|  |
| --- |
|  |
| Progressive Games Engine Assignment |
| Golf Game |

|  |
| --- |
| Eimantas Stoskus  P2672054 |

Contents

[Design 2](#_Toc133206103)

[Game Idea 2](#_Toc133206104)

[Narrative 2](#_Toc133206105)

[Implementation 2](#_Toc133206106)

[Golf ball 2](#_Toc133206107)

[Level Design 3](#_Toc133206108)

[User Interface 3](#_Toc133206109)

[Level Trigger 4](#_Toc133206110)

# Design

## Game Idea

My plan for the project is to create a simple golf game using Unreal’s physics. For the advanced techniques, I will incorporate physics and collision techniques, user interface and HUD, and multiple levels and level design. The idea is to add an impulse to the ball towards the location the camera is facing and to use mouse movement to determine the strength of the impulse applied to the golf ball.

## Narrative

The narrative of the game is that you are a fan of mini golf, and you found an interesting course that you wanted to try and beat. You deicide to find out where this gold course is and travel to it.

# Implementation

## Golf ball

Graphical user interface, application

Description automatically generatedI began by creating an interactable golf that had a camera orbiting around it. For this, I used to attach a camera to a spring which was parented to the golf ball. I then made the camera move around based on mouse input. Once the camera movement was done, I proceded to creating a basic level, which consisted of a plane, for the golf ball to spawn in.

Graphical user interface

Description automatically generatedI then began implementing the physics of the ball being hit, which was one of the biggest hurdles of the project. I first tried to calculate the impulse vector from the camera’s position, which yielded no result. My second attempt calculated the direction from the camera towards the ball and applied a set amount of force to the ball. This attempt was close to working but had a flaw applying force in a downwards direction if the camera was placed above the ball, making the ball jump. This was fixed by breaking up the vector and setting the Z axis to be consstantly zero. This solved the problem of the ball bouncing into the air if the camera was above it, but created an inconsistency in strength later on as the golf ball would have a weaker impulse the higher the camera was placed above the golf ball.

A screenshot of a video game

Description automatically generatedGraphical user interface

Description automatically generated

## Level Design

A picture containing text, monitor, electronics, indoor

Description automatically generatedOnce the golf ball was functioning, I began trying to implement an ending to the level. The first step was to create a hole for the ball to fall into. My first attempt was to create an illusion of a hole by having a cylinder with inverted normals placed into the ground and changing the which collider the golf ball interacts with then it is close to the hole. This worked method worked very well initially by creating an illusion of a ball inside the static mesh. Unfortunately, when it came to creating a trigger for completing the level, the trigger would not work inside the hole, which I believe was due to the golf ball interacting with the plane that it was passing through and not triggering the overlap event inside the hole.

For my second attempt, I used BSP levels to create a level and add a hole using the cylinder subtractive brush. This method worked almost flawlesly, allowing me to design a level with relative ease using the brushes whilst at the same time allowing me to create a static mesh that already included a hole in it. Unfortunately, for reasons unknown to me, when building the final static mesh using BSP geometry, the ball would not interact with it and would fall through the level.

For my third attempt, I decided to rebuild the level using BSP geometry and not build the static mesh. Instead I created the level and assigned materials to the surfaces to give the level colour. This allowed me to design the level as I liked without losing the ability to interact with said level.

Graphical user interface, table, timeline, Excel

Description automatically generated

## User Interface

Once I had a playable level to test everything, I decided to start work on the UI and menu. I began by arranging all the elements for all the menus and UI that I would need. I kept all UI simple at first so that I could focus on the functionality of everything and add any elements I would need as I went along.

A picture containing treemap chart

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenceA picture containing text

Description automatically generated

A screenshot of a computer

Description automatically generated with low confidenceI then began on implementing the main menu functionality, such as loading the level, opening the options menu, changing resolution and exiting the game.

A picture containing text, indoor

Description automatically generatedA group of electronic devices

Description automatically generated with low confidenceOnce the menu was working, I implemented the HUD elements. I began by having the HUD show if the ball is moving by checking a boolean from the golf ball blueprint. It then enabled and disabled a pop up based on that boolean.

A screenshot of a video game

Description automatically generated with medium confidenceI then began working on the timer, by incrementing the the seconds up to 59 seconds, which would then be set to 0 and increment the minutes by 1. I also made the timer pause when the level is completed and made the HUD blueprint show the HUD widget when the menu is not loaded. Lastly I made the Level complete button take you to the main menu once the button is clicked.

## Level Trigger

Once all the UI elements were completed, I created a trigger actor that would be placed in the hole to show the level complete screen, pause the timer and unlock the cursor. First, I made the Level Complete screen show up in the viewport. Once that was tested I realised that the mouse needs to be unlocked and the camera needs to be frozen, which were easy to implement. Lastly I needed to pause the timer. This required me to cast the HUD and level complete blueprints to the golfball to access the Pause Timer boolean. This was because I was unable to cast directly to thed HUD.

A screenshot of a video game

Description automatically generated with medium confidence

## Respawn

Graphical user interface, application

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidenceFinally, I implemented a respawn system for the golf ball in case it gets out of bounds. I began by creating a transform variable in the golf ball blueprint that is set to the player’s position at the start. I then made the game mode blueprint spawn a new golf ball actor that the player possesses. Once the player is respawned, the respawn event binds itself to the new golf ball.