

CGP600 – ASSIGNMENT 2

KANAGA MANIKANDAN SOLAIKANNAN – Q12389501



January 5, 2018

SOUTHAMPTON SOLENT UNIVERSITY

Contents

[Core Features: 1](#_Toc502739991)

[Collision Detections: 1](#_Toc502739992)

[Scene: 3](#_Toc502739993)

[Multiple Cameras: 5](#_Toc502739994)

[Entity AI: 7](#_Toc502739995)

[Lighting: 10](#_Toc502739996)

[Development Process: 12](#_Toc502739997)

[Implementation Changes: 14](#_Toc502739998)

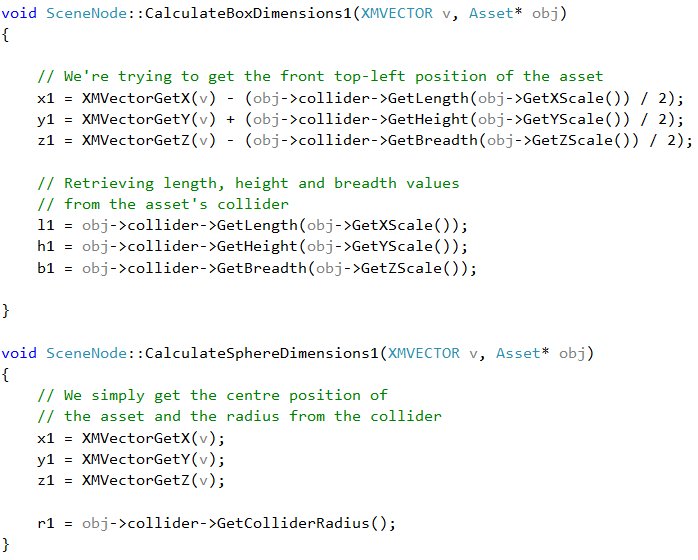
[Testing: 14](#_Toc502739999)

# Core Features:

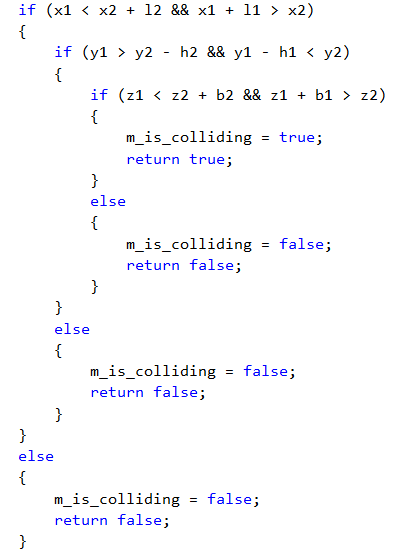
There are various features used in the project and each will be explained briefly.

## Collision Detections:

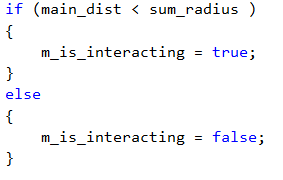
The game comprises of both, box, and sphere collisions. Box collisions are used for objects to collide and halt their movement. Sphere collisions are used for combat because the weapon’s scale values prevented from producing reasonable results when done with it box collisions. The equations are displayed in Code Snippets 1, 2, and 3. The implementation can be seen in Screenshot 1.



Code Snippet 1. Calculating values for respective collisions



Code Snippet 2. Box Collision Checks



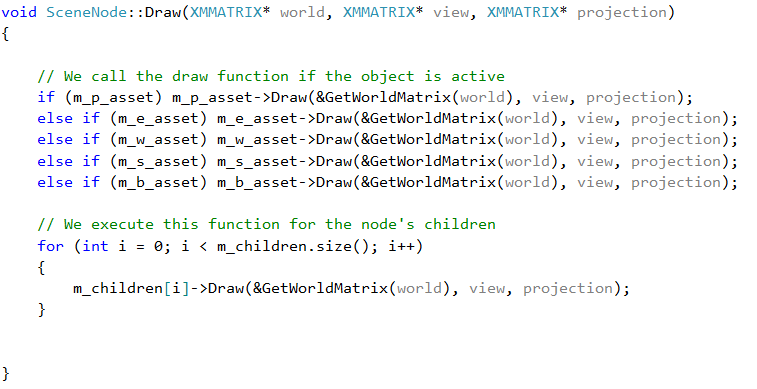
Code Snippet 3. Sphere Collision Check



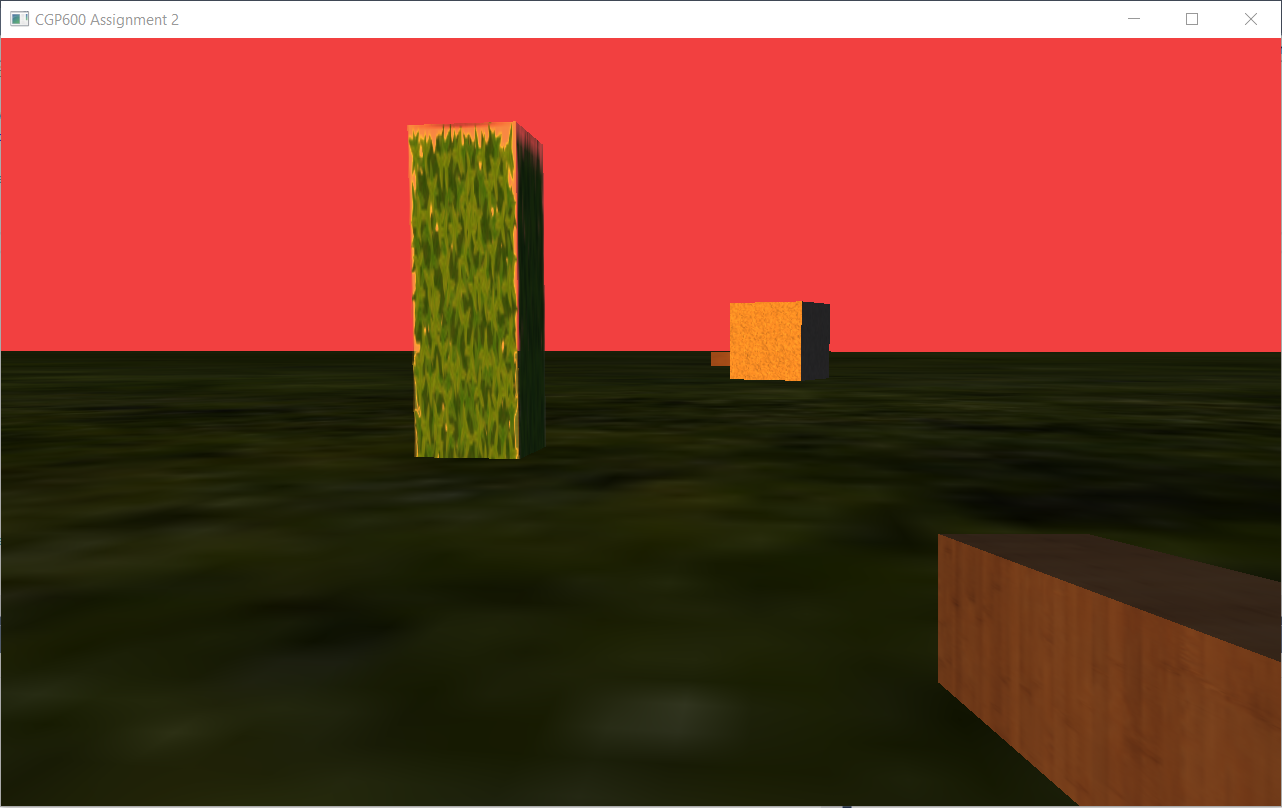
Screenshot 1. Player colliding with a Tree

