Faculty of Computer Science
and
Engineering

COMP3421 - Computer Graphics

Final Project Report

Group - Goofy Golf

Group Members:

1. **Xielin Wu (z5313388)**
2. **Izaac Coffey (z5218543)**
3. **Jefferson Zheng (z5312313)**
4. **Weichen Tie (z5308889)**

Table of Contents

Section 1
Purpose and Features of our System

1.1 System Overview

Our System, GoofyGolf is a third-person casual single player/multiplayer game in which the player takes control of a golf ball to navigate through different courses each with their own theme and obstacles. Along the way, players may come across power-ups, which will either help the player in finishing the course, or cause chaos amongst other players in a multiplayer setting. The goal of our system is to provide a family-friendly casual "goofy" experience for players to enjoy and/or play and compete with their friends.

1.2 Key demographics of the target audience

The key demographics of our system are children, families and casual gamers. We cater to our target audience by providing them with a simple and intuitive game with easy to understand controls and low skill expression gameplay. Finishing a level faster or faster than someone else is determined a little by skill, but mostly by the randomness of power-ups and obstacles. This provides a casual experience in which children and casual players can play and enjoy without the need to master the game.

1.3 System Style

Our system is a game system, with its primary purpose being to entertain. We have maintained a low poly cartoon theme throughout both desktop and VR versions of the games to ensure a consistent and enjoyable experience. Our game also aims to be "goofy" by incorporating various unique power-ups and obstacles not commonly seen in golf games. This choice of a less serious theme mirrors the concept of our game which is to deliver a casual golf game experience to those less competitive or serious about golf games, namely children and casual gamers.

1.4 Differences and Similarities with Existing Systems

In our project proposal we analysed many existing golf games, namely, Wii Sports Golf, Golf with Your Friends and Walkabout Minigolf VR. We drew many inspirations which we built upon from these games. For example the mouse control system for our desktop version of the game is largely inspired by Golf with Your Friends and our motion controlled VR version's controls share similarities with those from Walkabout Minigolf VR. This is because we wanted to ensure that the controls of our game were consistent with similar systems such that the learnability of our game was improved for new players who may have played similar games in the past.

However the power-ups, level design and obstacles are almost all unique to our game quite different from those seen in the existing systems, this difference in ambience and gameplay is largely what separates our system from others. Also our system provides both VR and desktop game functionality which none of the existing systems we researched are able to provide, either specialising in one platform or the other. This creates a value proposition for users who want to play in both VR and desktop as they will not need to purchase two different games to do so. Also our VR version features a “pass the headset” multiplayer feature, allowing for a multiplayer experience offline and with one headset. This is a feature that is not provided by Walkabout Minigolf VR, which only allows for online multiplayer, which also requires both players to have their own VR headset. This functionality makes our game more attractive for players who want to play multiplayer but only have one VR headset, which is quite a likely scenario as VR headsets are not yet commonplace.

1.5 Engagement with Target Audience

The target audience of our 3D system ranges from users between 10-70, while our target audience of our VR system audience ranges from 15 - 60. The reason for this is because younger and older audiences may be more susceptible to experiencing motion sickness due to the VR nature of VR systems, so for the sake of the safety of users the age range has been reduced for VR. Our system is also aimed more towards groups up to 6 people.

Our system is designed to engage our audience through a competitive and “goofy” experience alongside an immersive in-game environment. When playing in singleplayer the competitive experience is significantly bottlenecked, as you would be trying to beat either the par or your best stroke count for a stage. Although some users may find this boring, this mode is mainly aimed for users who enjoy playing by themselves and trying to beat their high score.

While playing in multiplayer both the competitive and “goofy” aspects of our game are further emphasised. Playing in multiplayer most likely means you are playing against people you are familiar with. This makes the game more competitive and enjoyable for all users due to the nature of multiplayer games.Our multiplayer mode also features our multiplayer exclusive power-ups, the ballistic missile and the tornado power-up. These two power-ups are our system’s signature “goofy” power-ups, so being able to use these power-ups against other users adds to the “goofy” aspect that our system is trying to show.

1.6 Technical and Non-Technical Components

1.6.1 Technical components

1. Player
    - In the non-vr version of our game, the player takes a third person perspective of their golf ball and is controlled by clicking and dragging to determine the direction and power in which you want to hit the ball.
    - In the VR version of the game, the player takes the perspective of a person, and can hit their golf ball using a golf club in which they hold.

2. Power-ups
    - Throughout the golf courses there will be power-up boxes that, when broken, give the player a random power-up.
    - There are power-ups that benefit the player or disrupt other players.
    - Power-ups that disrupt other players will not be available in single player.

3. Golf Course
    - Throughout the golf course there will be obstacles that can impact the movement of the golf ball.
    - Obstacles are in the style of each course's theme and can either block or destroy the player's ball.

4. Lighting
    - Our system uses a combination of direction light as well as skylight

5. Completion conditions
    - The condition for determining the winner after all players have putted each hole are the same as real life golf

6. Single and multiplayer option
    - Users have the option to play the game either in singleplayer or multiplayer mode.
    - Multiplayer is in real-time meaning multiple instances of the same game, may join a host and each control their own ball simultaneously. Multiplayer can be toggled between LAN and online.
    - Multiplayer for the vr version is not real time, but rather players take turns hitting their ball.

7. Vs game mode

- If the multiplayer option is selected, users can select the vs gamemode. This game mode consists of players each taking turns to try putt their own golf ball. The aim of this gamemode is to be the first person to putt their own golf ball.

8. Course Completion
    - Once the user/users have all putted a hole, they are moved to the next level

9. Out of bounds
    - If the ball is hit out of bounds, or hit by certain obstacles, they are returned to the original position of their last hit.

10. HUD
    - Stats such as strokes and time in addition to obtained power-ups will be displayed on the screen
    - In vr, these HUD elements are displayed on the player's left hand as a hologram when players turn their palm facing down and look slightly above the hand.

1.6.2 Non- technical

1. Aesthetic Style
    - Our system uses low poly models along with stylized textures to create a cartoonish aesthetic style.

2. Theme
    - All of our courses are designed with different themes which differentiate and make them unique. The different themes also impact the obstacles of each course. For example, our course 2 is dungeon themed, so the obstacles are designed to be traps found inside a dungeon.

Section 2
Explanation of Implementation

2.1 Courses (Level Design)

Each course is designed around a different theme, with theme specific obstacles, models and textures. They are all designed however, with a standardised layout of a flat golf course, with walls on either side and a hole at the end of the course.

2.2 Course 1

Course 1, is designed around the theme of fantasy floating islands.

Assets

- Course layout, and floating island models are original.
- Remaining models, textures and materials used are from *FANTASTIC - Village Pack* by Tidal Flask Studios on the unreal marketplace.

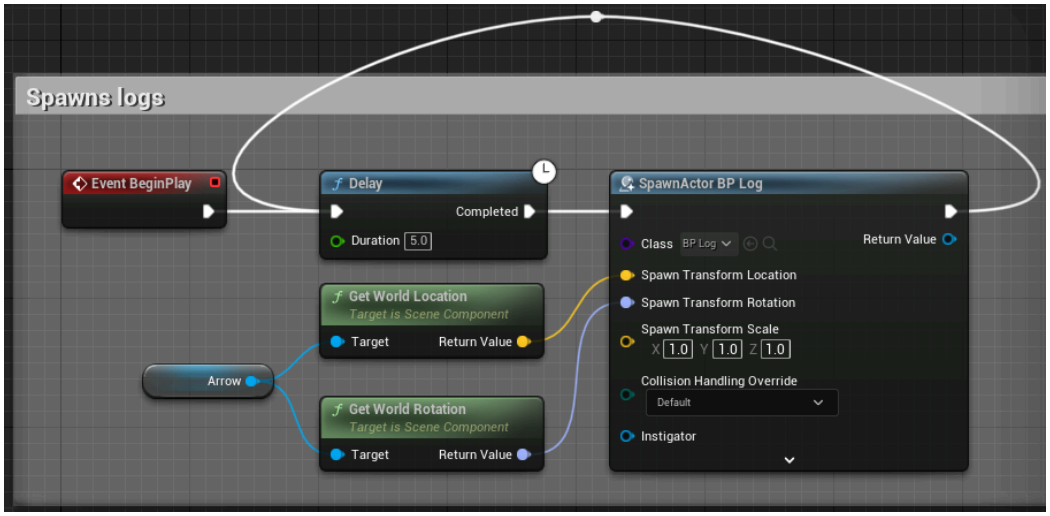
Modification to assets

- Added a rotation to the windmill
- Added velocity to the rocks so they float up and down slightly
- Modified variables in the water material to make it seem like it is flowing

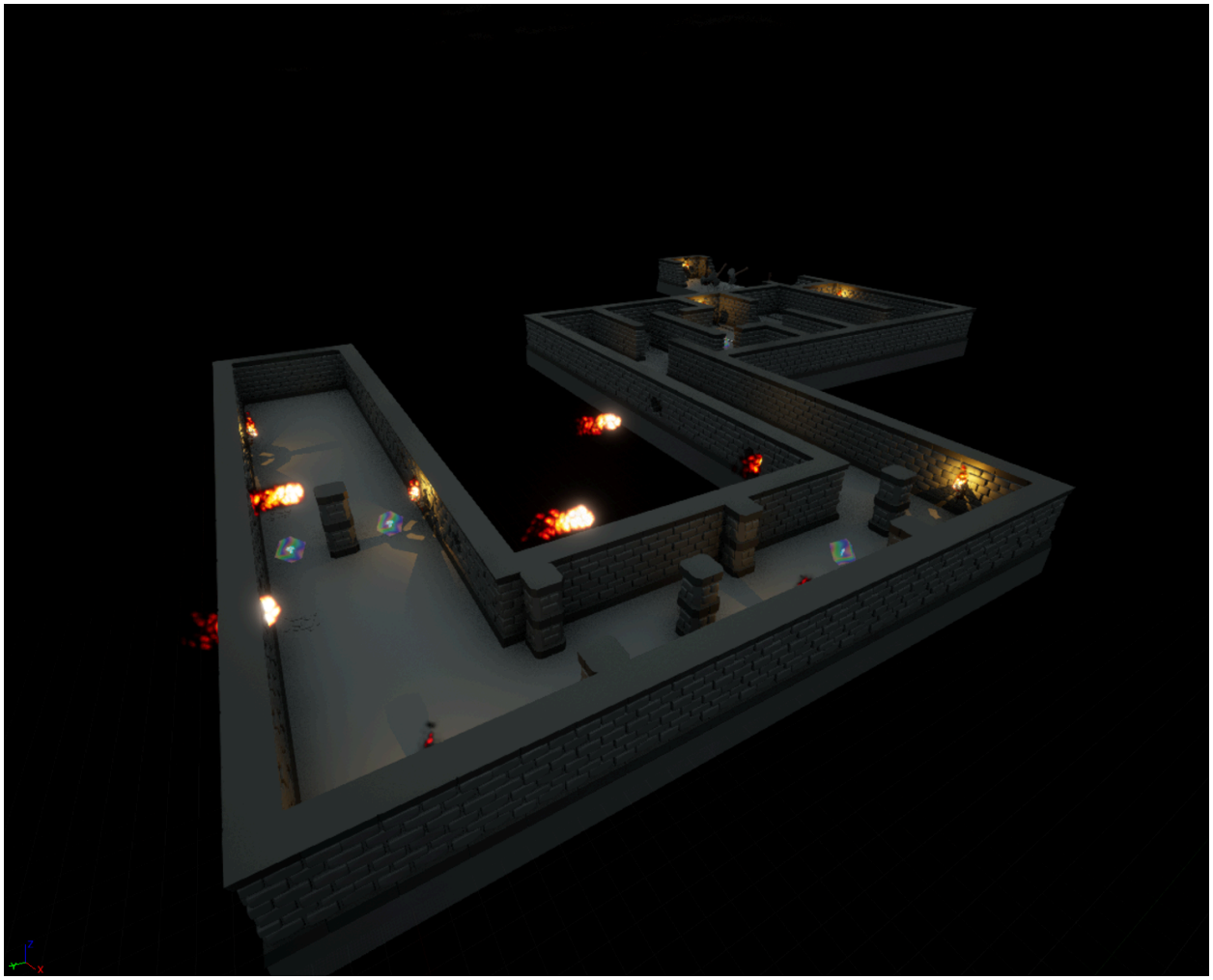
Obstacles

Many static objects are placed on the course to act as obstacles such as trees and walls.
One dynamic obstacle in this course are the logs which float across the river to block the path of the player.

The logs are simply a blueprint with a projectile movement component attached to them. They are then created behind the waterfall by a blueprint which uses the spawn actor of class function to spawn the logs after a variable amount of time.



2.3 Course 2



Course 2 is designed around the theme of a fantasy dungeon.

Assets

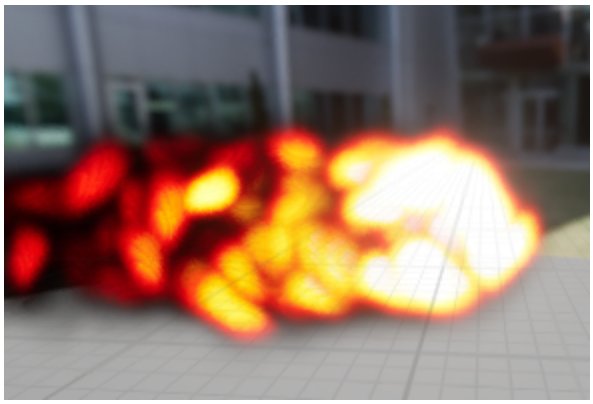
- Course layout, fire particles, and sawblade models are original.
- Remaining models, materials and textures used are from *KayKit - Dungeon Pack* by Kay Lousberg found at https://kaylousberg.itch.io/kaykit-dungeon

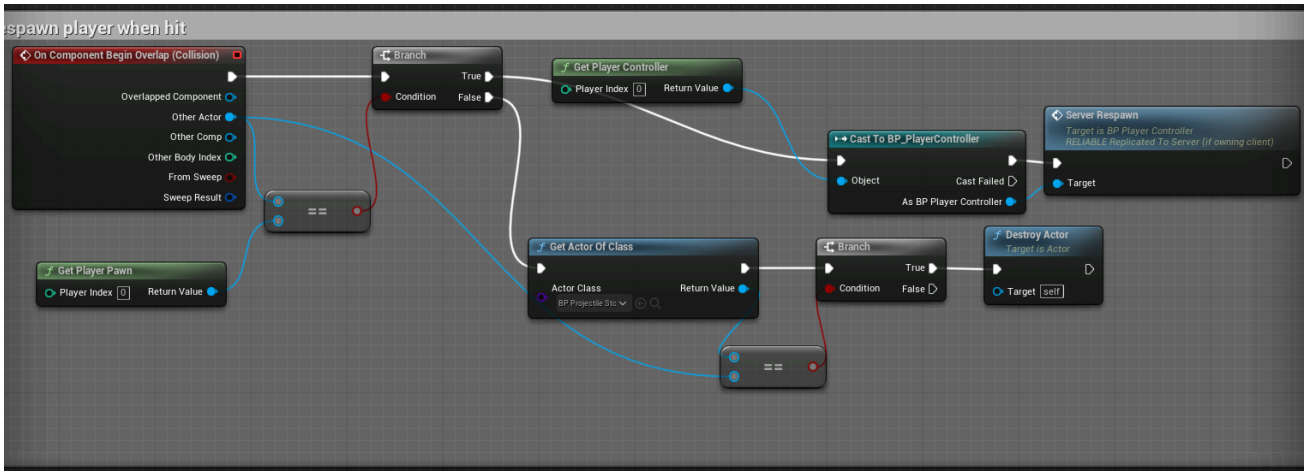
Modification to Assets

- Torches were originally unlit, added fire particles and point light to light them up
- Modified some walls to create seamless transition between walls textures

Obstacles

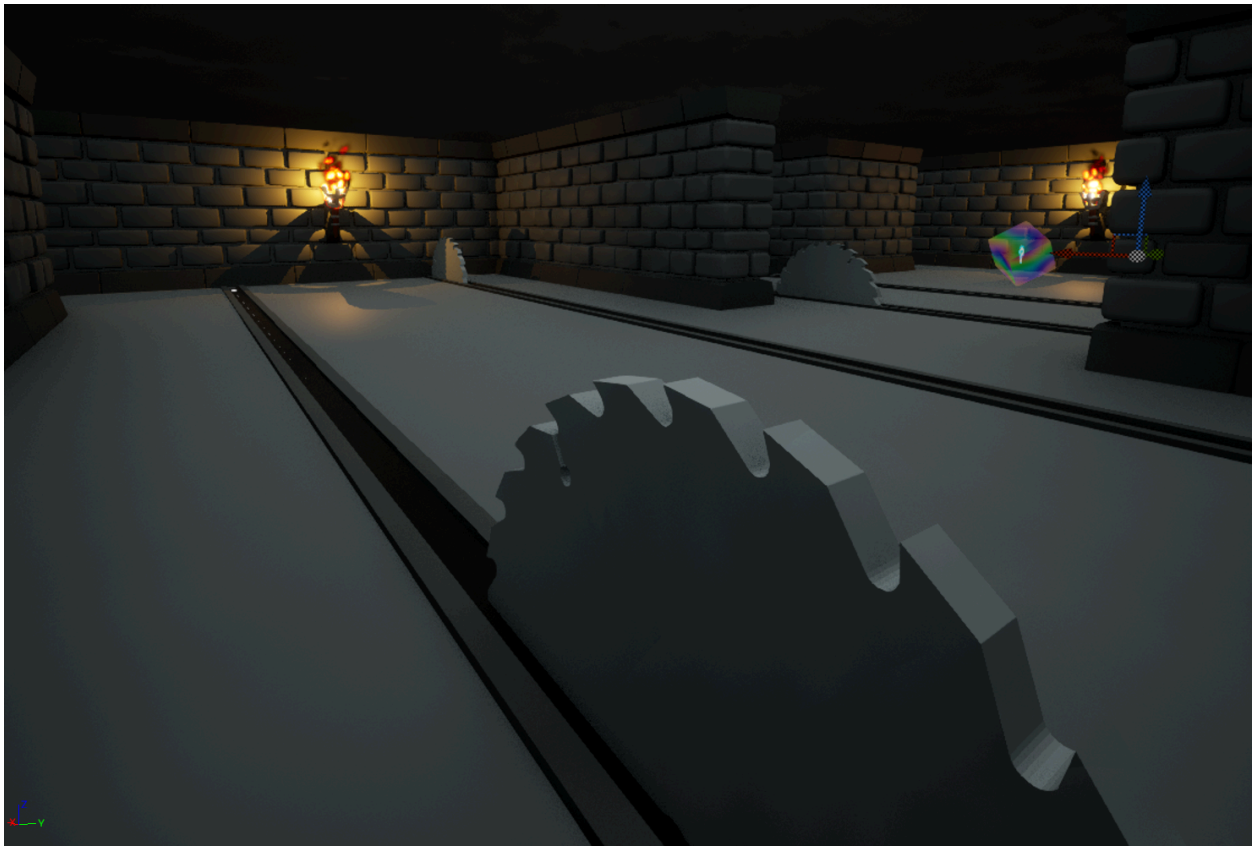
The fireballs are the first obstacle encountered in course 2 and they respawn the player's ball to their last location when coming in contact. They are spawned similarly to the logs in course 1.



