

Games Programming 3 Aug Coursework report

David Muir

Matriculation Number: S1428976

Computer Games (Software Development)

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: David Muir

**Games Programming 3**

Introduction

**Aim of the Project**

The aim of this project was to create a 3D OpenGL based game using the C++ programming language. To meet the project specification, the game required several 3D models, textures, cameras, collision detection and Audio.These components were combined to create a basic playable game

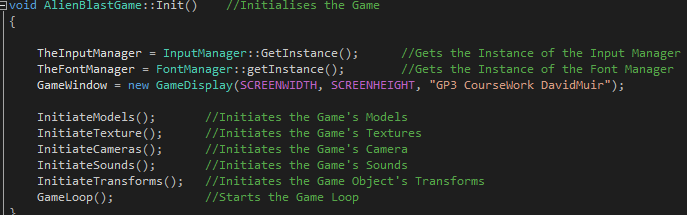
**Initial Design**

For this project, the author aimed to recreate the classic arcade game Space Invaders. In the game, the player must control their spaceship, moving it up and down the screen to defend against waves of alien ships. The player can fire a rocket to destroy incoming ships however if and enemy ship escapes the player then they lose a life. Once the player has lost all three lives, the game ends. While playing the game, the user can switch between game cameras to try playing the game from different perspectives.

**Implementation**

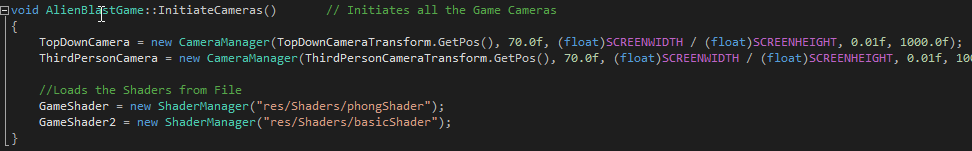
The game was implemented using Microsoft Visual studio and the C++ programming language.The game was developed as an OpenGL application. Several libraries were used throughout the development of the game including SLD2, Glew32, Ftgl and OpenAL. The libraries are used to implement fonts,audio,input and even the games display.

**Initialise**

****

When the game is booted up the first function that is called is the Init function in the AlienBlastGame class constructor. This function is responsible for the initial setup of the game. The function begins by creating instances of the Input and font manager and creating the game window that will display the game itself. The Models, textures and sounds that make up the game are then loaded from file in the Initiate Models , textures and sounds functions. The Models are loaded using the OBJ\_loader class and textures using the stb\_image class.

****

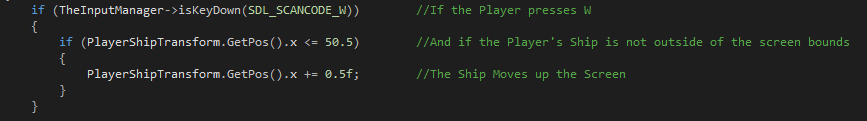
The game’s cameras are then created using the Initiate Cameras function. This creates two new cameras using the CameraManager class constructor, passing in the camera’s position, field of view and aspect information. For this game two camera’s were created, a top down camera and a third person camera. The shaders that are used for the game are also loaded from file.

The Initiate transforms function then moves all of the games objects to the correct positions around the scene, setting their size and rotation values.

Finally the Game Loop function is called, this function makes up the game itself and continues to run until the player either loses or closes the window.

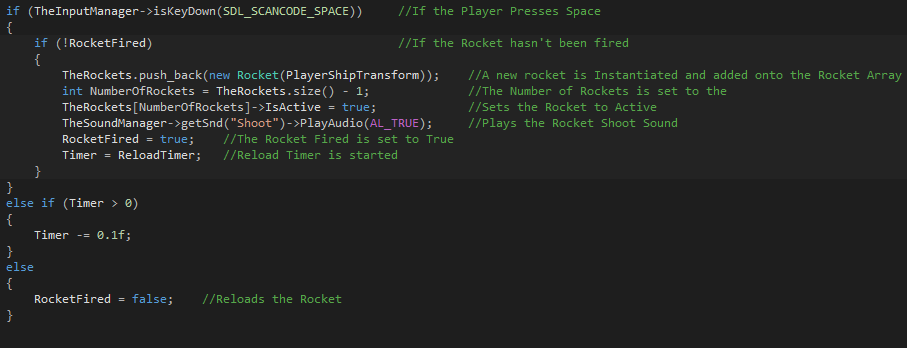
**Player Input**

For this game, the player takes control of a Space ship armed with rockets to defeat attacking aliens. The player is able to control the ship using several keyboard keys. The keys used to move the ship depend on the camera view that the player currently using since the camera can be switched between a top down and third person view. This was done to make the game less confusing to play. Player inputs are handled by the Input manager class, this is responsible for checking if a keyboard button is being pressed. The class makes use of the SDL library to track which keys have been pressed, specifically the isKeyDown function takes in an SDL\_Scancode. This is then used in the game loop to check if a key responsible for movement is currently being pressed.