## Scene:

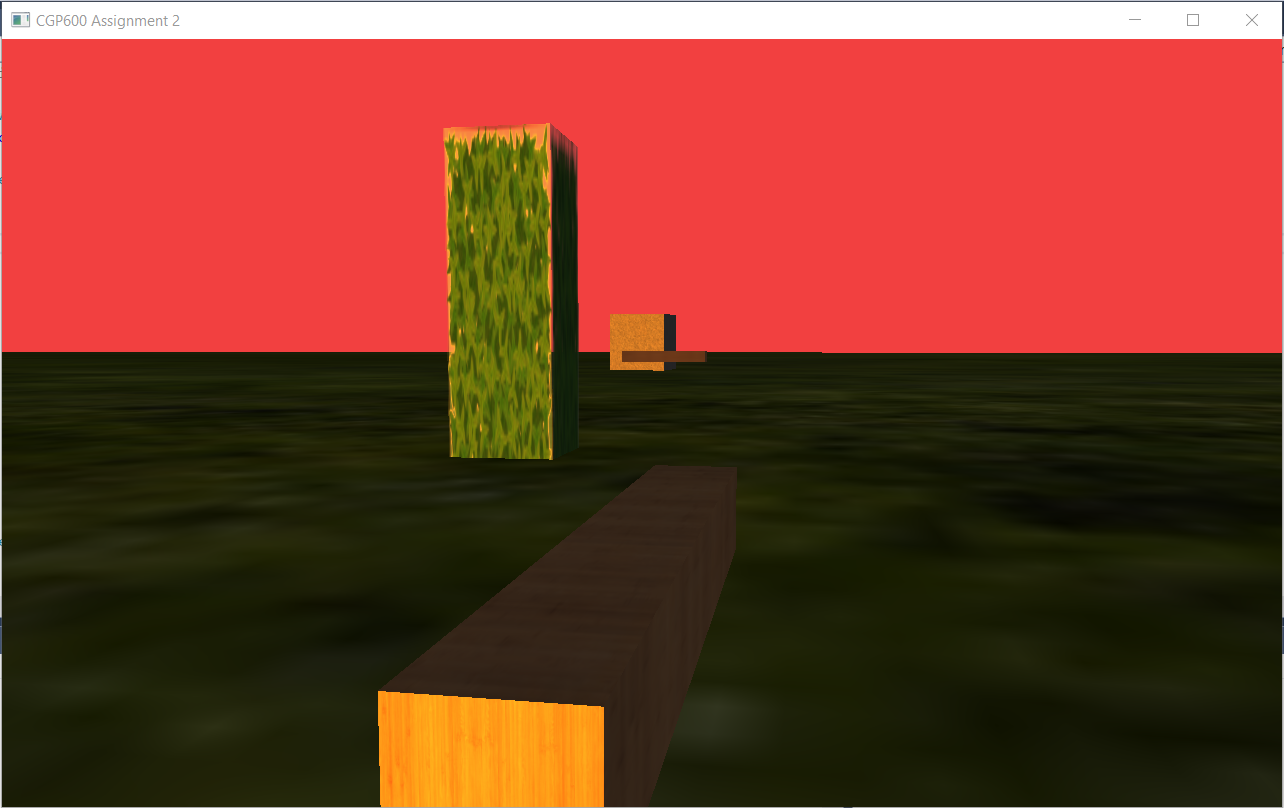
Every object in the game is a child of one abstract root node. Drawing of all objects is performed by calling the draw function of the root node and it will continue to call its children’s draw functions as well (see Code Snippet 4). The player can also equip and drop weapons which translates to adding (pushing back to a children vector) and detaching as a child of the player (erasing from the children vector). The same logic is applied for the moving block as well. The implementation can be seen in Screenshots 2 and 3.