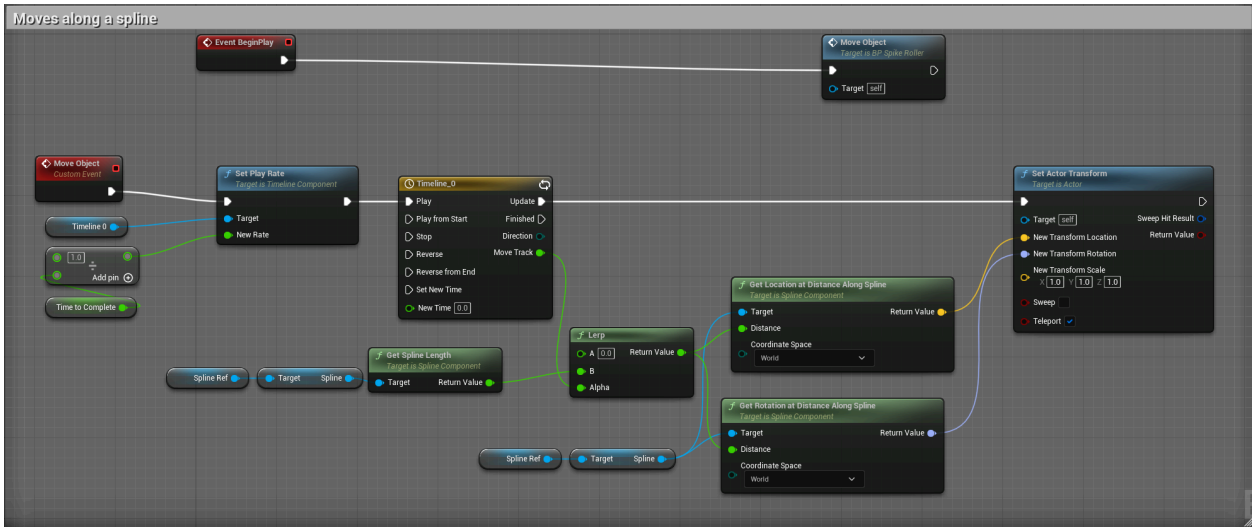


The blueprint code checks whether the fireball is coming in contact with the player, and if it is, call the respawn function in the player controller which resets the ball to its last location.

The sawblades are another obstacle, which moves from side to side and blocks the player's ball when colliding with it.

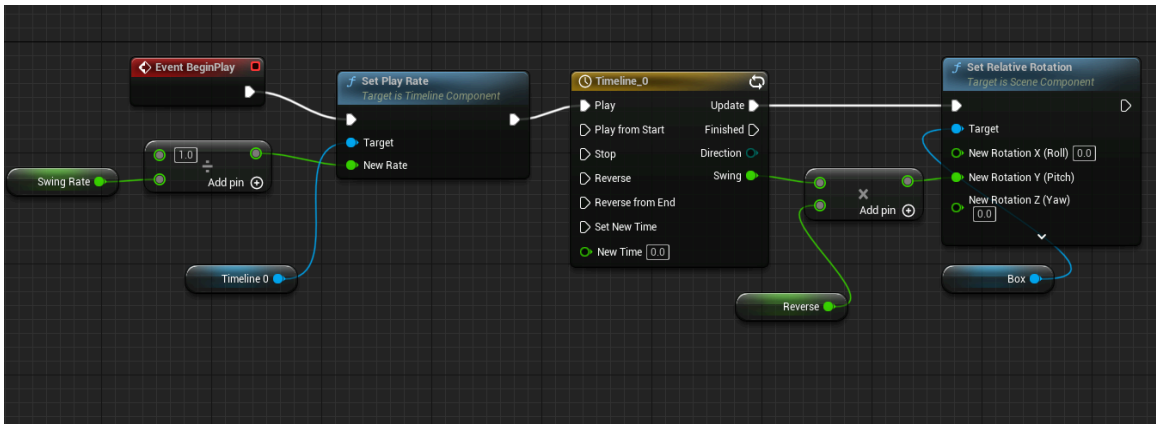


The sawblades move along a spline, which is done by using a timeline and lerp to move the blade along the length of the designated spline.



The last obstacle before the hole are swinging axes which attempt to block the player's path to the hole as well as hit them out of the course.

The axes swing left to right and are done by simply creating a timeline between two values and applying those values to the rotation on the axe.



2.4 Course 3

Course 3 is designed around the theme of tropical islands and pirates.

Assets

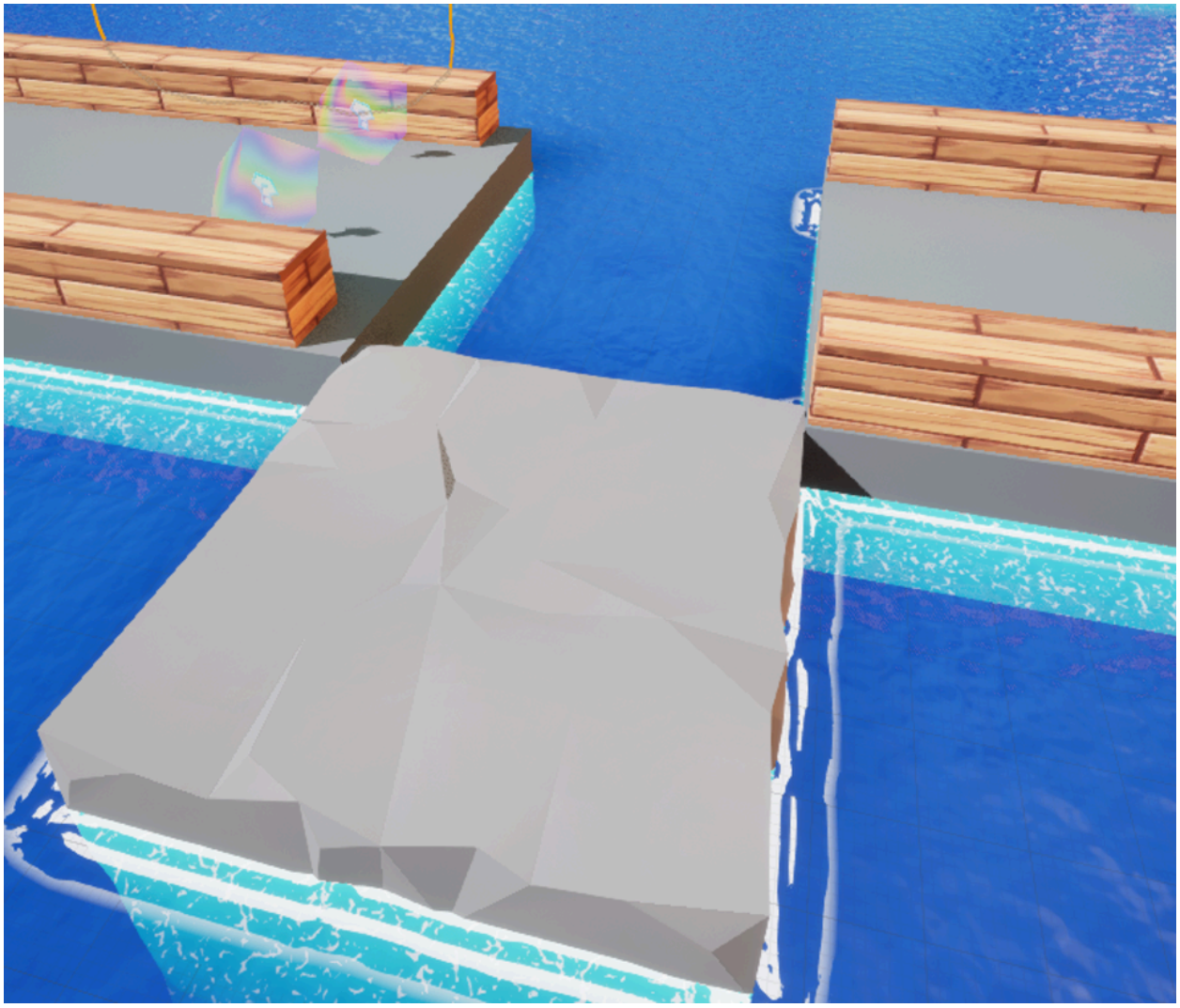
- Course layout, Fire balls, volcano smoke, and cannonball are original
- Remaining models, textures and materials used are from *Low Poly Style Deluxe 2: Tropical Environment* by WolfDigitalLLC on the unreal marketplace

Modification to assets

- Added smoke and spewing fireballs to the volcano
- Pirate ship shoots cannonballs from the cannons on its side
- Fireballs also explode on impact with the course

Obstacles

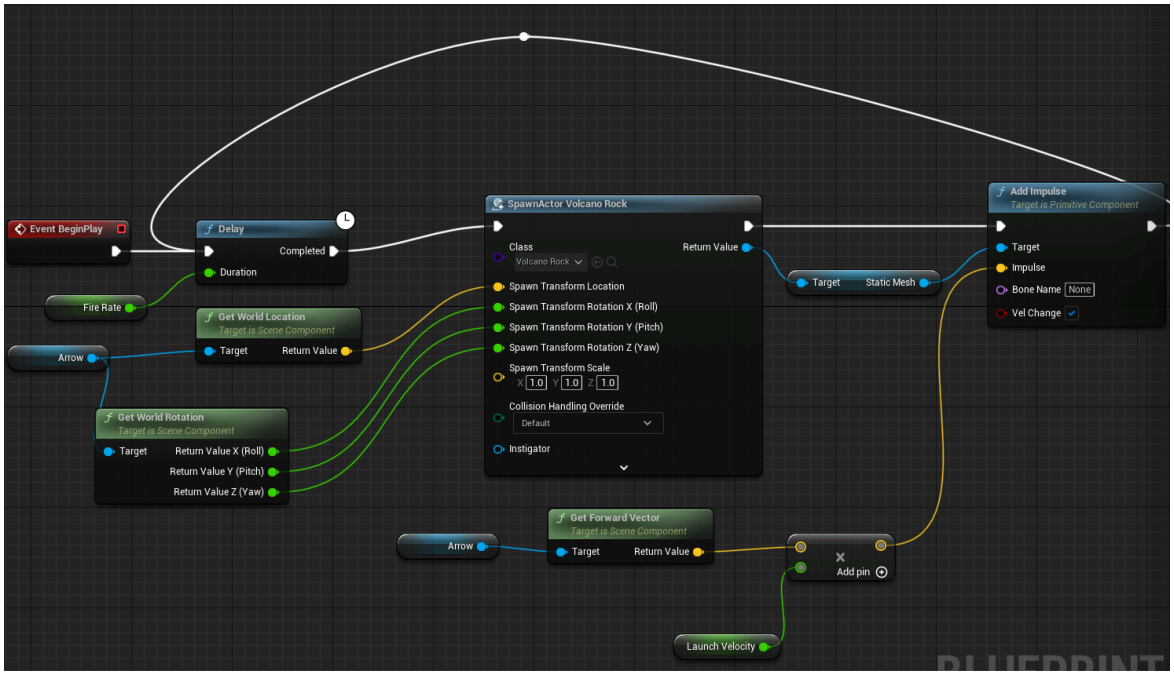
The first obstacle is a moving platform which travels side to side. Players must successfully roll across the platform as it moves to traverse the course.



The platform, like the sawblades in course 2, moves along a designated spline. The blueprint code for it is also identical.

The next obstacle are fireballs which spew out of the volcano into a part of the course. The fireballs respawn the player when they are hit.

The fireballs are spawned using a blueprint with an arrow component. Force is then added in the direction of the arrow component and placed on top of the volcano, to simulate the volcano spitting out the fireballs.



The last obstacle of the course are cannonballs which shoot out of the cannons in the pirate ship. They are solid and block the player's advance to the hole by detonating upon collision with a player, launching them in a random direction.

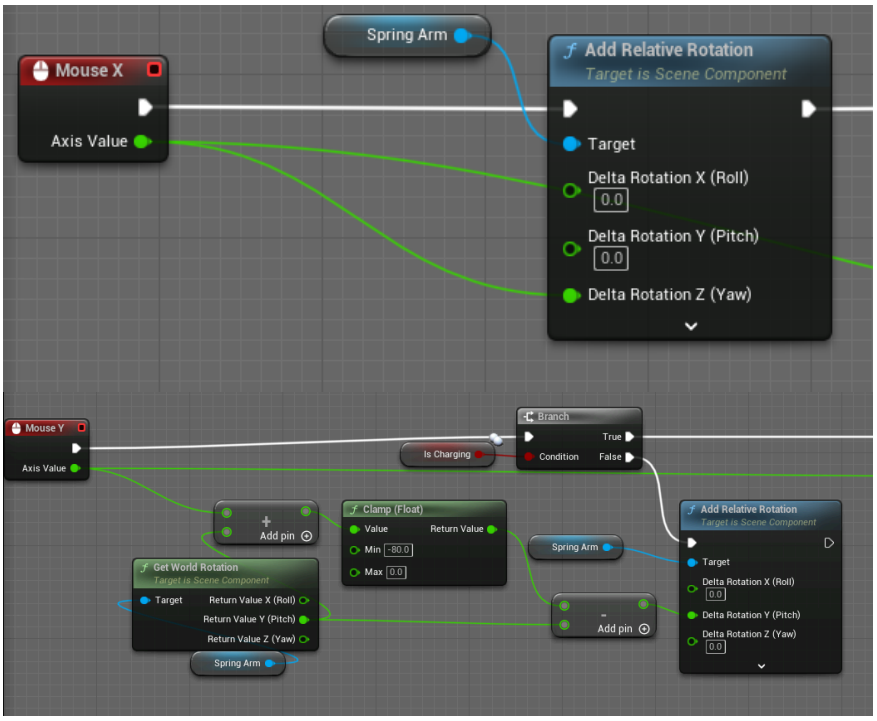
The cannonballs are spawned identical to the fireballs with the explosion mechanics identical with missile detonation.

2.5 3D-Golfball

The 3D version of the ball utilises a mouse only approach for the camera and launching related inputs and is also responsible to be reactive to both the environment and the effects of power-ups.

2.5.1 Camera

The ball camera is in third-person perspective as it is mounted to a spring arm and looking around is achieved by moving the mouse around on the mousepad. The camera's pitch is also locked between 0 and 80 degrees to prevent gimbal lock issues.