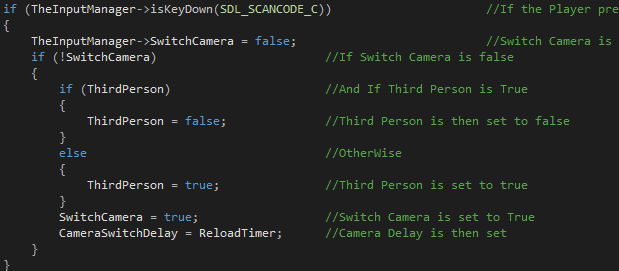
If a key is being pressed and the ship isn’t already at the screen’s boundary then the ship will begin to move across the screen.

The player’s ship has a rocket attack that can be used to destroy the attacking aliens. The player can trigger this attack by pressing the Space bar, Input detection for this is carried out in the same way as the ships movement. When the spacebar is pressed, if the player hasn’t recently fired the rocket, a new rocket is instantiated at the players ship and added to the rocket array. The number of rockets is then increased and the IsActive boolean is set to true so that it will be rendered and updated. The sound manager then plays the rocket shooting sound.Finally the Rocket fired boolean is set to true and Reload timer is started, these are in place to stop the player spamming rocket and ruining the game.

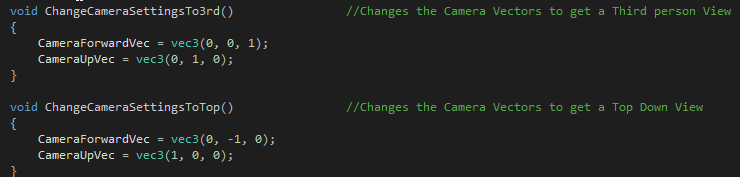
****

**Camera Switching**

Since the game has several cameras that the player can view the game from, a method for switching between them had to be implemented. The player is able to switch between cameras by pressing the “C” key. The Input manager detects that the C key was pressed and sets the Switch Camera boolean to false. The function then checks which camera mode is currently being used. For example, if the third person camera is being used and the C key is pressed then the Third person boolean will be set to false, switching the camera to the top down view. A timer is in place to stop the camera instantly switching back and forth.

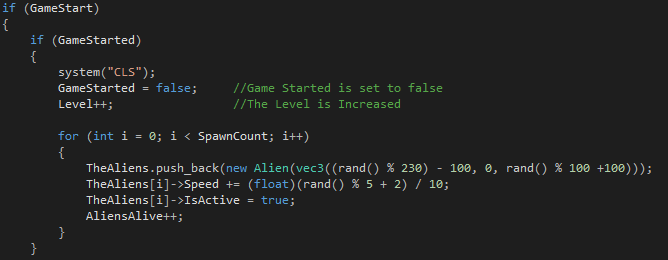


The change between cameras occurs in the Update and Render function of the AlienBlastGame.ccp. Each frame the game checks which camera is being used and if it detects a change then the Camera manager’s Change Camera setting functions are called for their respective camera. These functions are responsible for adjusting the camera’s vectors so that the camera is facing in the correct direction.

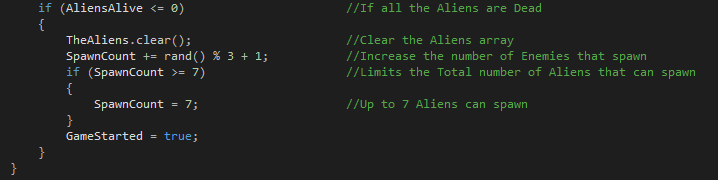
****

**Alien behaviour**

Throughout the game, the player must defend themselves against increasingly large waves of alien ships. These ships spawn at random points across the level from the player and quickly move towards the players ship. The player must destroy all of the ships attempting to get past or lose a life. If the player destroys all of the ships then a new round begins and the number of enemies is increased. The Alien.h class is responsible for the aliens position, speed, movement and for carrying out collision detection. The spawning of alien waves is carried out in the Update and Render function of AlienBlastGame.ccp.



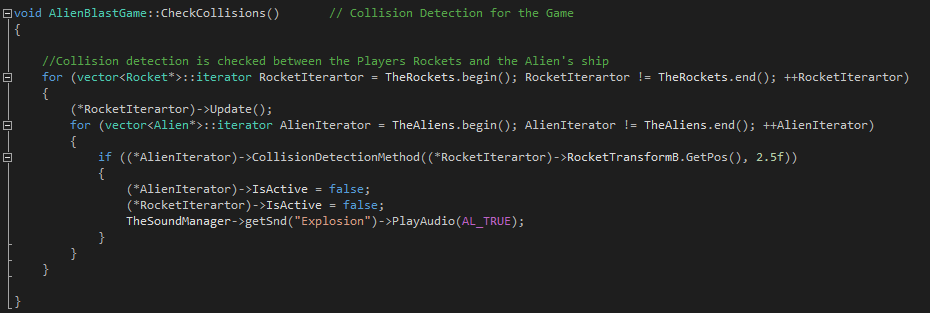
Once the player starts the game, the Game Started boolean is immediately set to false and the level counter is increased. This stops the game from looping and spawning a constant stream of waves. A FOR loop is then used to create each of the new aliens up to the current spawn limit. A new alien ship is then instantiated at a random location along the spawn axis and is placed into the alien array. The speed of the alien ship is also random so that all of the ships don’t arrive at one time overwhelming the player. Finally the IsActive boolean is set to true and the number of alive aliens is increased.