Code Snippet 4. Drawing of Scene



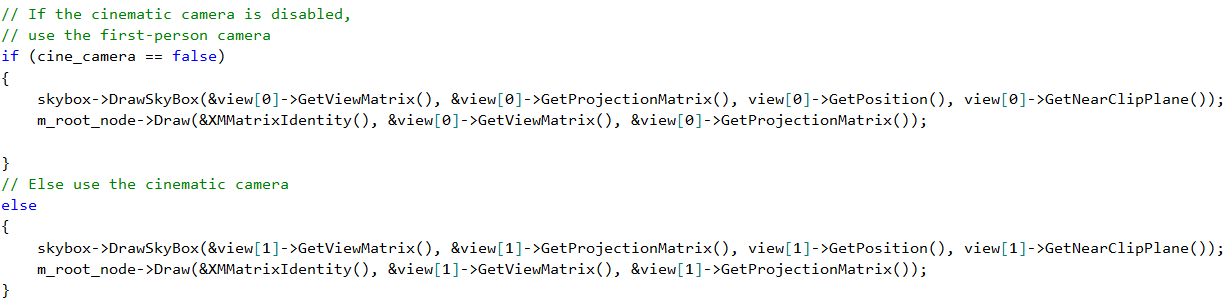
Screenshot 2. Weapon added as a child



Screenshot 3. Weapon detached from the player

## Multiple Cameras:

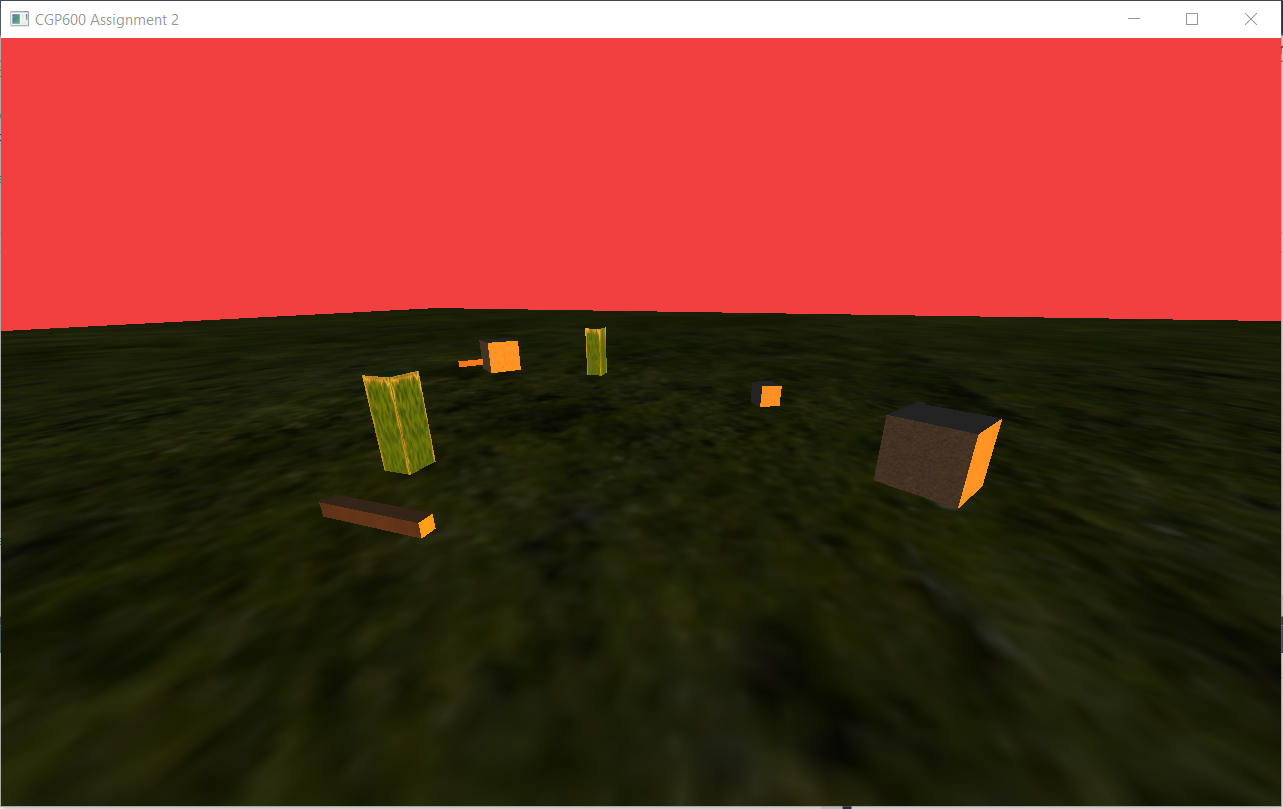
The game consists of 2 cameras. First, is the first-person camera from which the player can view and play the game. Second, is a cinematic camera which pauses the entire game and enables the user to freely move around the environment and obtain a better view of it. The player can switch between cameras with a press of a button (see Code Snippet 5). Both the cameras can be seen in Screenshots 4 and 5.



Code Snippet 5. The respective camera will be used



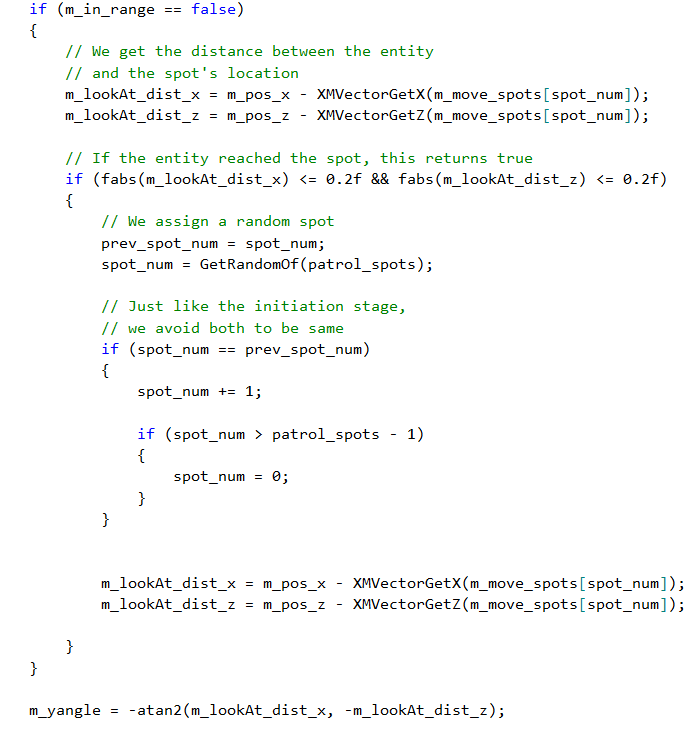
Screenshot 4. First-Person Camera



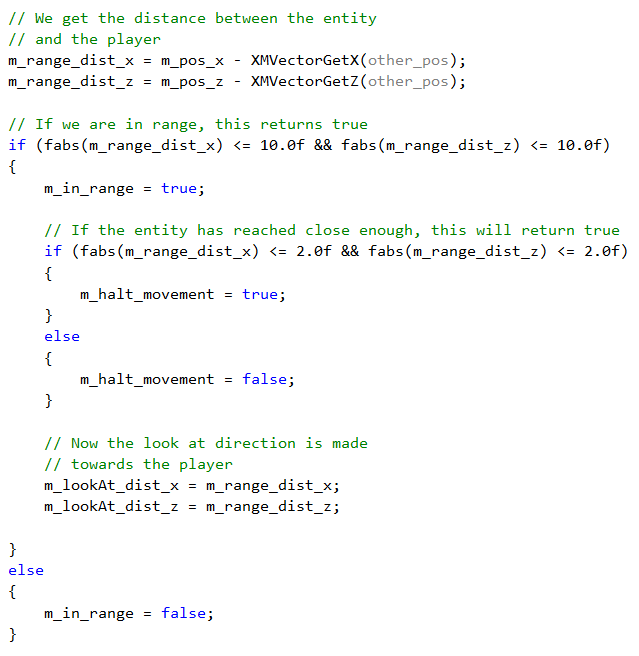
Screenshot 5. Cinematic Camera

## Entity AI:

The only entity in the game are the enemies. They have a fixed number of spots to move towards and when they reach a spot, they move to another randomly chosen spot (see Screenshot 6). As the player approaches an enemy, they abandon their patrol and start moving towards the player (see Screenshot 7). Once, the entity is close enough to the player, they start attacking the player (see Screenshot 8). If the player moves a short distance, they still stop attacking and start to move closer to attack again. If the player flees, they resume their usual patrol. The logic can be seen in Code Snippets 6, 7, and 8.