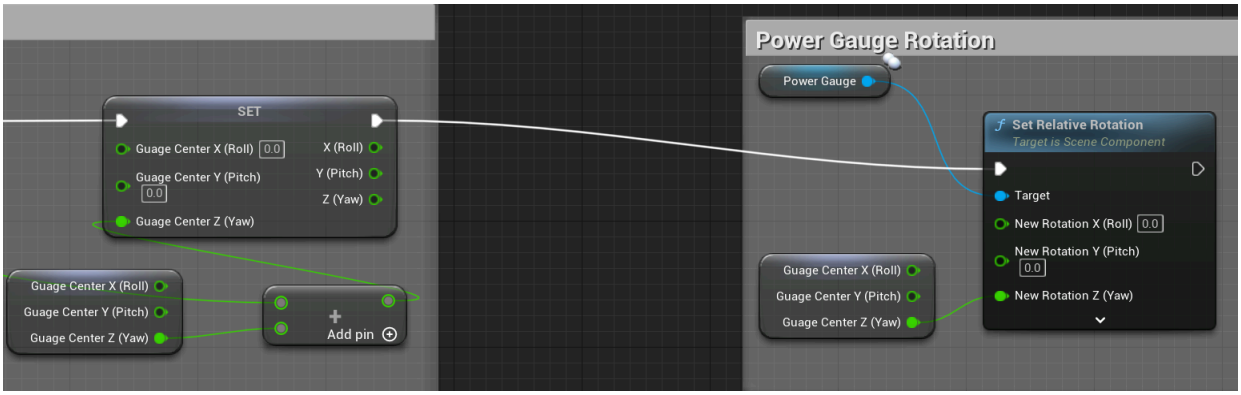


2.5.2 Power Gauge

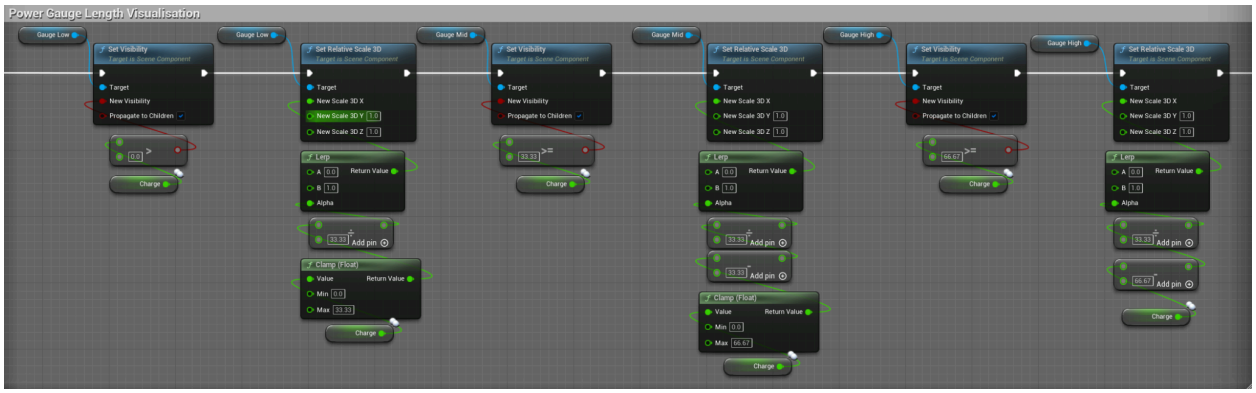
The power gauge is used as a visual aid for the player to visually gauge the amount of charge and therefore impulse delivered to the ball upon launch. Though conceptually simple, the powerbar utilises many rendering techniques to create the desired effect. Many of which will be discussed in further sections.

2.5.3 Power Gauge Rotation

Rotation of the power gauge is achieved by setting the relative rotation of the powerbar to be the sum of the previous yaw of the power gauge and deltaX mouse movement.

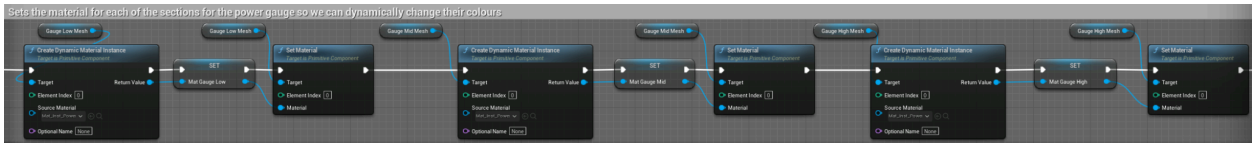


The total length and visibility for each section of the power gauge is determined by the charge present in the ball. By checking if the charge is greater than a certain critical value, we can enable or disable the visibility of each section while also scaling that bar section in the X-direction to mimic the effect of the bar growing and shrinking.

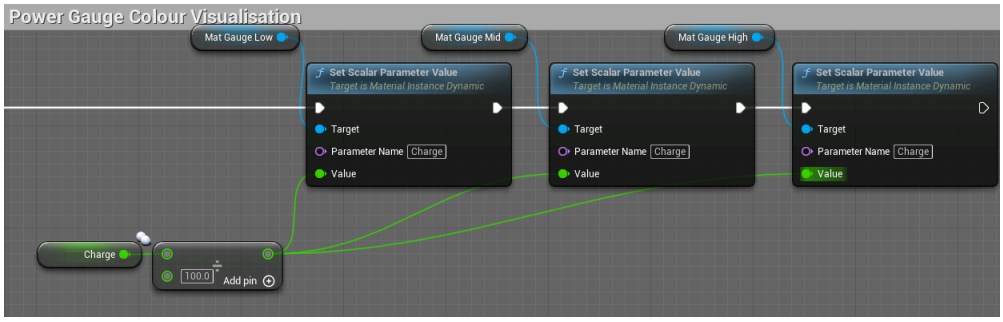


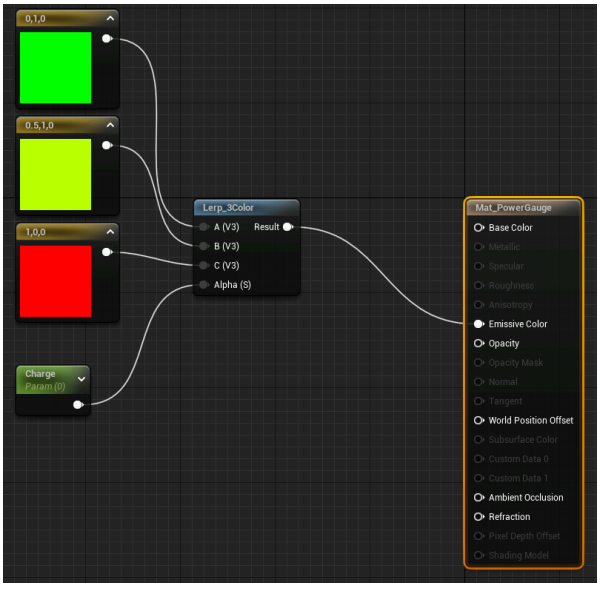
2.5.4 Power Gauge Colour

Colour changing in the power bar is achieved by creating dynamic material instances upon the ball spawning into the level.



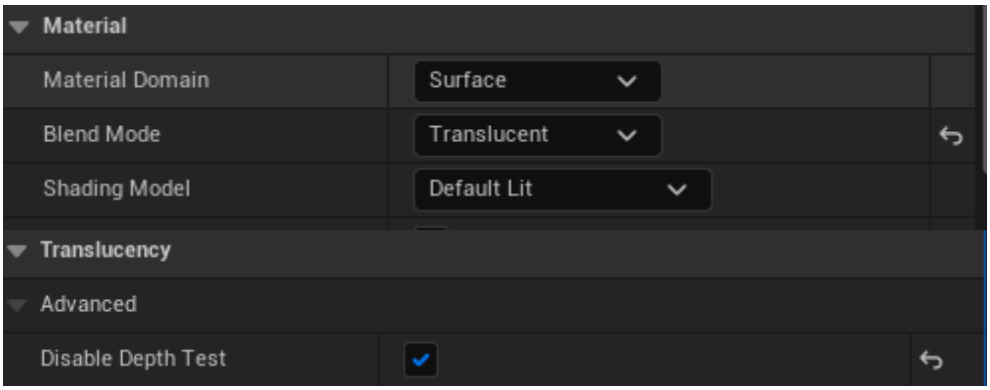
The current charge of the ball is then passed into those dynamic materials where a three way colour lerp allows it to smoothly change colours from green to red.



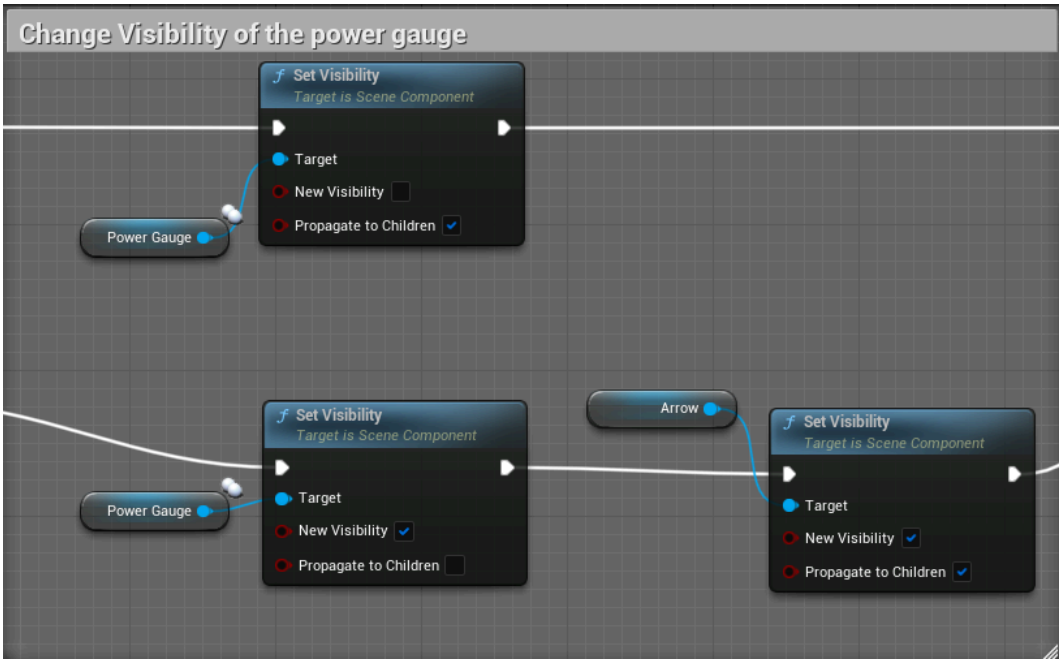


2.5.5 Power Gauge Rendering

Initially, a issue that we came across was due to the presence of the Z-Buffer, if we face the power gauge towards a wall, any part of the gauge that is behind the wall will be clipped and hence the player will be unable to visually gauge the exact amount of charge built up apart from the colour changing of the bar. Hence we took the approach of making the power gauge use a blend mode of translucent and also disabling depth testing to have it render above all other fragments in the GPU.

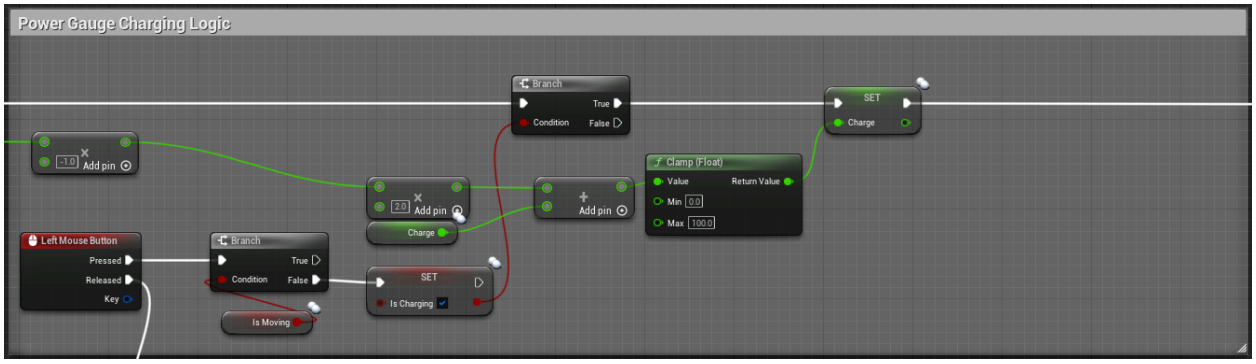


When the ball is in motion, the power gauge is hidden from the player so as to not over clutter the screen.



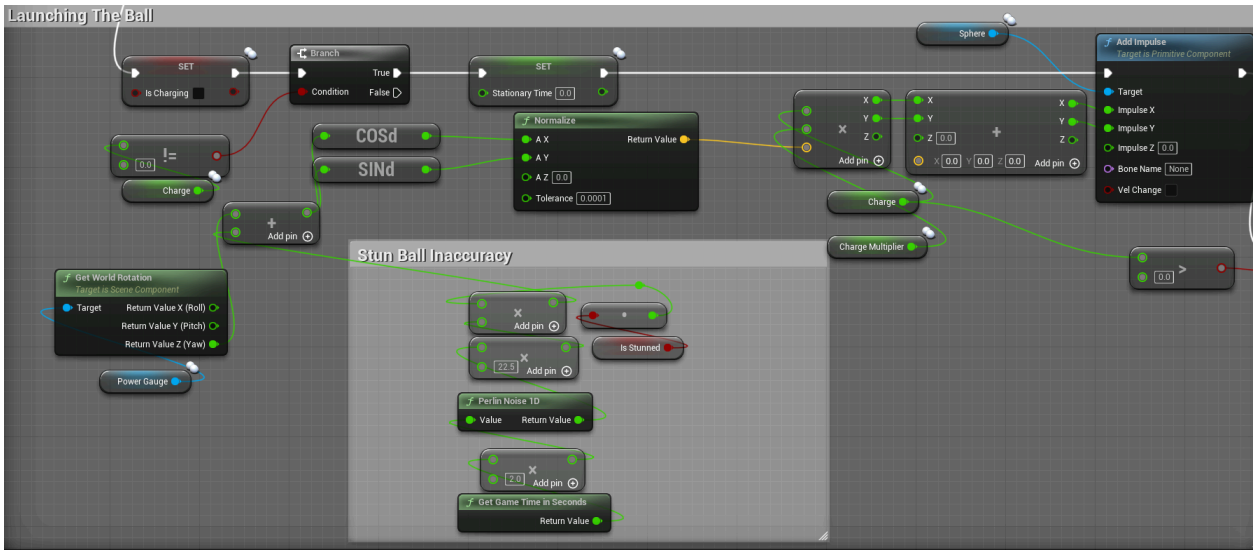
2.5.6 Charging

Charging of the ball is achieved by holding down the left mouse button and pulling the mouse downwards which will then linearly increase the total charge that is stored in the ball. The ball will check whether it is first stationary prior to allowing the user to charge.



2.5.7 Launching

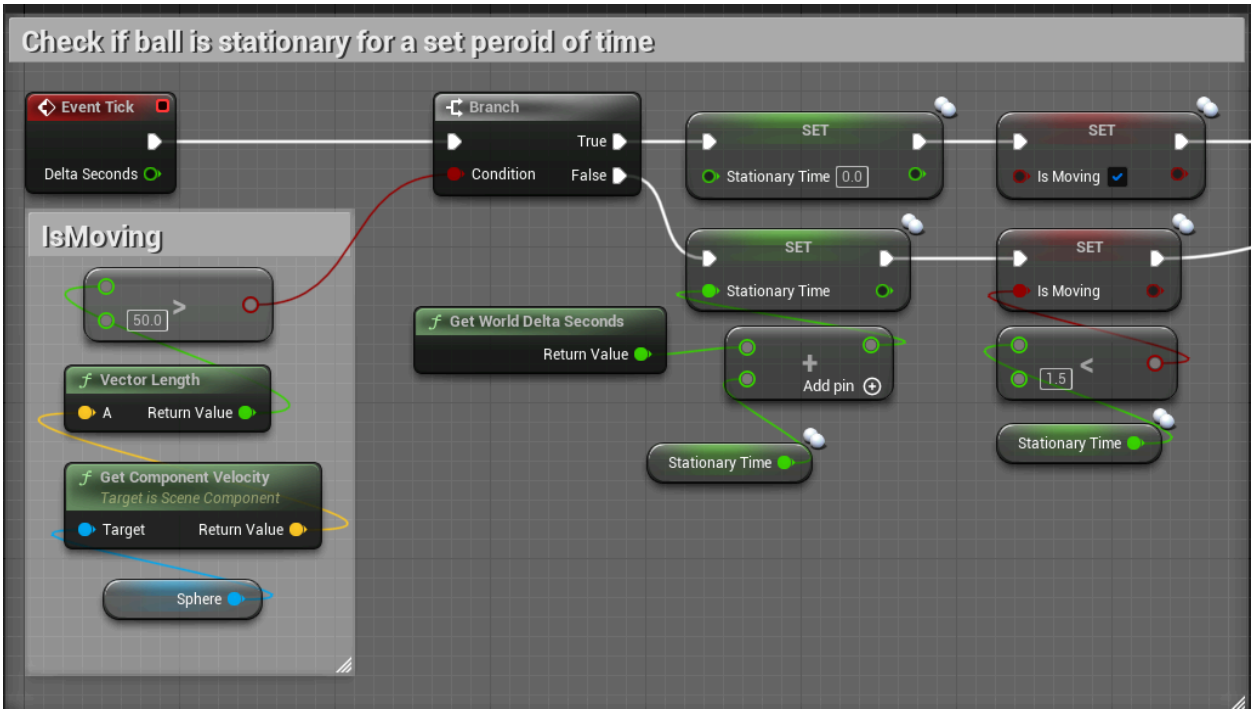
The ball launching algorithm is achieved via application of linear algebra to determine the launch direction.



