asteroid, the game checks if it has collided with either a laser bullet or the player’s spaceship. Collision with a laser bullet destroys the asteroid but not the bullet, and collision with the spaceship destroys both, ending the game.

   
*Figure 5: Code snippet showing the collision detection function*

The SetActive() function is part of the GameObject class and if set to false, will destroy the GameObject to take it out of play.