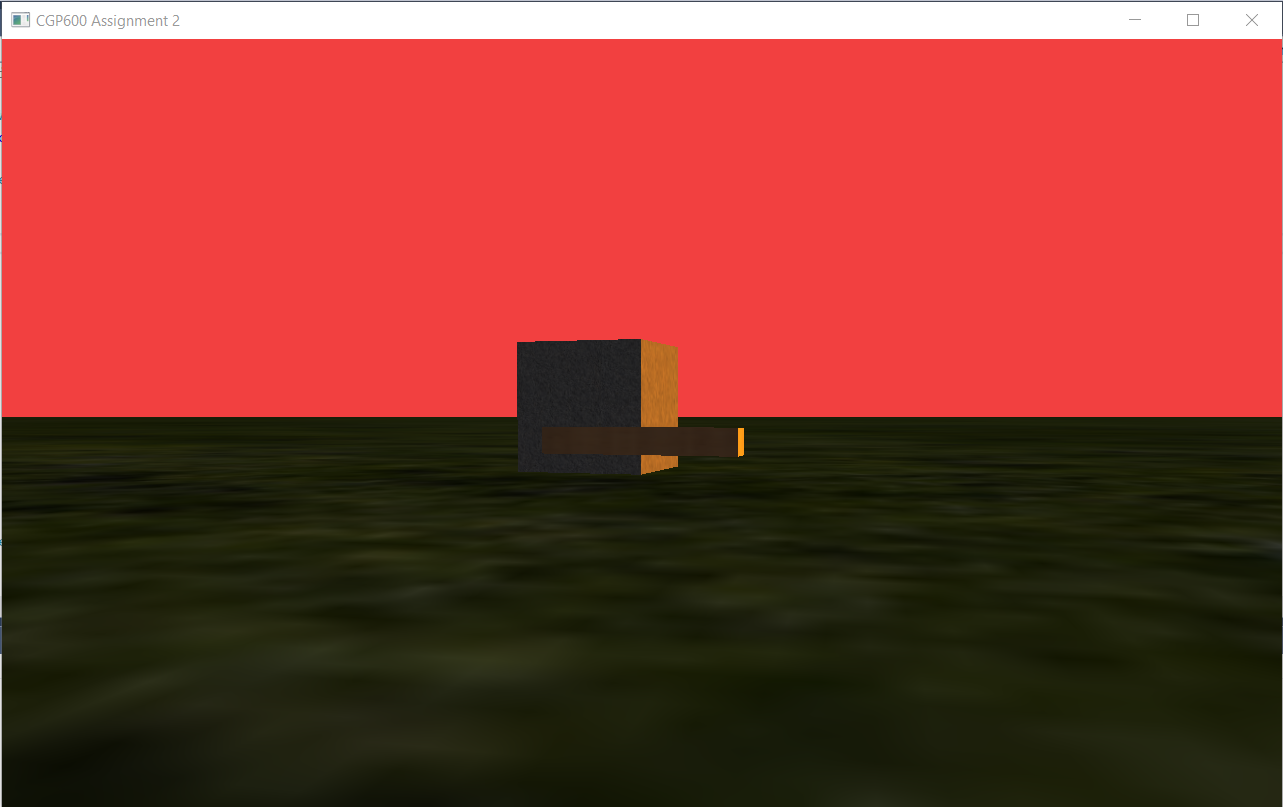
Code Snippet 6. Enemy Patrol Logic



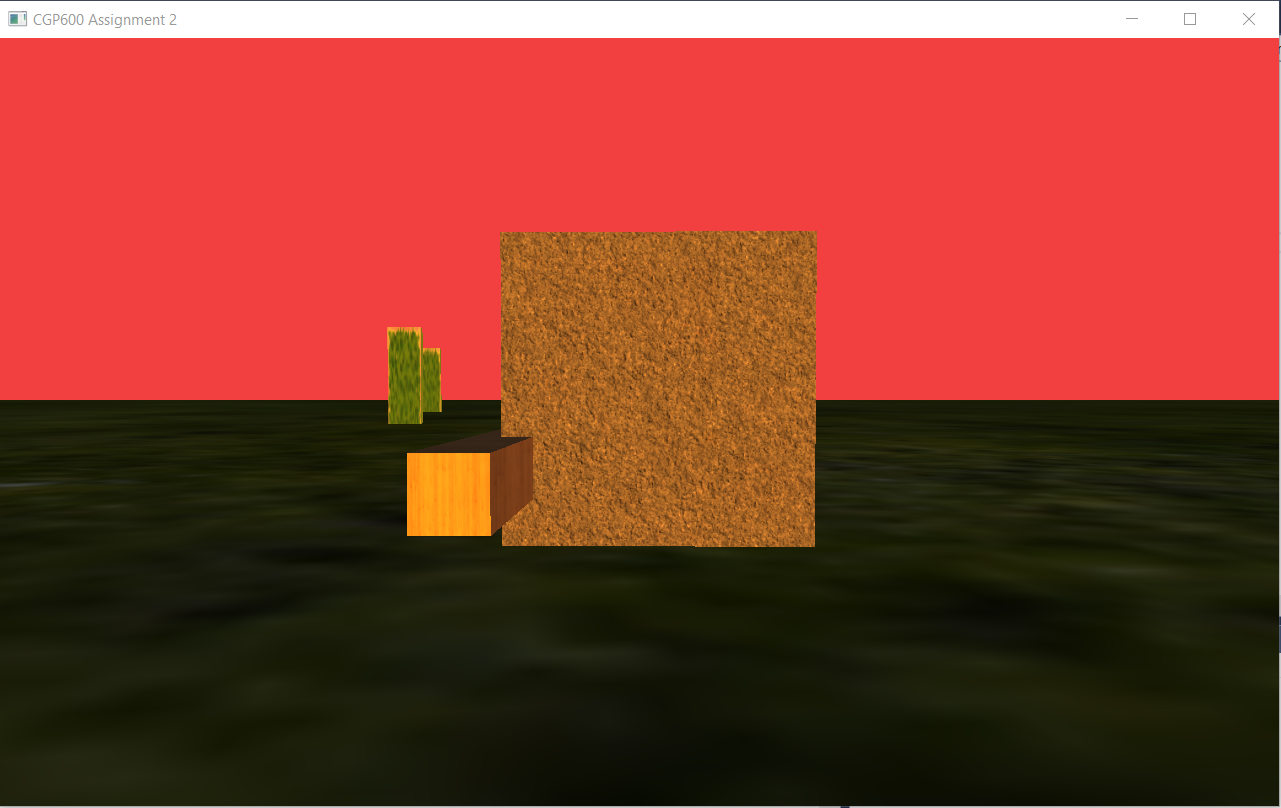
Code Snippet 7. Enemy Approach Logic



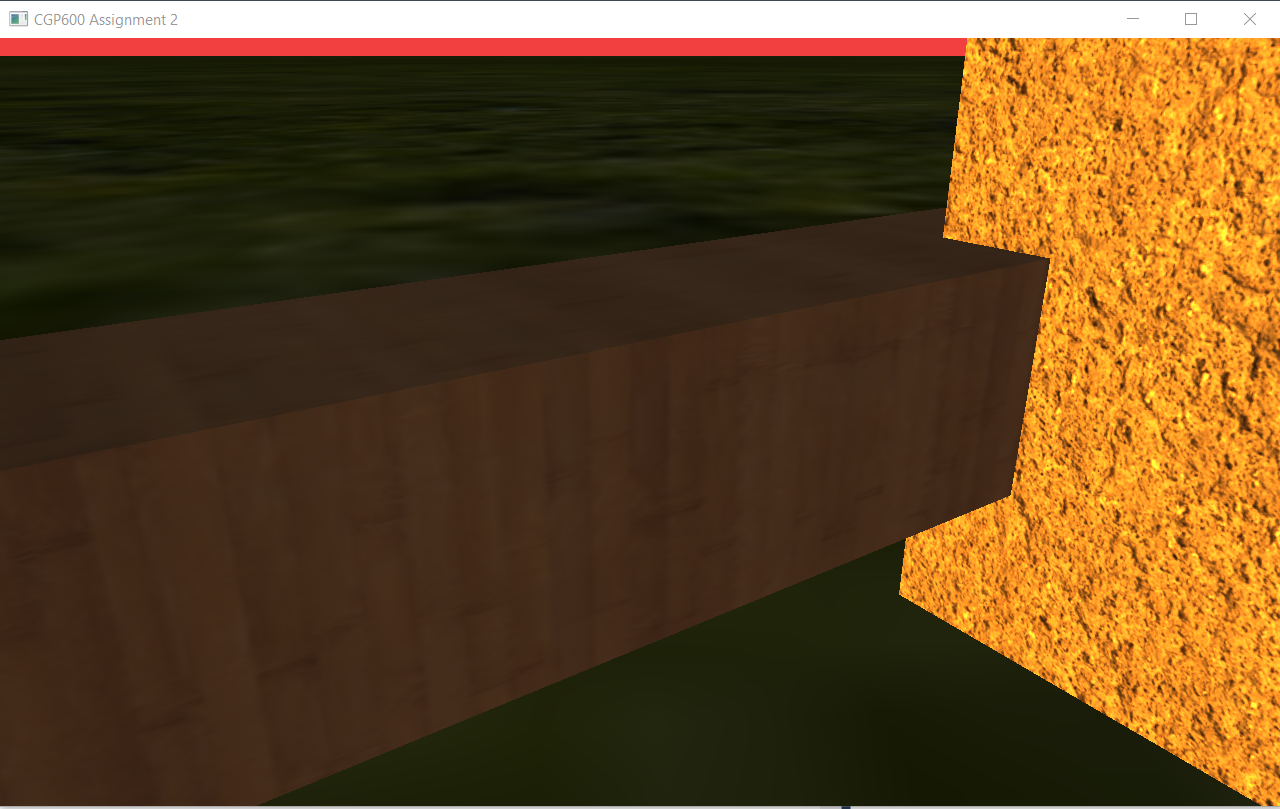
Code Snippet 8. Enemy Attack Logic



Screenshot 6. Enemy Patrolling



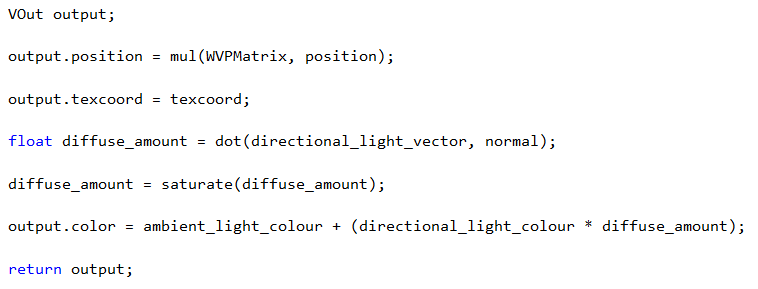
Screenshot 7. Enemy Approaching