By realising that we can break down the yaw of the power gauge heading into sine and cosine components, we can henceforth determine the ratio of impulse to deliver to the ball in the world Y and X directions respectively. At this stage, we also have an opportunity to add inaccuracy towards a shot which will be discussed further on in the report for the “Stun Ball” power-up. The direction vector is normalised before multiplying it by the charge since any deviation from a unit length will cause the velocity upon releasing the mouse to not be consistent between shots.

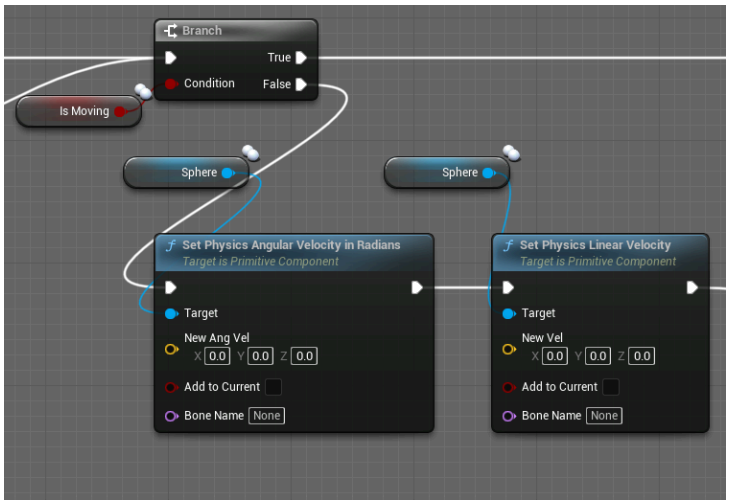
2.5.8 Stopping

The process of stopping the ball is split between two parts; Checking if the ball has been moving below a critical speed for more than a specified amount of time and to stop the ball from moving if it has.



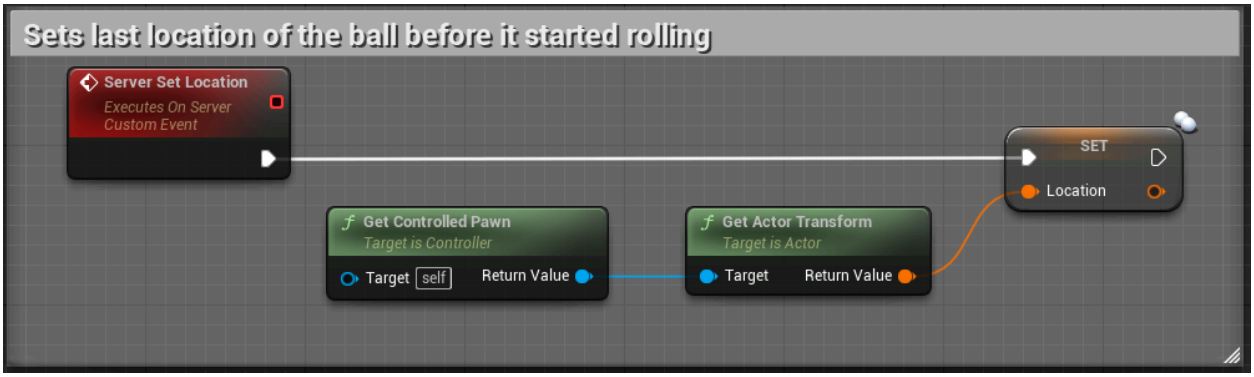
As per the diagram above, a ball in motion is defined as the ball having a velocity of greater than 50 units. If this condition is not satisfied, the ball will begin a “Stationary Time” timer which, as its name implies, tracks the total time the ball has been idle. If at any point in this stage the ball returns to having a velocity greater than 50 units the timer will be reset. By doing so, we can account for the case where the ball rolls up a ramp but doesn’t make it past and thus rolls back down as when the ball reaches its maximum, the ball will briefly have a velocity of zero before rolling back down.

If the time has successfully surpassed the critical threshold, then the ball will immediately stop in place and the turn will be considered over.

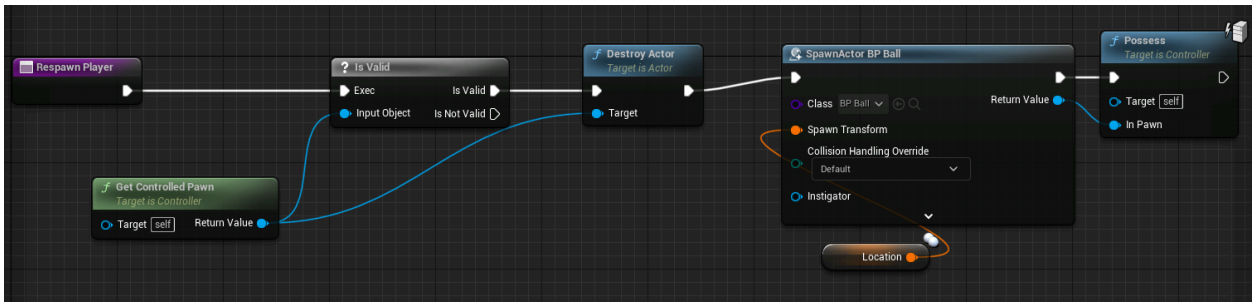


2.5.9 Respawning

Upon the ball being deemed stationary, the ball's respawn point is updated to be the current location of the ball by the following code on the server side.



If the ball falls out of bounds by overlapping with a trigger box or comes in contact with an instant death object such as a fireball, the ball will be sent to its last valid position by destroying the current ball and spawning a new ball in its original location and possessing it.

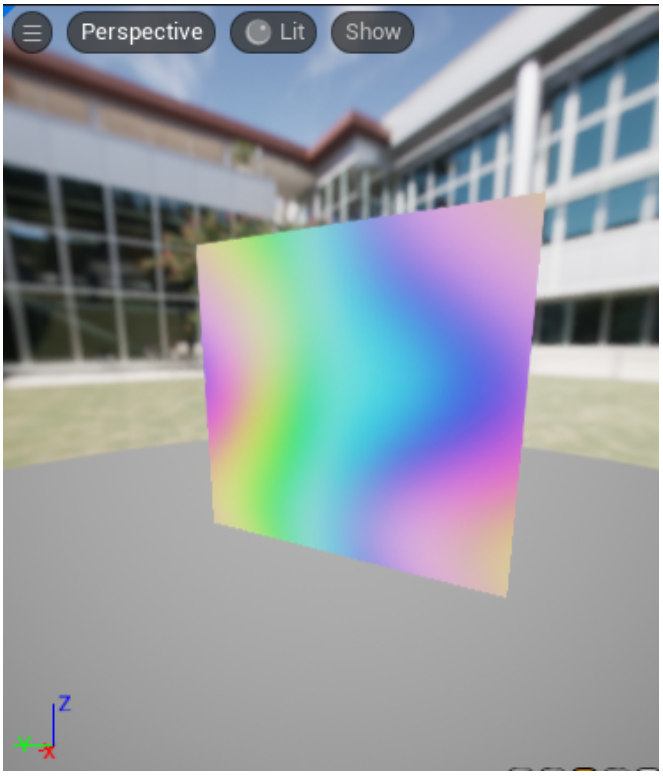


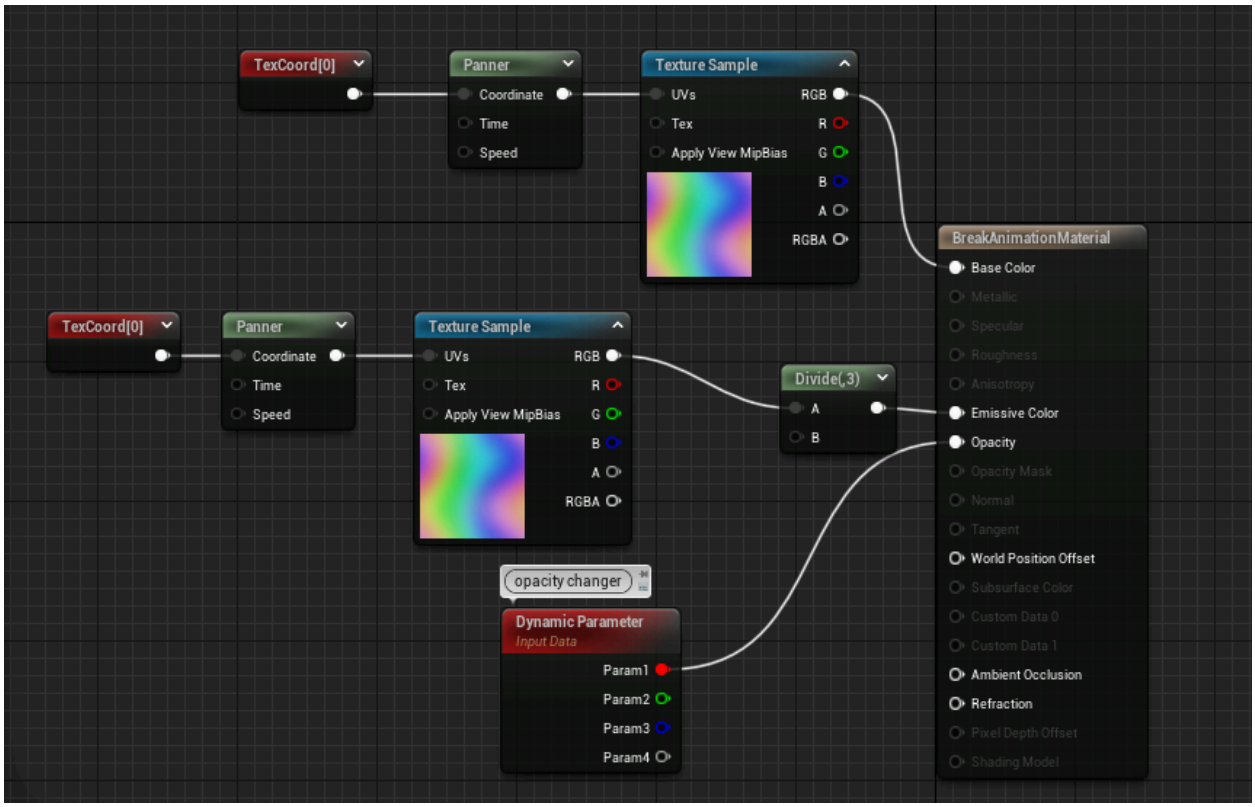
2.6 Power-up Niagaras

All the niagara systems are original and are used as animations or special effects in our system. They are used as obstacles in courses but mainly in animating the different power-ups that the player obtains

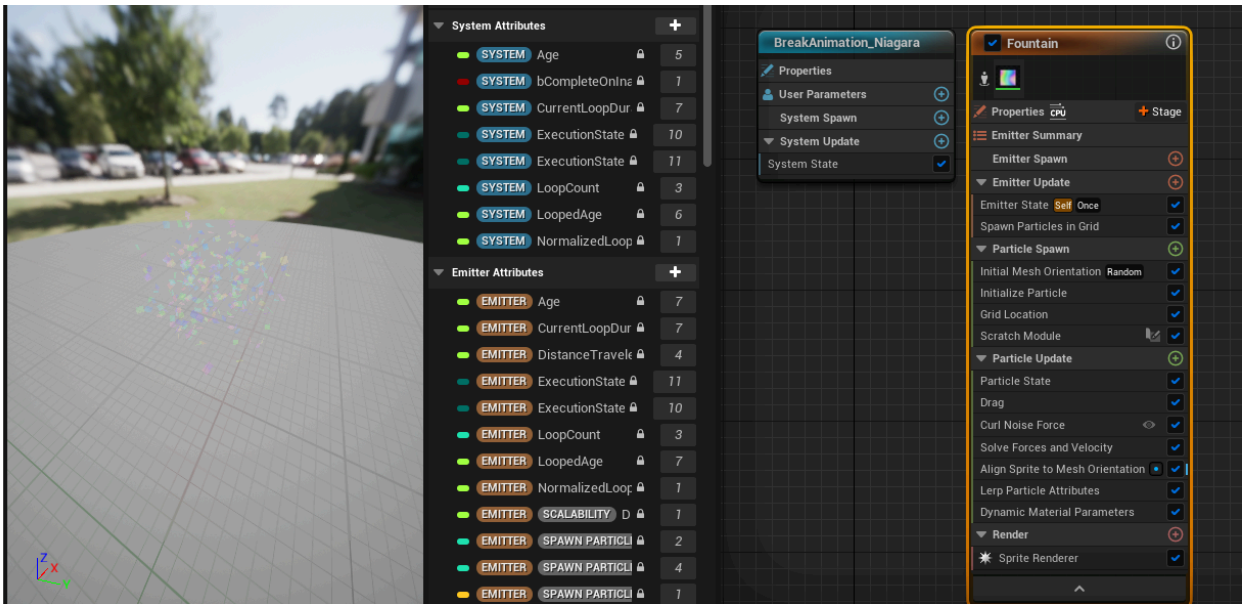
2.6.1 Break Animation

The Box break animation is used when a power-up box is destroyed by the player.
The material is made from the same seamless rainbow texture that is used to make the power-up box.

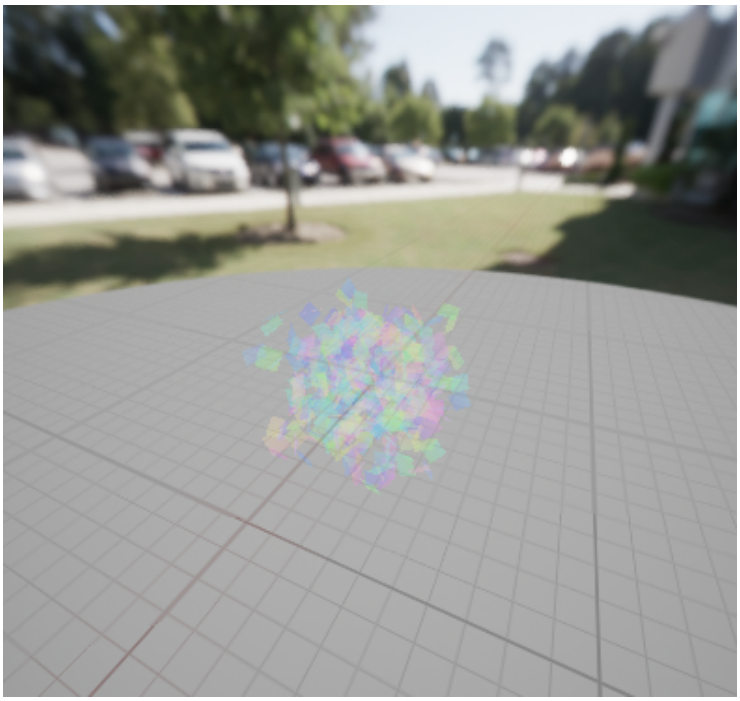




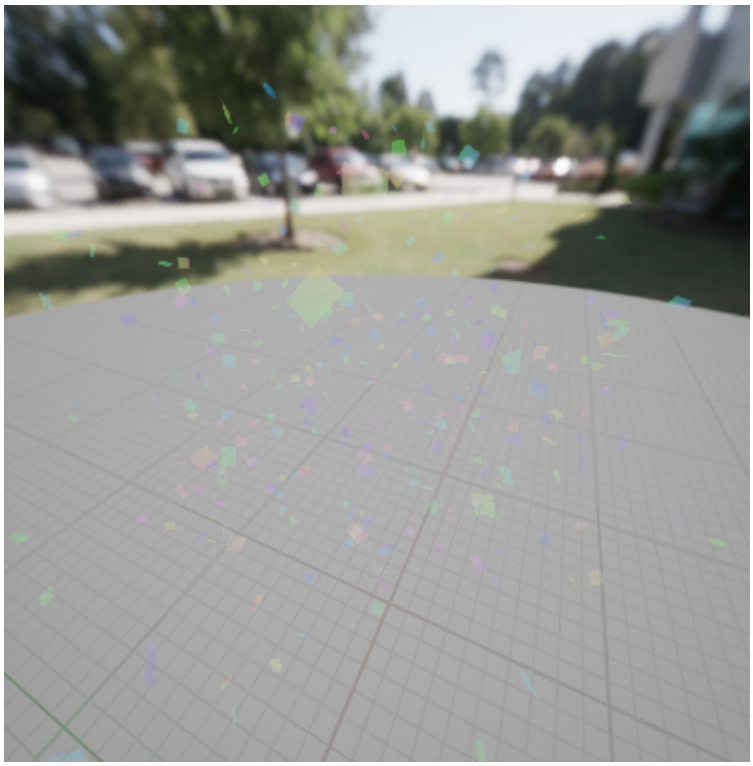
This material is then used in the break animation niagara which emits the materials as particles.



The initial spawn variables are adjusted so that the particles initially start in a cube like state.