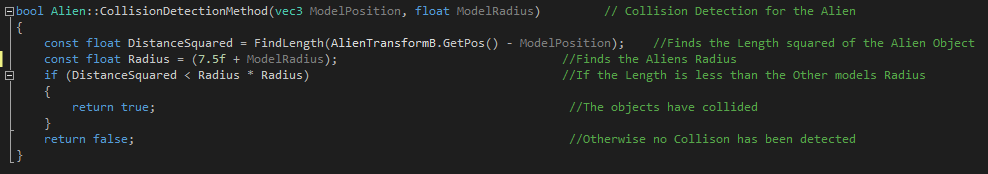
Once all of the aliens have been destroyed, the array is cleared and the number of enemies that can be spawned is increased to slowly increase the difficulty. The total number of alien is capped at 7 to avoid the game getting too difficult. Finally the GameStarted boolean is set to true which will spawn the next round of aliens.

**Collision Detection**

For this game, collision detection is carried out between the player’s rockets and the alien ships to check if the player has managed to hit the enemy. Collision detection has been implemented using the radius overlap collision method. A check for collisions between the rocket and an alien ships is carried every single frame. This occurs in the the CheckCollisions() function of the AlienBlastGame.cpp.



This function uses a nested FOR loop to check for collision between every active rocket in the scene and every alien in the scene. The function begins at the start of the array carrying the rocket objects and updates the first rockets position in the game. It then checks for collisions between the current rocket and the first object in the array holding the alien ships using the Collision detection function from Alien.ccp.



This method finds the distance between the alien and the rocket and then checks if the rocket is within a predetermined radius.

If the objects are colliding then the function returns true. If the rocket is colliding with the alien ships then the IsActive boolean for each object will be set to false. This will remove them from the game. The Sound manager will also play the explosion sound. On the other hand if rocket is not colliding with an alien then it will move onto the next object in the array and cycle through all of the active aliens. Once it has either reached the end of the array or hit a target, the For loop will carry out the process again for the rest of the rockets.

**Resources**

**Bibliography**

Models:

Player Ship Model: https://www.turbosquid.com/FullPreview/Index.cfm/ID/611566

Alien Ship Model: https://www.turbosquid.com/FullPreview/Index.cfm/ID/612994

Rocket Model: https://www.turbosquid.com/FullPreview/Index.cfm/ID/309350

Planet Model: https://www.turbosquid.com/FullPreview/Index.cfm/ID/553741

Sun Model: https://www.turbosquid.com/FullPreview/Index.cfm/ID/309350

Textures:

Player Ship Texture: https://www.turbosquid.com/FullPreview/Index.cfm/ID/612994

Alien Ship Texture: https://www.turbosquid.com/FullPreview/Index.cfm/ID/612994

Rocket Texture: https://www.turbosquid.com/FullPreview/Index.cfm/ID/612994

Planet Texture: http://2.bp.blogspot.com/\_t3CsQGWP8IE/TCRLREFvD6I/AAAAAAAAAF8/r9825Gd2Tkg/s400/01.jpg

Sun Texture: <https://www.brusheezy.com/textures/49324-surface-of-the-sun-texture>

Audio:

Shoot: https://freesound.org/people/smcameron/sounds/51468/

Explosion: https://freesound.org/people/sharesynth/sounds/344507/

Main Music: https://trello-attachments.s3.amazonaws.com/57eaaaf32fef37c9b200bf65/580f89c46f23b94488a51ef9/d0de4e3379f442cfcefb0cf98e613e1e/Battlecats\_menu.mp3

Libraries:

SDL: https://www.libsdl.org

GLEW: http://glew.sourceforge.net/

GLM: https://glm.g-truc.net/0.9.4/api/index.html

FTGL: http://ftgl.sourceforge.net/docs/html/index.html

OpenAL: https://www.openal.org

External Files:

Object Loader: <https://github.com/BennyQBD/ModernOpenGLTutorial/blob/master/obj_loader.cpp>

Image Loader: <https://github.com/nothings/stb/blob/master/stb_image.h>

Full code documentation: <https://blackboard.gcal.ac.uk/webapps/lobj-campuspack-bb_bb60/launch/setupForm.form?course_id=_84871_1&contentId=_2236119_1>