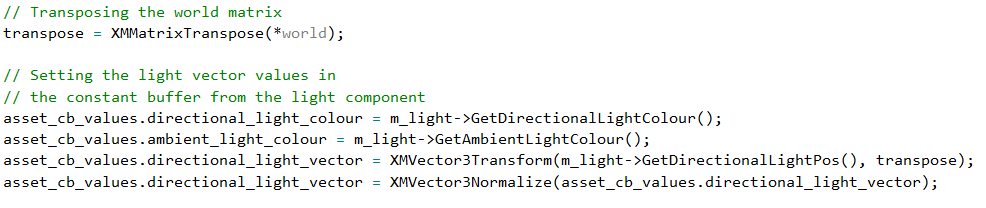
Screenshot 8. Enemy Attacking

## Lighting:

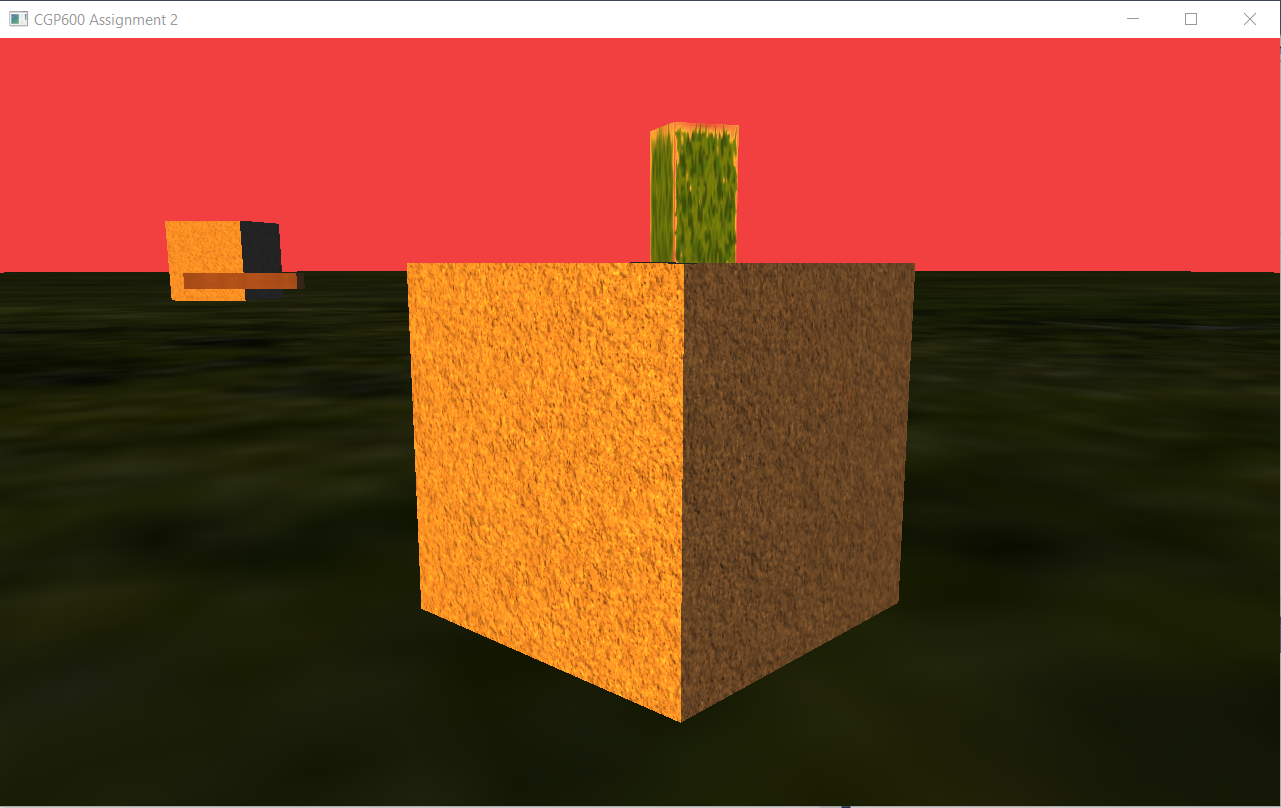
The game consists of a single directional light. It is made to be orange in colour to represent sun at sunset. It is casted diagonally from the left. The implementation can be seen in Screenshots 9 and 10. The logic can be seen in Code Snippets 9 and 10.