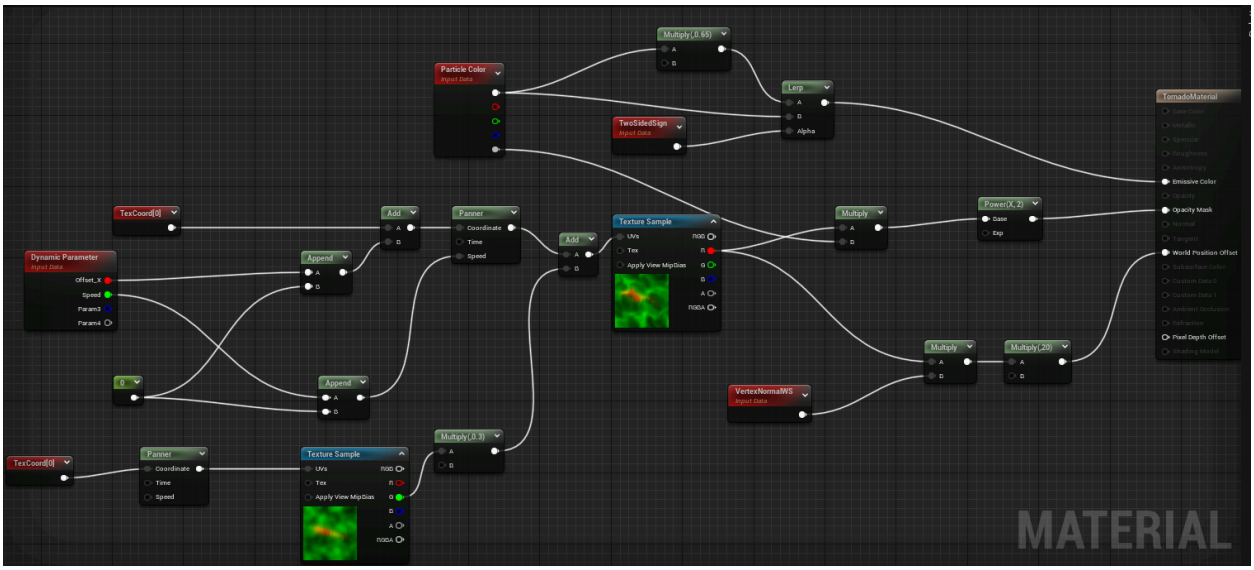


While the curl noise force and dynamic material parameters are adjusted so that the particles “burst” and fade away.

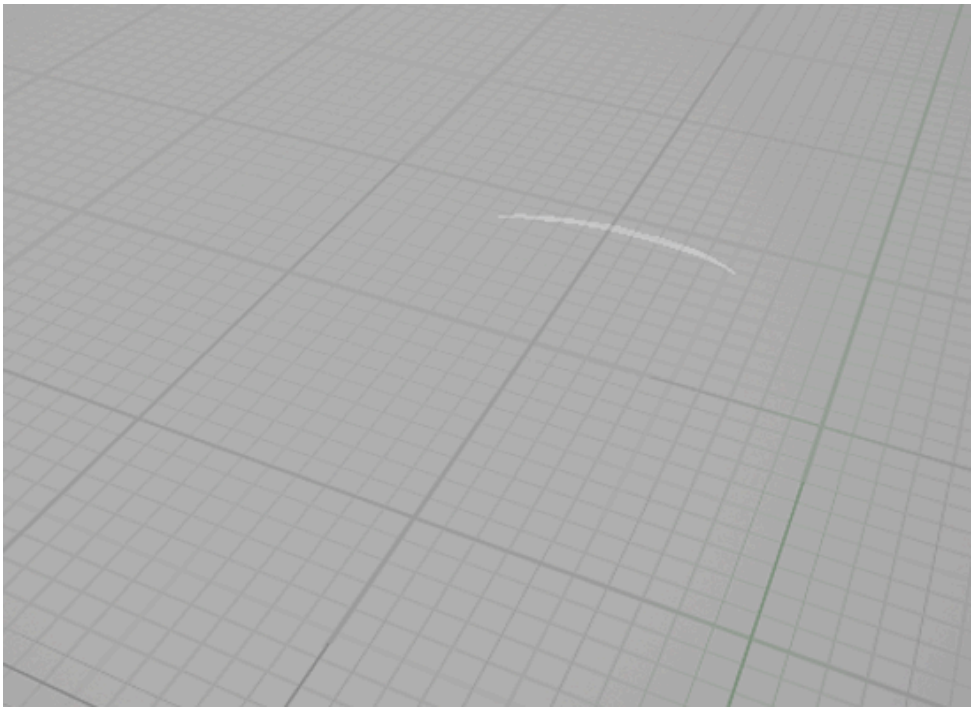


2.6.2 Tornado

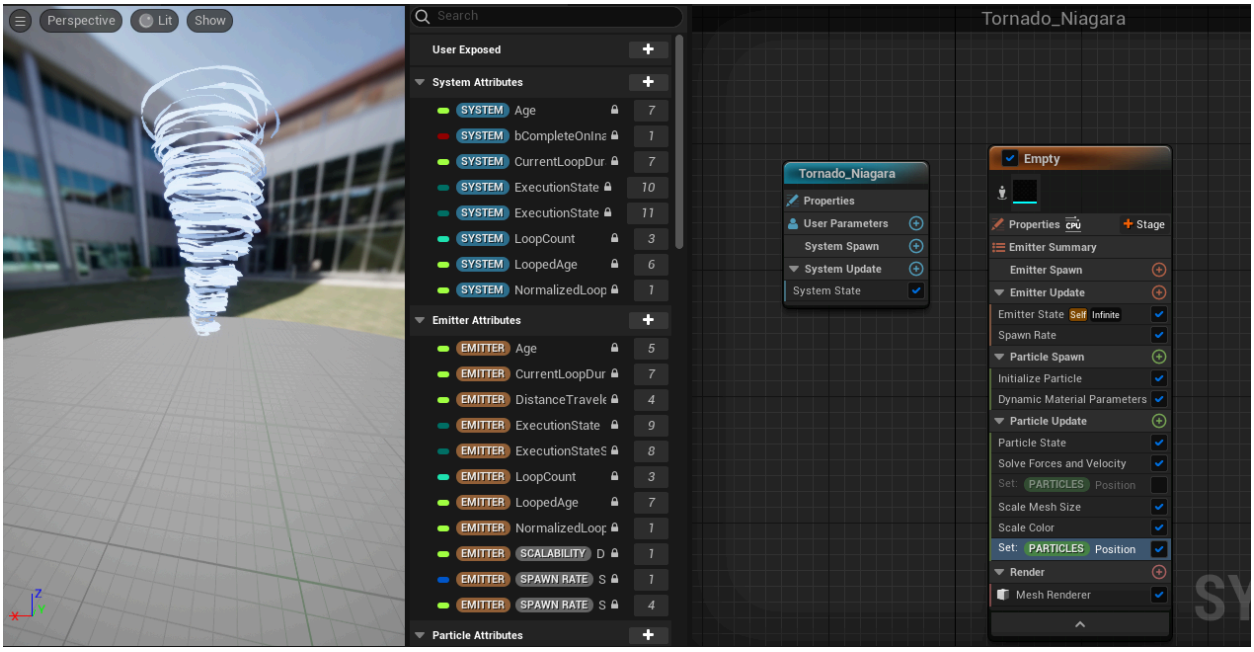
The Implementation of the tornado starts from a custom made tornado material.



Which is made to look like a spinning line. This material is then converted into a mesh

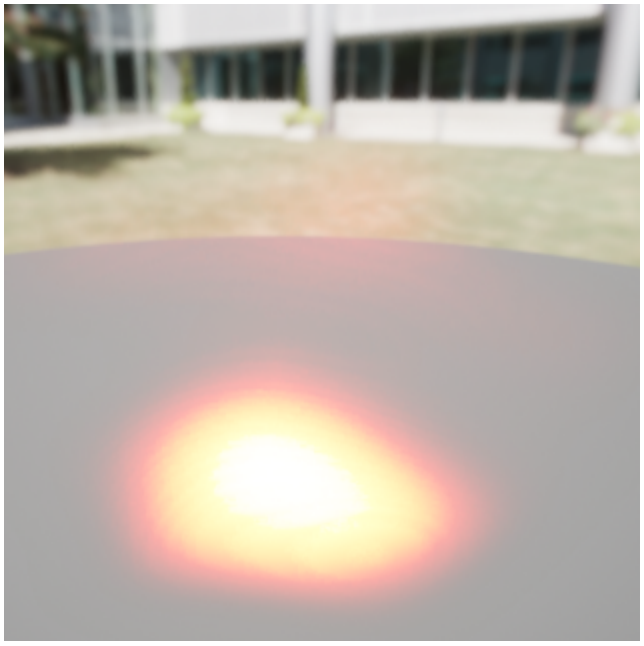


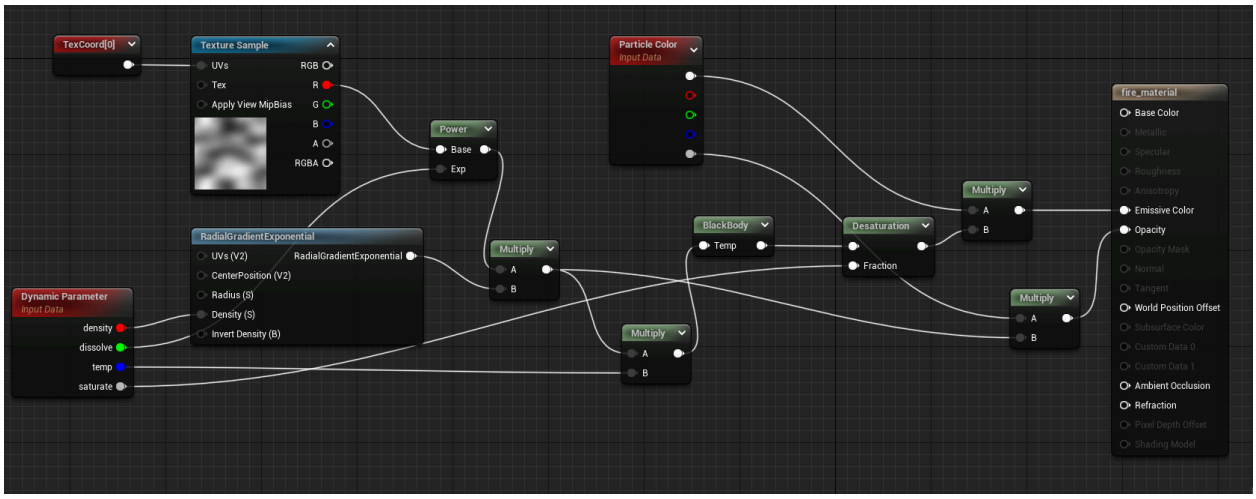
This mesh is then used as the base of the tornado niagara. By adjusting the spawn rate, mesh size, particle position and initial particle position the tornado niagara is created.



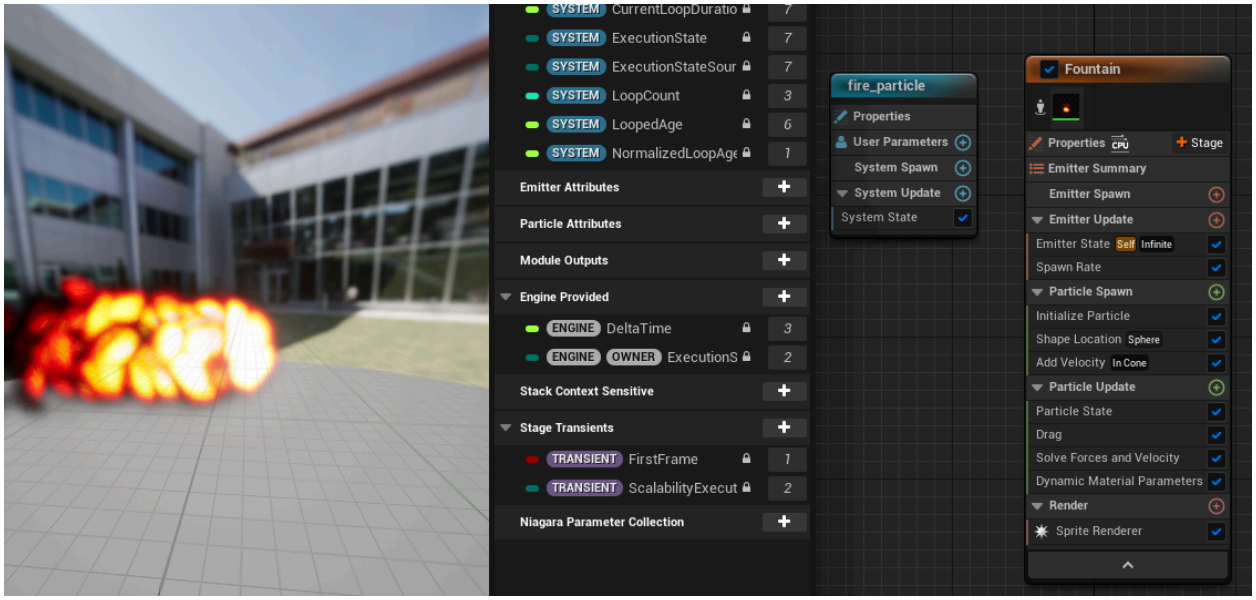
2.6.3 Fireball

The fireballs are made by first creating a new material for the niagara system to use.
The material is made by multiplying a noise texture together with a radial gradient exponent node and many adjustable variables into the material's emissive colour and opacity, to create a blurry orange material.



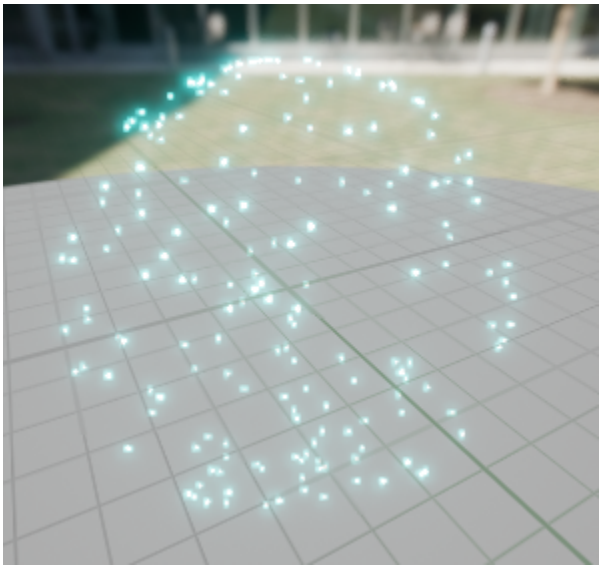


The material is then used in the sprite renderer of the niagara, which emits the material as particles in the niagara. The fireball is then created by adjusting the spawn rate, colour, velocity and size of the particles.



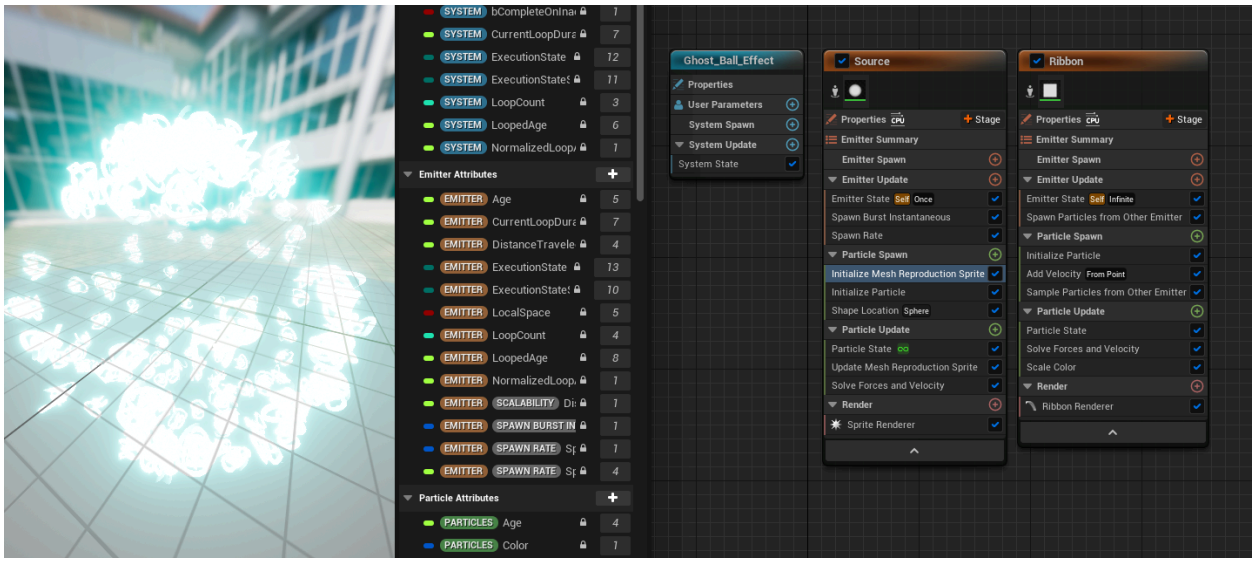
2.6.4 Ghostball

The ghostball niagara is used when the player uses the ghostball power-up. The niagara is created by first creating a skeletal mesh of the player ball. The mesh is then used as a spawning location for the sprites in the niagara. By adjusting various other properties, we have particles spawn around the surface of our skeletal mesh.



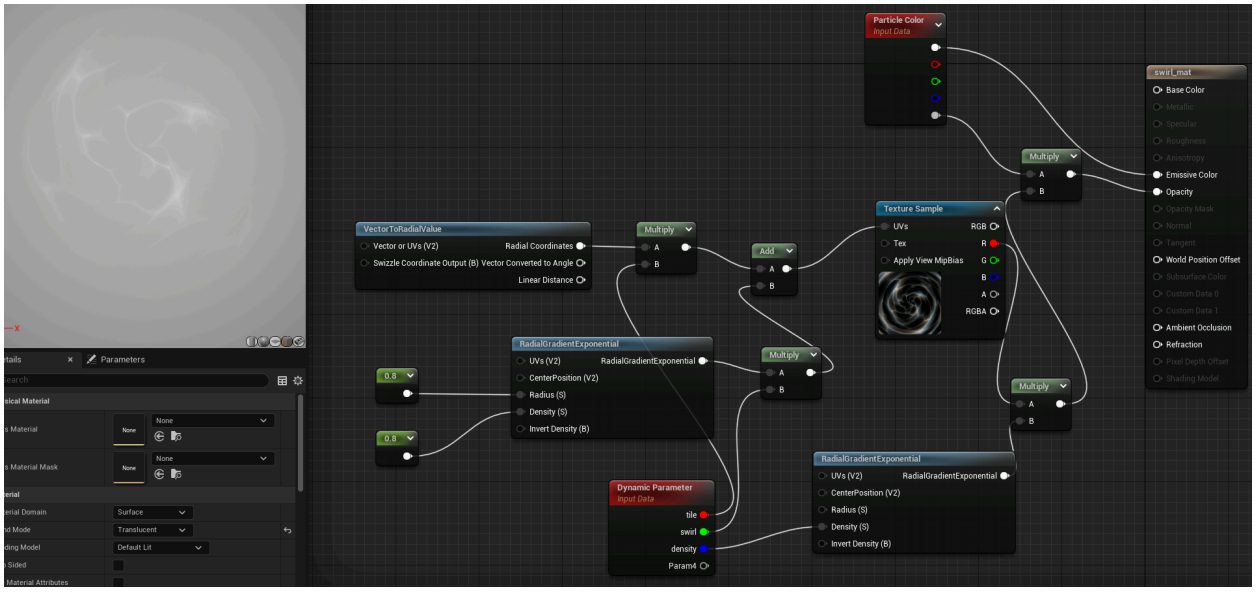
After this, we created another emitter, but instead of using a sprite renderer, we used ribbon renderers and sampled its spawn location to be from the previous particles. Now each of those

individual particles spawn other ribbon particles and together form the niagara for our ghostball.



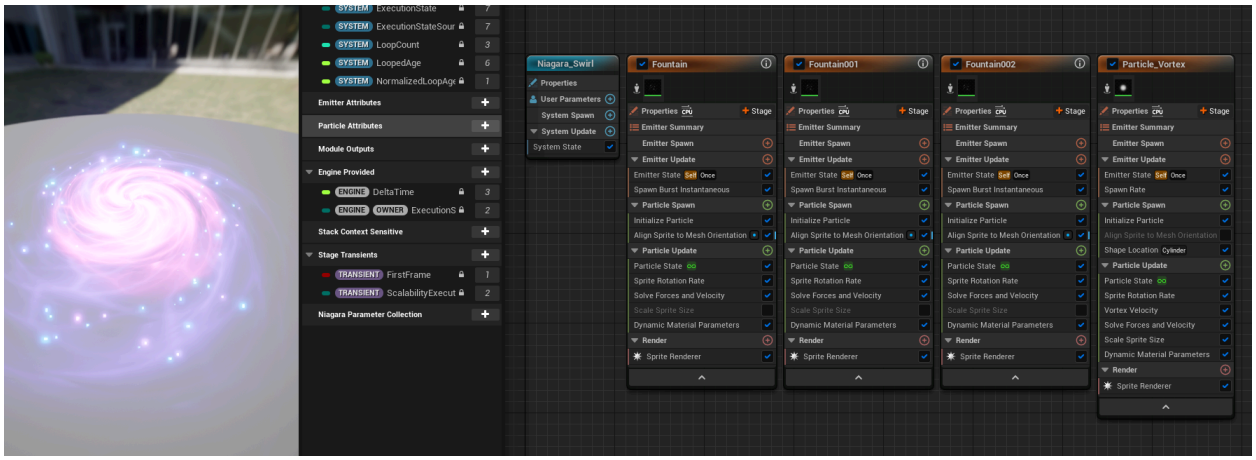
2.6.5 Magnet Hole

The magnet hole, is displayed on top of the hole when a player uses the magnet hole power-up. The niagara is made by first creating a material that looks like a swirl. This is done similar to the fireball material, but rather we plug a node called VectorToRadialValue into the noise texture to create a swirl sort of look. By multiplying various other variables and nodes into the emissive and opacity of the material, we get a swirl texture.



The material is then used in the niagara system, oriented to the xy plane, and added a rotation to create the initial black-hole look. The same emitter is then duplicated and changed colours to

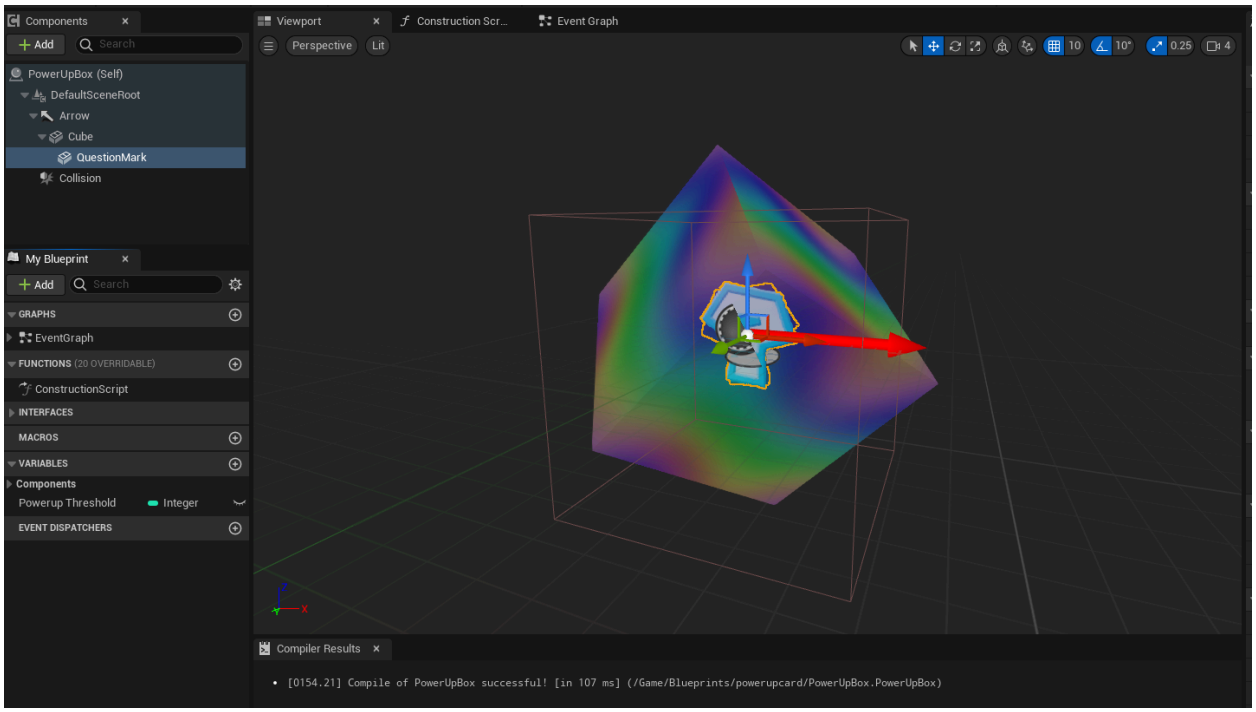
create more layers to the swirl. Finally, particles with varying sizes and colours are added and given a vortex velocity to spin around the existing swirls, creating a convincing black hole look.



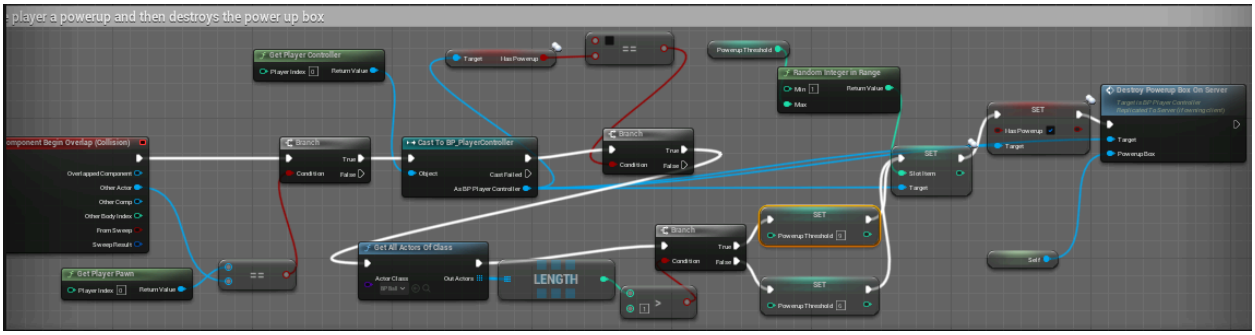
Section 3
Power-ups Implementation

2.7.1 Power-up Box

A user is able to obtain a power-up by hitting a power-up box with the golf ball.

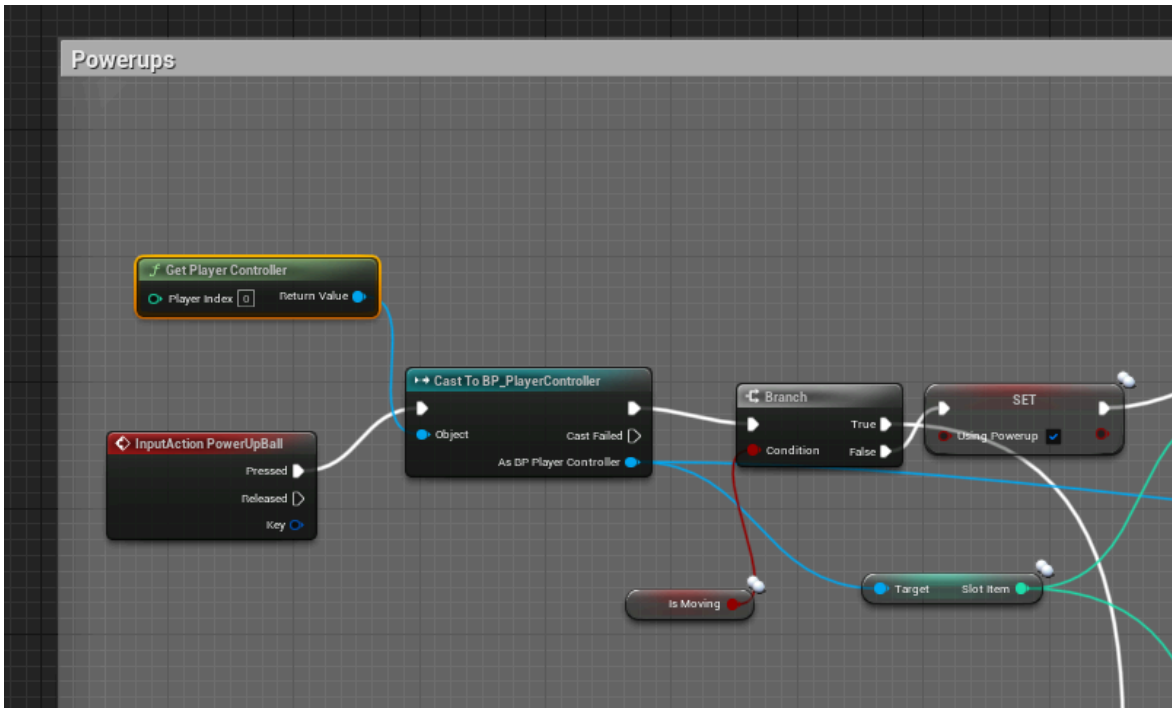


The power up box consists of mainly 3 layers. The question mark texture, the rainbow box itself and the collision box. The rainbow box consists of an opaque seamless rainbow texture so that the question mark texture can be seen inside the box. The collision box is there so that overlap events with the golf ball can be detected.

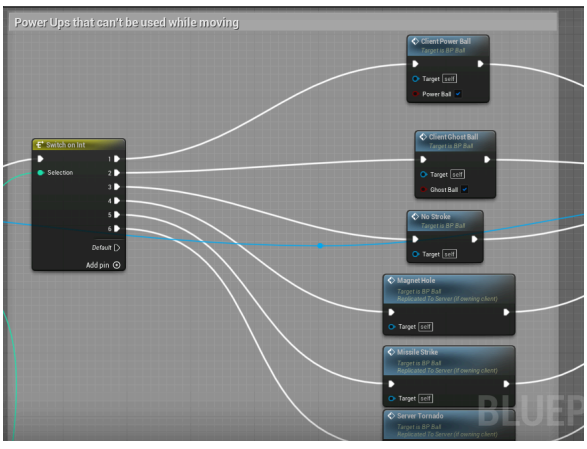


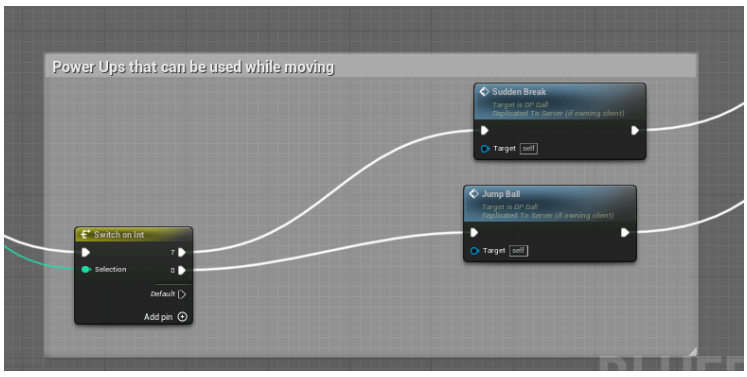
When the power-up box detects an overlap event the code above is executed. The code first checks whether or not the user already has a power-up as the user can only hold one power-up

at a time. If they have a power-up, then nothing happens. However if they don't have a power-up then the "slot item" variable takes on a random integer between 1 - 9 if the gamemode is multiplayer else it takes on a random integer between 1 - 6, with each number representing a specific power-up. Afterwards the power-up box is destroyed.

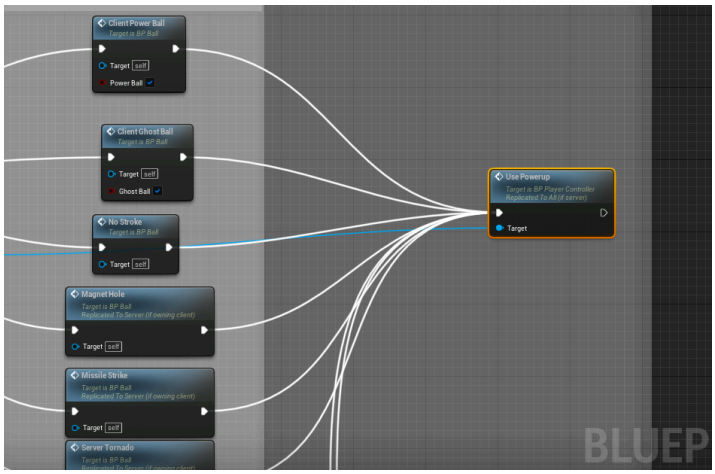


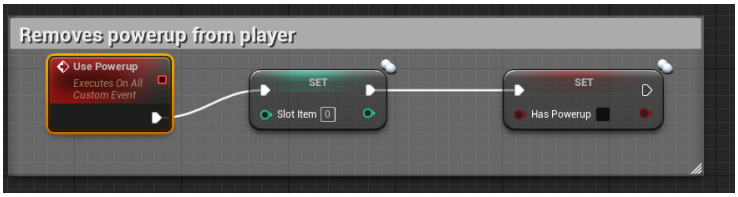
To activate a power-up the user presses the "E" keyboard button. When this is done the code above is executed. It checks if the ball is currently moving and what value the "slot item variable" is and proceeds to execute the respective power-up. The reason why it checks if the ball is moving or not, is because in Goofy Golf there are 2 power-ups that can be used while moving, the sudden break and jump ball power-up. So this check was necessary in order for these two power-ups to be implemented. The custom functions seen in the image below will be further explained further on.





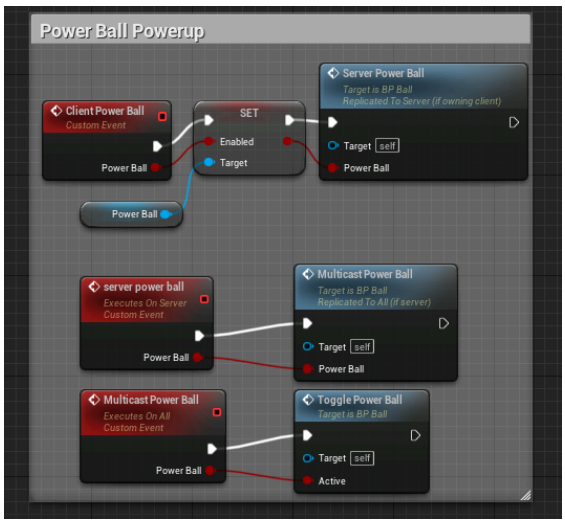
After a power-up is used, it is then removed from the user. The following two images show how this is done.