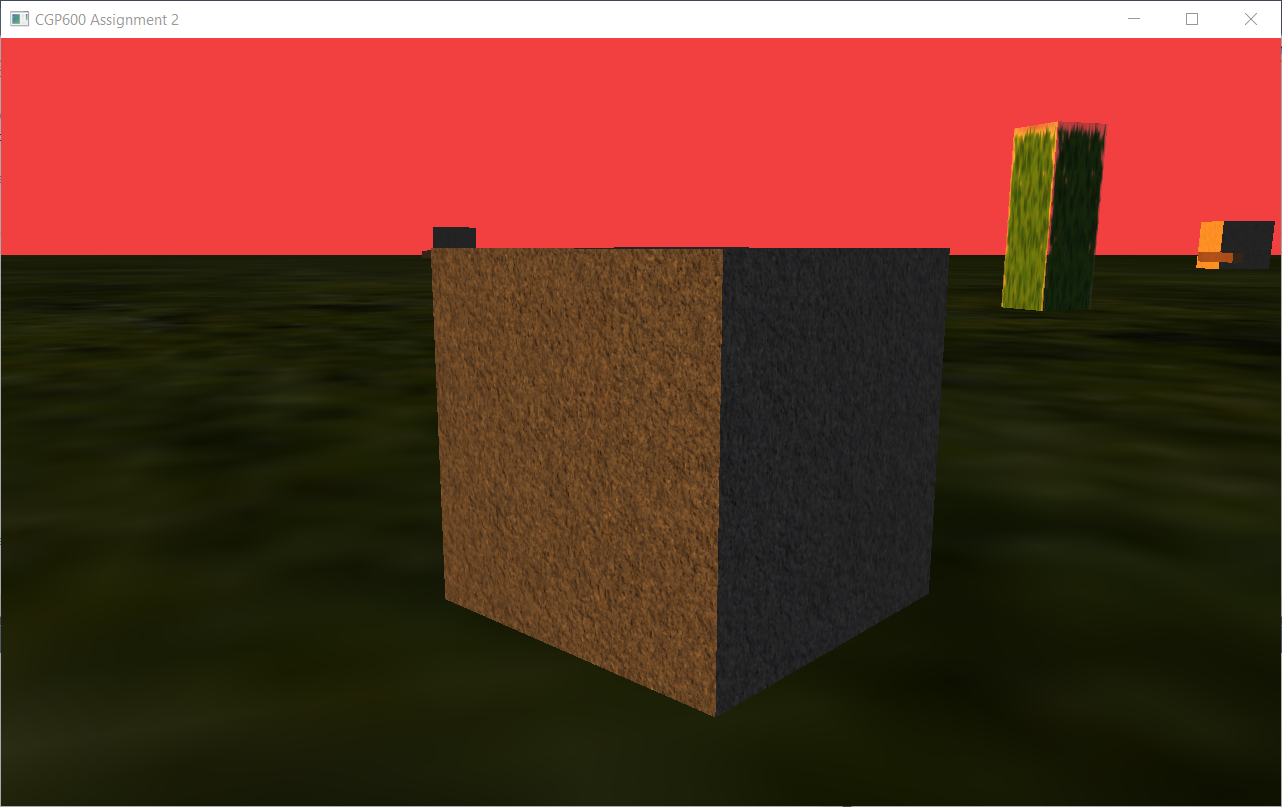
Code Snippet 9. Lighting code in asset shader