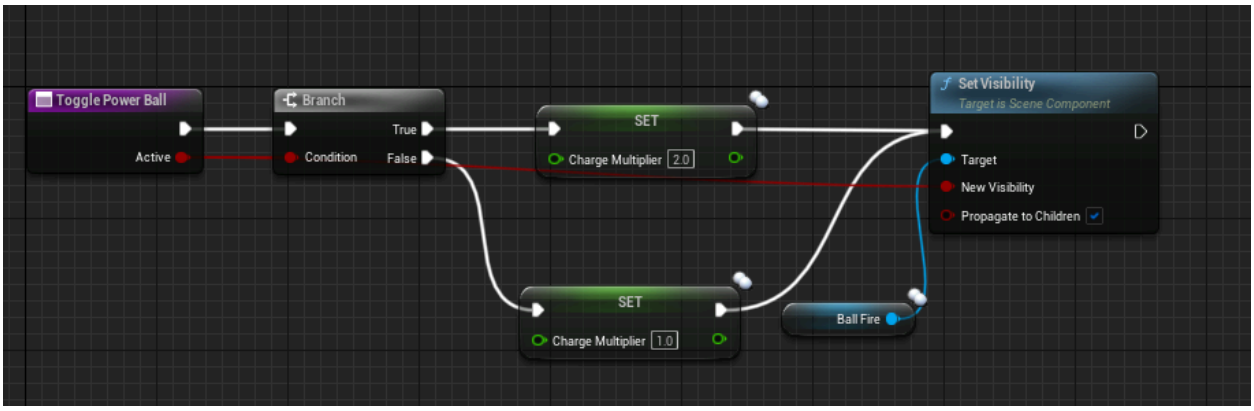


2.7.2 Power Ball

The power ball power-up increases the power in which the user can hit the ball with. The Following 3 images show how it is implemented.



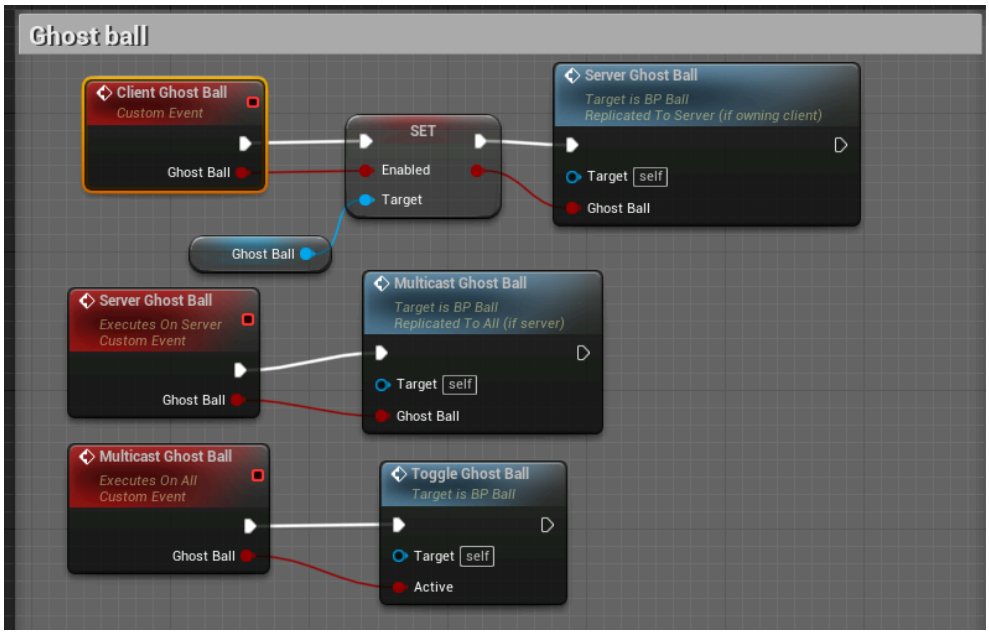
The Server and multicast functions are necessary so that the fireball niagara appears to other users in multiplayer.



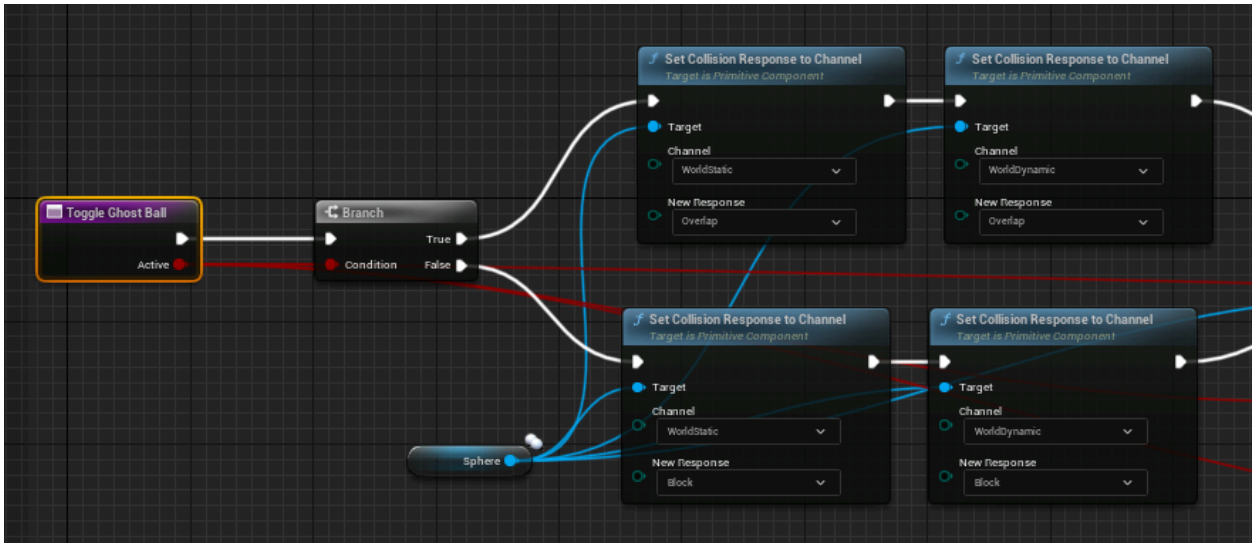
When the power-up is used, the charge multiplayer is increased and power ball niagara is played.

2.7.3 Ghost Ball

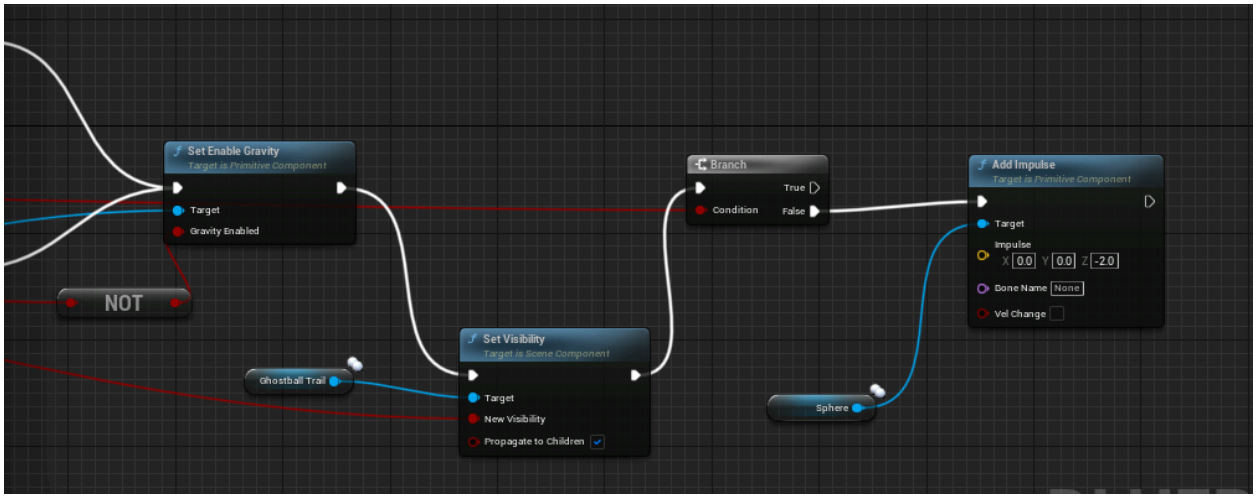
The ghost Ball is a power-up that allows users to phase through walls and obstacles. The following images show how this power-up is implemented.



The Server and multicast functions are necessary so that the ghost ball niagara appears to other users in multiplayer.



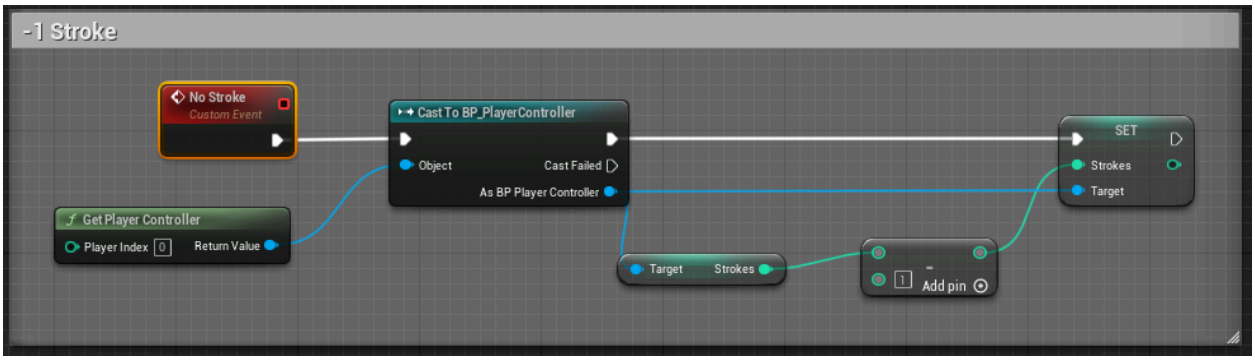
The above code allows the ball to ignore all collisions for world static and dynamic objects.



The gravity is then changed, an impulse is added along with the ghost ball niagara. The combination of these gives the ball a “ghosty” aspect.

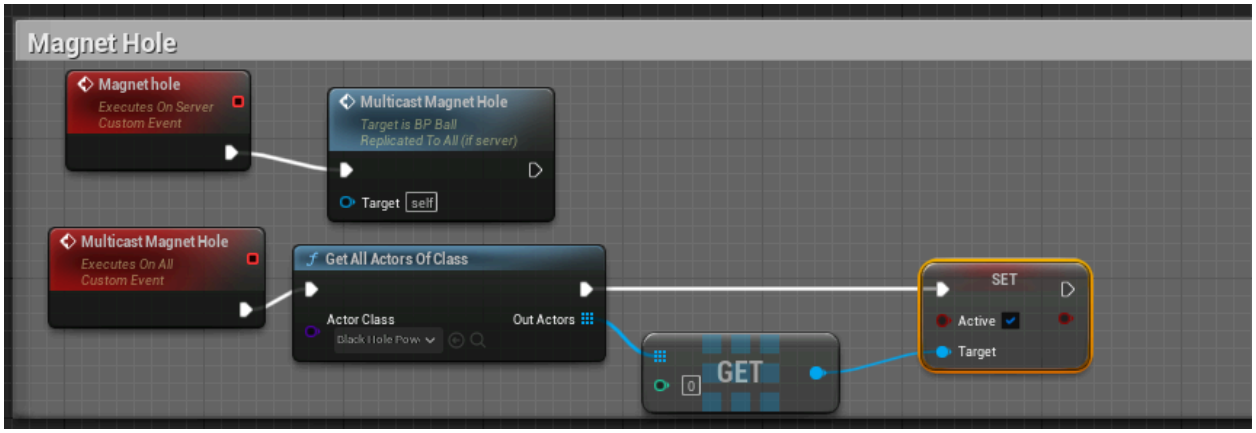
2.7.4 -1 Stroke

The -1 stroke power-up deducts a stroke from the player's stroke counter. The following images show how it is implemented.

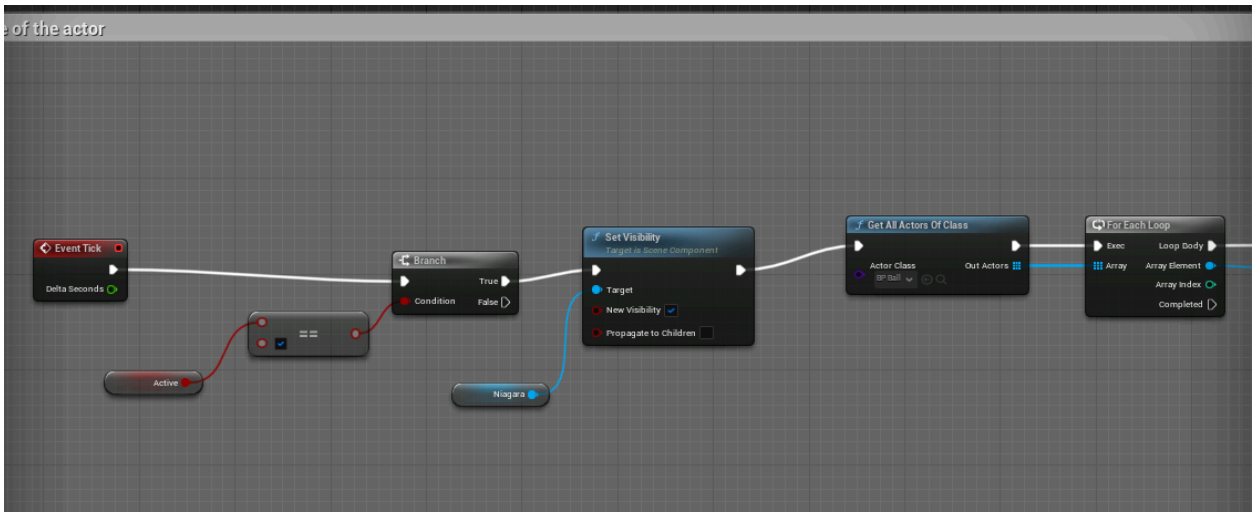


2.7.5 Magnet Hole

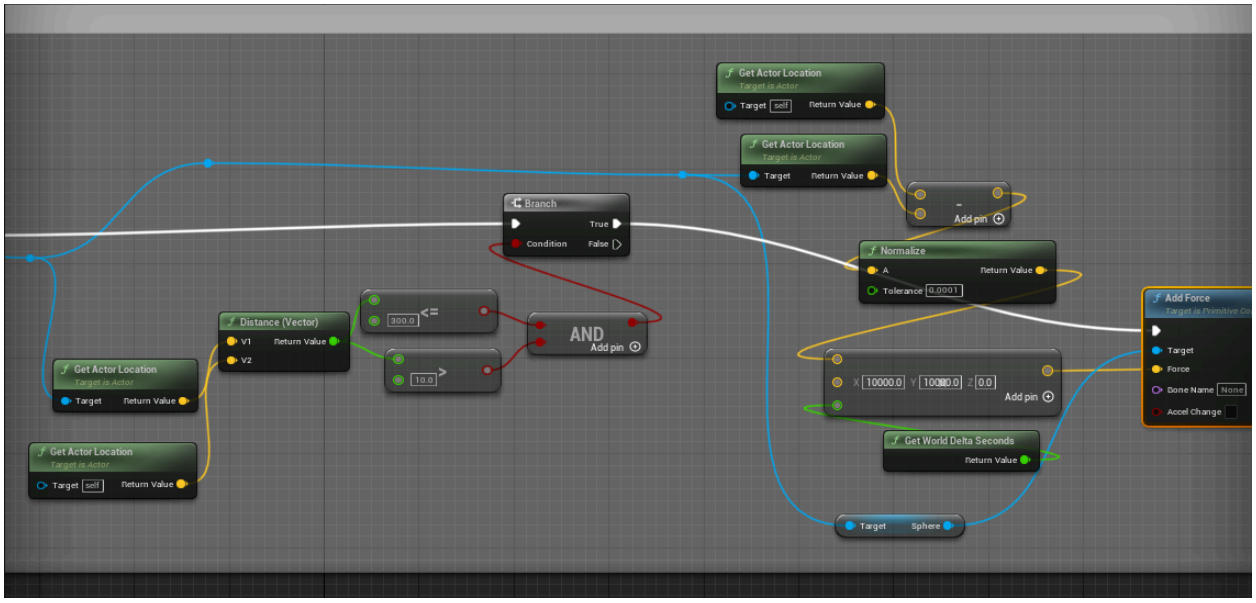
The magnet hole power-up “sucks” golf balls towards the hole, making it easier for them to be putted. The following images show how it is implemented.



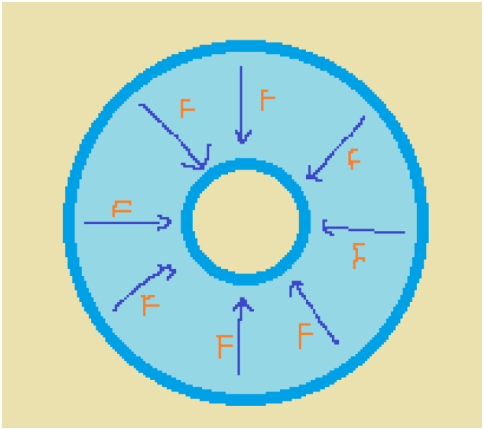
The Server and multicast functions are necessary so that the magnet hole niagara appears to other users in multiplayer. For each of our 3 levels, each hole has a magnet hole actor placed on top of it that stays deactivated until the user uses the magnet hole power-up.