Code Snippet 10. Passing in Light info to the constant buffer



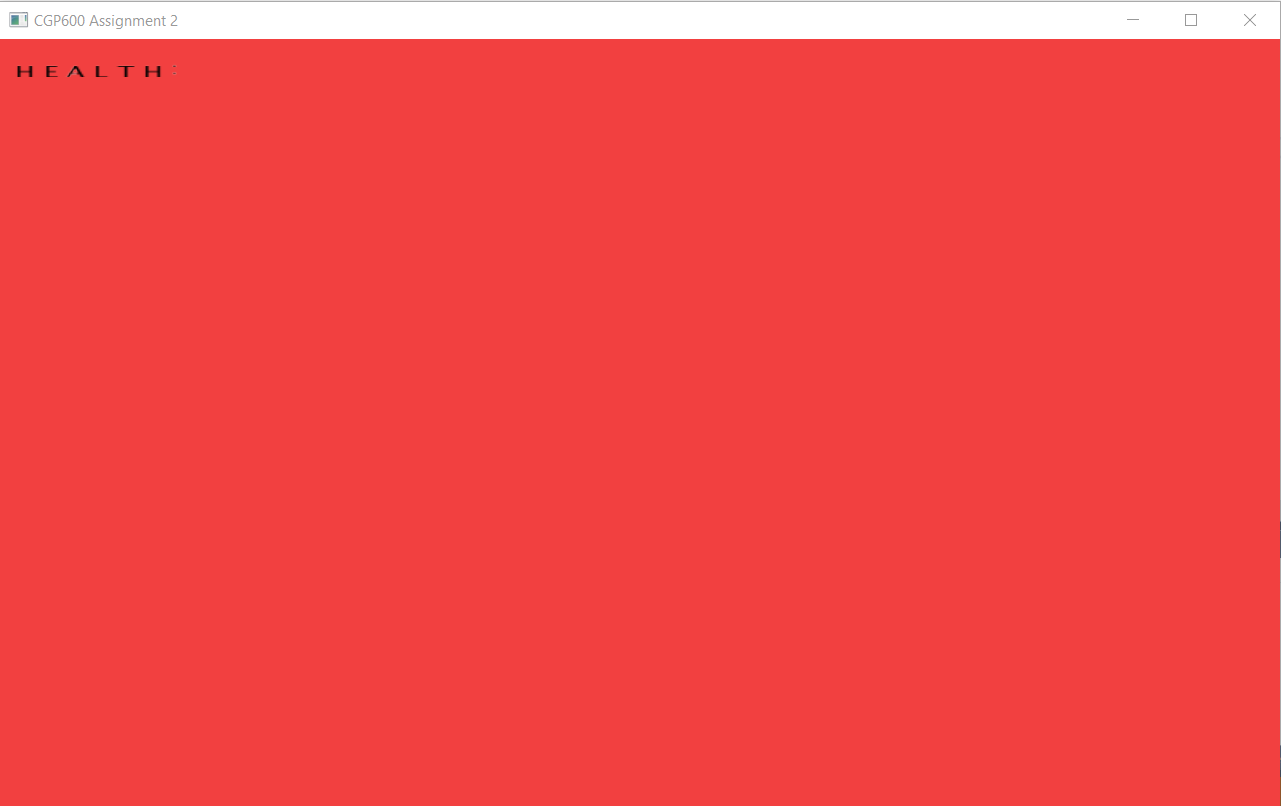
Screenshot 9. Full and partial lighting



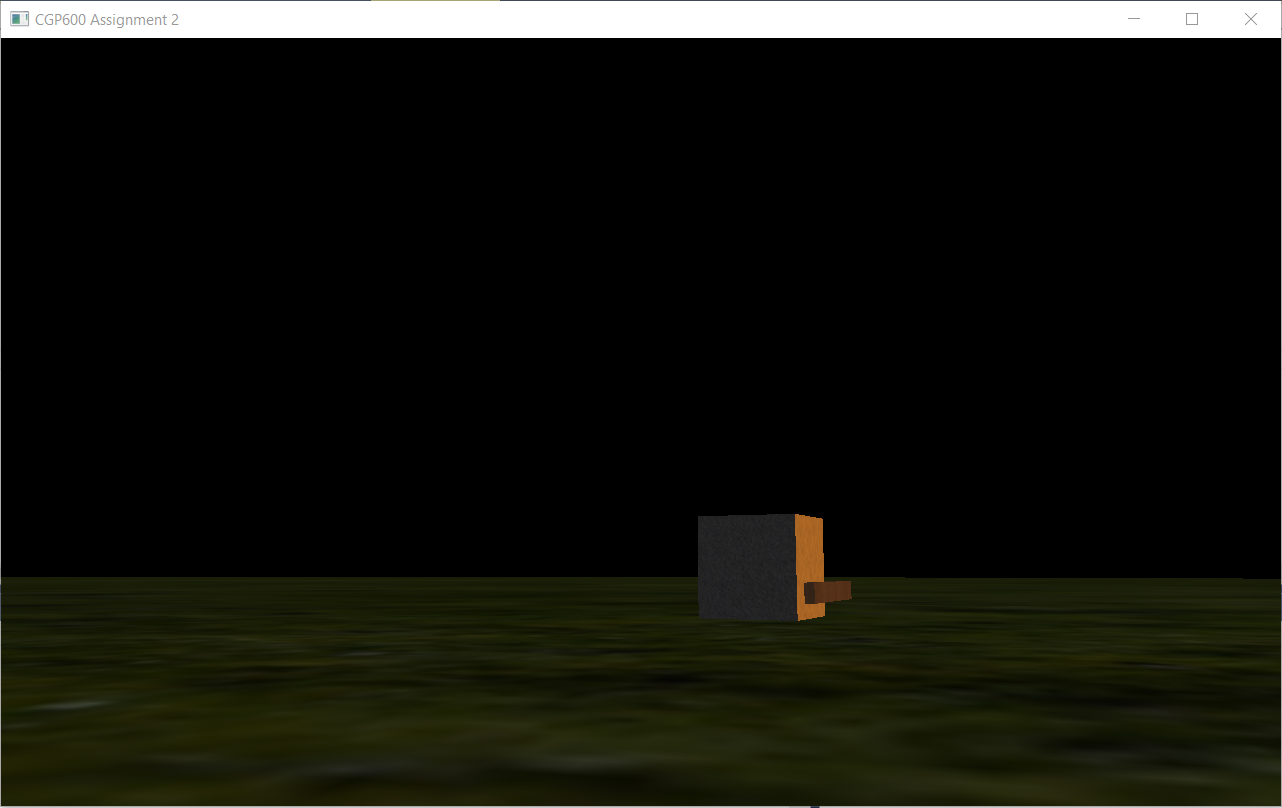
Screenshot 10. Partial and no lighting

# Development Process:

Each concept and technique of DirectX 11 was first developed in separate projects to test them individually. This enabled to understand each concept in clear detail and have a better understanding. Then, a new project was created to put together all the concepts and develop the game at whole. The code was clearly split into various classes and all the relevant details were then declared, initialised, and used appropriately. The newer concepts were also developed separately to ensure it doesn’t disturb the main project. Some of the new concepts were implemented but failed to work. Some of these are UI which had overwritten all other models and textures when rendered (see Screenshot 11) and a skybox texture which failed to load in a texture (see Screenshot 12).



Screenshot 11. The game screen with UI but everything else overwritten



Screenshot 12. The black portion is the skybox but with no texture

# Implementation Changes:

The game was designed to be in third-person, but the game was implemented in first-person. The reason for this change is because the implementation process for third-person proved to be far more complex than the first-person camera. The axis needed to be relative to the player for the camera to rotate around the player and it required a lot of processes. So, the idea of third-person camera was discarded. Moreover, the game was better viewed and played in first-person and provided a lot of ease for the developer to test and play the game.

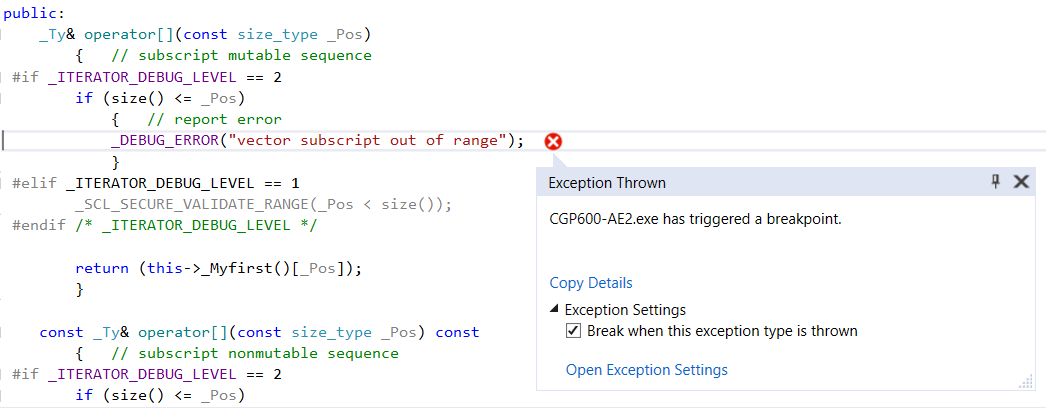
The game was also designed to have fruits which increases the player’s health. It could have been easily implemented but with the issue of not having UI in the game, the developer was unable to display the player’s health, so the relevant changes could not be displayed. Because of this issue, an elapsed timer, main menu and leader boards from the design couldn’t be implemented as well.

# Testing:

The testing plans and results undergone during the project will be displayed in this section.

Table 1. Test Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Task** | **Expected Result** | **Actual Result** | **Status (Pass/Fail)** | **Notes** |
| 1 | Creation of window | The window should be created and be blank. | The window was created and was blank | Pass | - |
| 2 | Rendering a background colour | The program should render a solid colour | The program rendered a solid colour | Pass | - |
| 3 | Rendering an object | An object should be rendered on screen | The program crashed on run (see Code Snippet 11) | Fail | The model used had irregular information on vertices, normal and UV co-ordinates. So, a cube was used. |
| 4 | Direct Input for Keyboard | The keyboard should be initialised and be used for input | The keyboard was initialised, and the program received input | Pass | - |
| 5 | Direct Input for Mouse | The mouse should be initialised and be used for input | The mouse was initialised, and the program received input | Pass | - |
| 6 | Camera Movement (Yaw) | The camera should rotate in yaw axis when the mouse moves sideways | The camera rotated correctly in yaw axis | Pass | - |
| 7 | Camera Movement (Pitch) | The camera should rotate in pitch axis when the move moves vertically | The camera started oscillating when the mouse kept moving up/down | Fail | The yaw and pitch were sharing the same ‘Z’ value used for ‘look at’ vector. It was fixed by restricting the pitch within +90 and -90 degrees. |
| 8 | Object Movement (Z-axis) | The object should move forward and backward in z-axis | The object moved forward and backward | Pass | - |
| 9 | Object Movement (X-axis) | The object should strafe sideways | The object strafed sideways | Pass | - |
| 10 | Object Movement (Forward in facing direction) | The object should move forward in the direction it is facing | The object moved forward in the direction it is looking | Pass | - |
| 11 | Object Movement (Strafe movement in facing direction) | The object should strafe in the direction it is facing | The object strafed but not always | Fail | The equation and logic were flawed. It was fixed by using the same principle for forward direction except, we add 90 degrees to move sideways |
| 12 | Object Jump and Fall | The object should jump and fall | The object jumped and fell correctly but movement speed was reduced when on ground | Fail | The check for the object on ground caused it to move up and down constantly. It was fixed by having a different collision check against the ground rather than directly colliding with it. |
| 13 | Retrieval of Asset’s Scale and Radius | The program should correctly calculate the length, height, breadth, and radius of the asset for collision detections | The length, breadth, and height were calculated correctly but the radius wasn’t | Fail | The distance between the surface point and the centre point was calculated individually in each axis thus resulting in irregular values. This was fixed by calculating the distance on all axes and using Pythagoras theorem for the actual distance. |
| 14 | Object Collision | The player object should collide with other objects and stop. | The object collided but movement across the other object didn’t take place | Fail | The object’s position was restricted and updated right away. It was fixed by updating only after all restriction process is complete. |
| 15 | Camera Movement | The camera should move along the player | The camera moved up when the player looked up | Fail | The camera was made to move at the same time the player moved to mimic the player’s movement. It was fixed by passing in the player’s position to the camera’s position |
| 16 | Scene Hierarchy (Draw) | All the objects should be drawn by calling one parent root node | All the objects are drawn by calling the parent root node | Pass | - |
| 17 | Equipping a weapon | The player should get close to a weapon and pick it up | The player equipped the weapon | Pass | - |
| 18 | Moving the weapon | The weapon should move forward then backward appropriately for combat | The weapon moved irregularly | Fail | The facing angle was included to move the weapon which wasn’t required. It was fixed by removing it. |
| 19 | Combat Collision | The weapon should hit and damage the enemy. | The collision did not take place | Fail | The object’s asset position values were used for collision. It was fixed by creating position variables for the nodes and using them |
|  |  |  | Collision takes place but it’s not consistent in all directions | Fail | Box collision detection was used for combat collisions. It was fixed by having sphere collision detections so it’s uniform in all directions. |
| 20 | Gravity | Besides the player, gravity should affect all objects. | All objects are affected by gravity, but sometimes caused them to go below the ground | Fail | The object was stopped at a minimum distance from the ground. The distance was too short for valid checks. It was fixed by slightly increasing the distance |
| 21 | Carrying blocks | The player should be able to carry the block | The player carried the block but halted the player’s movement on drop | Fail | The offset distance on drop was too close to the player which caused to return true for collision |
| 22 | Enemy Patrol | The enemy should move towards pre-defined spots | The enemy moved appropriately | Pass | - |
| 23 | Enemy on Collision | The enemy should move towards the previous spot on collision | The enemy passed through an object on start-up but worked appropriately the second time | Fail | The issue was that in the initialisation, the previous and current spots were the same causing the enemy to continue its path on collision. It was fixed by initialising different numbers |
| 24 | Enemy to look at Player | The enemy should look at the player, when it gets close | The enemy looked correctly | Pass | - |
| 25 | Enemy Combat | The enemy should start attacking the player if it gets close | The enemy attacked the player | Pass | - |
| 26 | Enemy Death | The enemy should disappear on death | The enemy disappeared on death | Pass | - |
| 27 | Player Death | The player should reset back to its starting position | The player’s position but the camera’s position wasn’t | Fail | Wrong position value was reset. It was fixed by resetting the correct value. |
| 28 | Directional Light | The light should be casted from the left and be orange in colour. | The light was casted and is orange in colour | Pass | - |
| 29 | Ambient Light | The ambient colour should light up the whole world | The ambient colour lit up correctly | Pass | - |
| 30 | Skybox | The skybox should be visible with all other objects rendered in front of it | The skybox was rendered but had no texture on it (see Screenshot 12) | Fail | The shader files seemed to be sharing the texture info in some way and no valid fix was found for this. |
| 31 | Text | The text should be rendered over the screen | The text was rendered but everything else wasn’t (see Screenshot 11) | Fail | Cause and solution weren’t found |
| 32 | Texture on Objects | The texture should be applied to objects | The texture applied appropriately | Pass | - |
| 33 | Transparent Texture on Objects | Transparent texture should be applied and should be able to see through it | The transparent texture was applied and can be seen through it | Pass | - |
| 34 | Delta Time | The program should calculate delta time | The program calculated delta time correctly | Pass | - |
| 35 | Delta Time with speed | Delta time should affect the speed of moving objects | It affected correctly but the resulting speed was very high | Fail | The delta time is in large numbers on start up which caused the objects to move in great speeds. This was fixed by passing in delta time only when it’s less than 1. |



Code Snippet 11. Program Crash on loading an External Model

# Self-Reflection:

This assignment helped me understand the DirectX 11 API well enough. It also helped me learn how each component works in a game engine such as lighting and skybox. The functions and logic of the DirectX 11 libraries were also straight forward and easy to grasp and implement. There was a level of satisfaction when implementing each mechanic and that wasn’t attained with other SDKs.

Some of the features from the design couldn’t be implemented, unfortunately. One of the reasons can be the problems encountered during this assignment. Taking UI for this case, I had debugged a lot of times and looked through the code thoroughly to see where the problem lied. I couldn’t locate the cause and only time was wasted. The SDK did not possess a font rendering engine, so we had to upload an image with all the letters, numbers and symbols included and the program would just crop the appropriate character from the whole image and render it. Other problems encountered were with the game mechanics such as gravity, directional movement, collision problems and such. These are explained briefly in Table 1. The fixes were hidden in plain sight and I consumed a lot of time in finding them.

If I had to improve this project, I would have a different mechanism to draw all the assets. Now, I load in cubes for each asset and draw them all which caused frame rate drops at times. Instead, I would load the cube once and use it as an instance for all objects. This can improve the performance and keep the FPS consistent. Other ways of improving would be by implementing a point source of light for bonfire as this game takes place in an island.