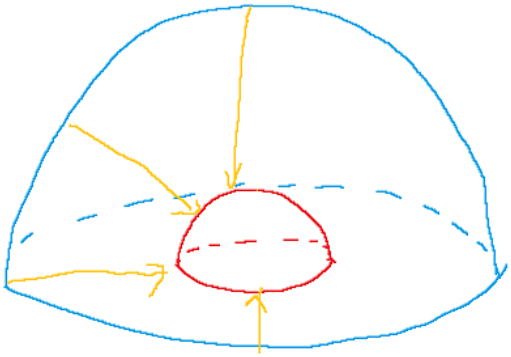


Once the magnet holes are activated then the niagara is set to visible and balls are able to be sucked into the hole.



The code above is the sucking logic. The code applies a constant force that is directed towards the hole, on any golf ball that enters the radius of the magnet hole.

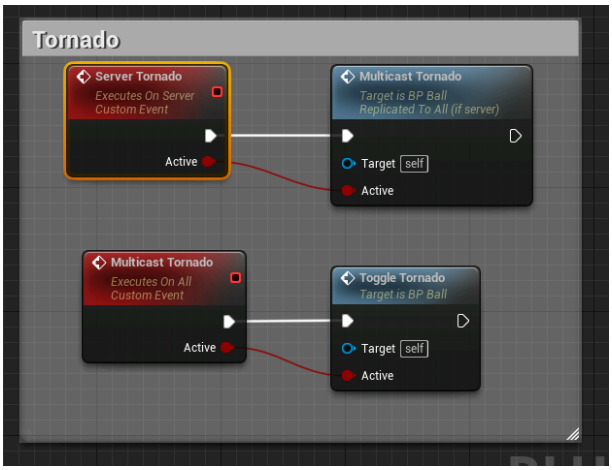




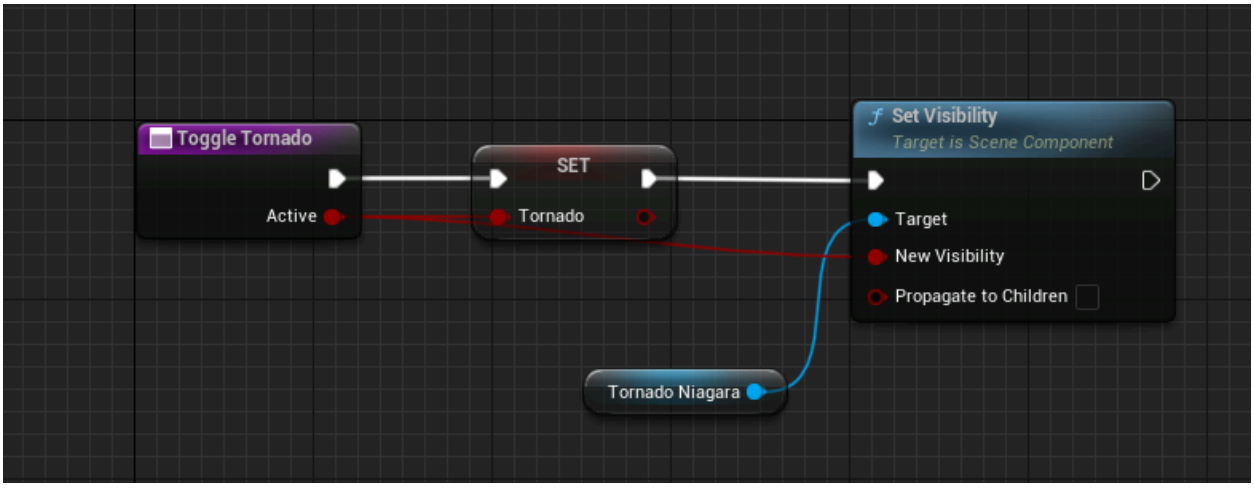
The images above is are a 3D and 2D visual example of how the forces are applied

2.7.6 Tornado

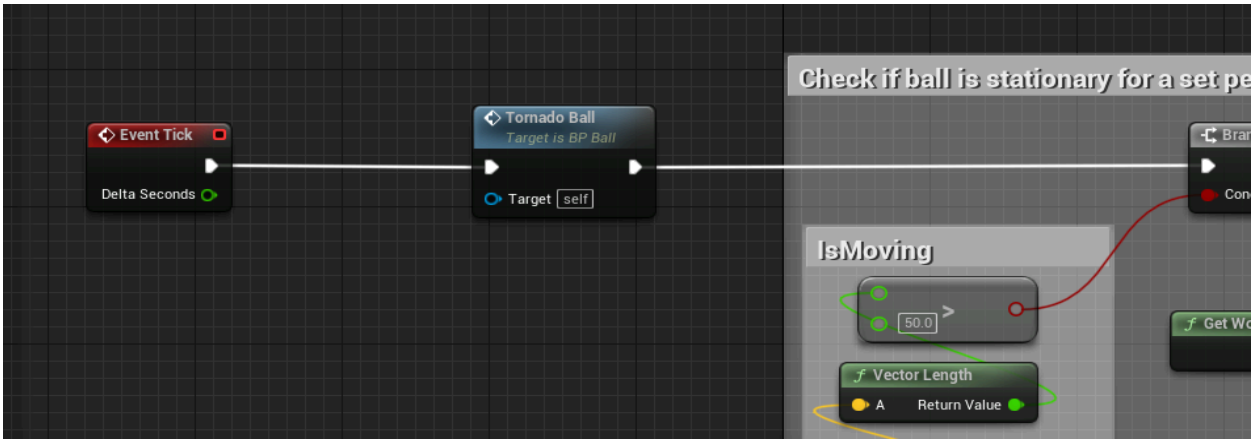
The tornado power-up is a multiplayer exclusive power-up that summons a tornado on the player's ball which sucks and flings any other balls that come close to the tornado. The following images show how it is implemented.

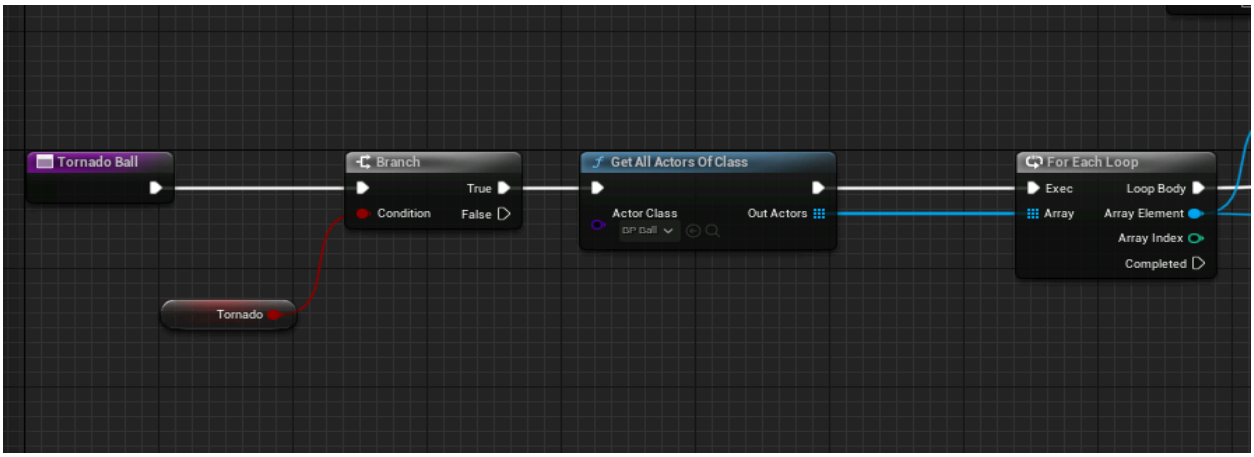


The Server and multicast functions are necessary so that the tornado niagara appears to other users in multiplayer.

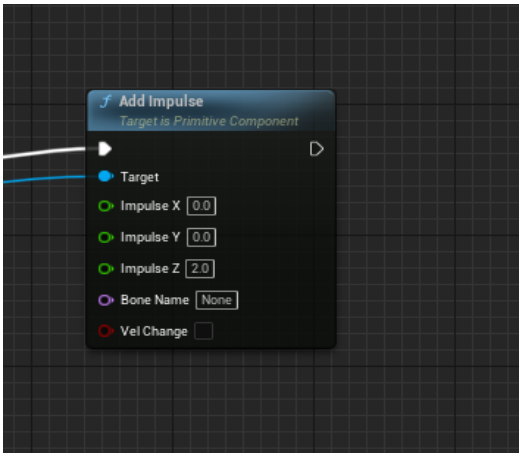


The tornado boolean is then set to true and the niagara is set to visible.



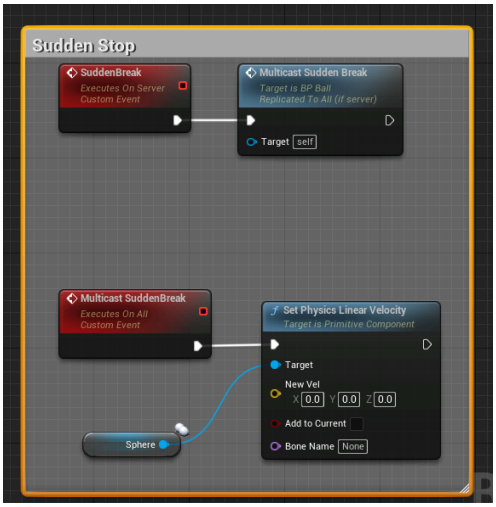


By setting the tornado boolean to true, other golf balls are now able to interact with the tornado. The logic for the tornado to suck other golf balls is the exact same logic as the sucking logic present in the magnet hole. The only difference is the addition of an upwards impulse force to simulate the tornado “flinging” the other golf balls away.



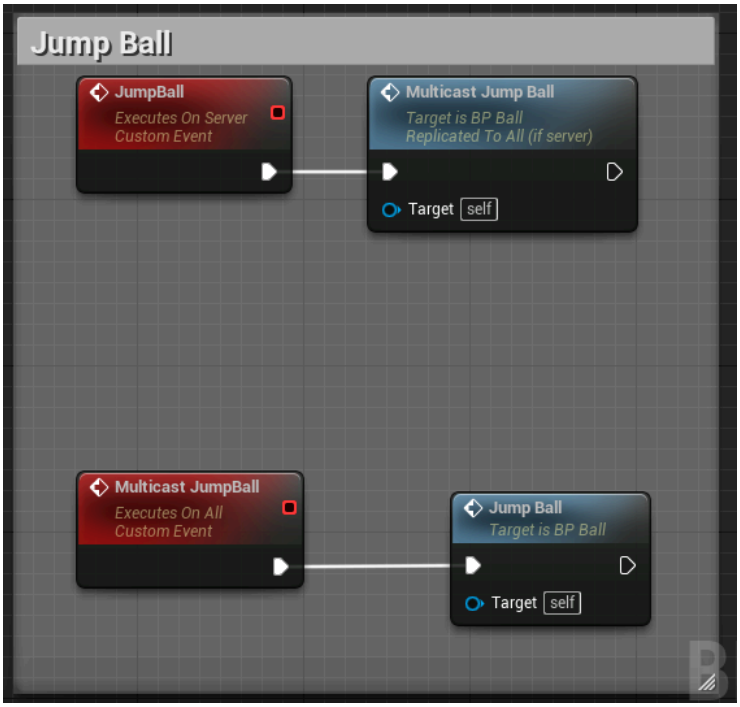
2.7.7 Sudden Break

The sudden Break power-up is a power-up that instantly sets the ball's velocity to 0 when used. The following image show how it is implemented

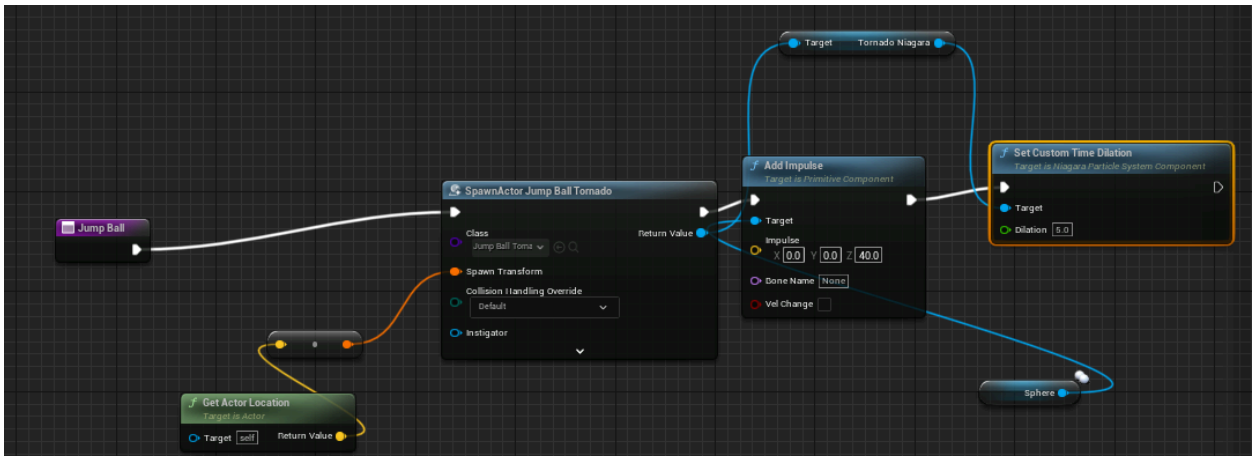


2.7.8 Jump Ball

The jump ball power-up is a power-up that summons a tornado to make the ball "jump" when activated. The following images show how it is implemented.



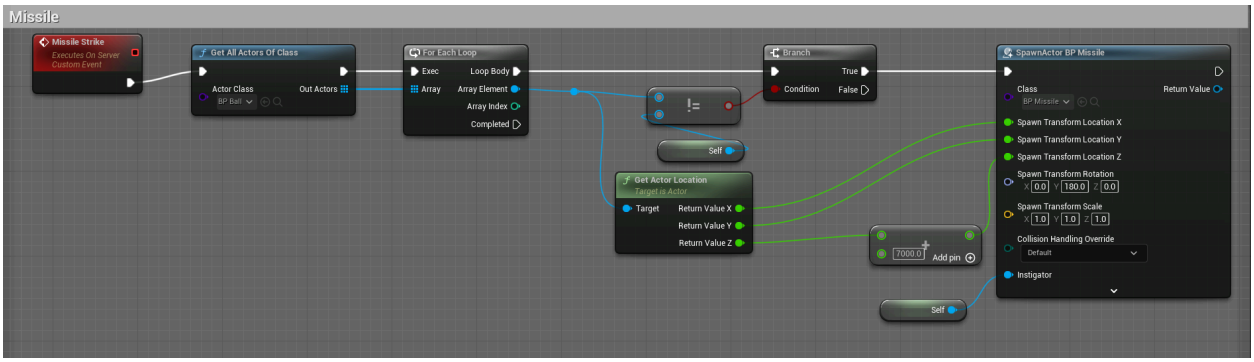
The Server and multicast functions are necessary so that the tornado niagara appears to other users in multiplayer.

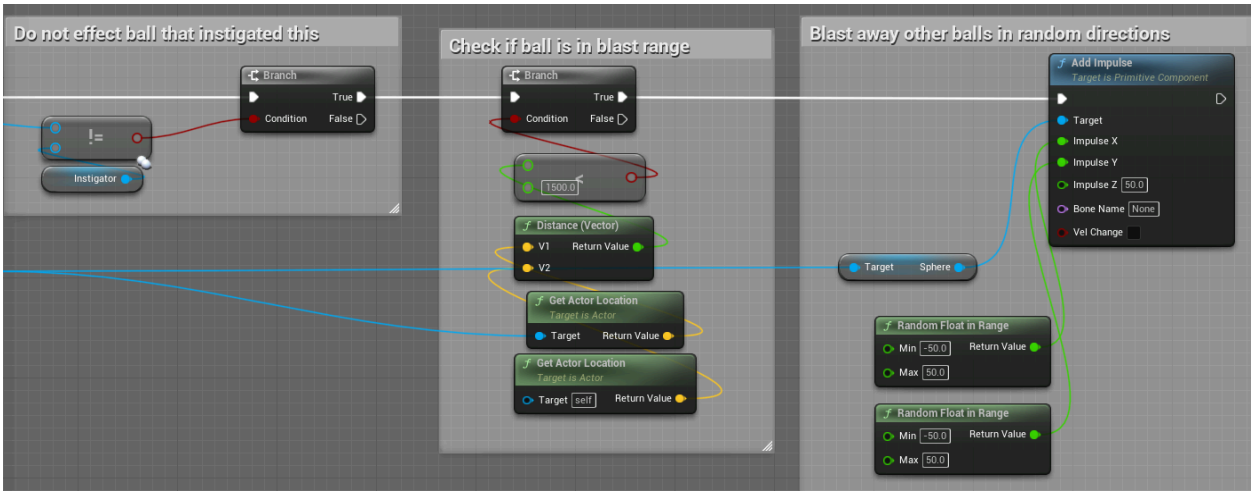


The above code spawns the tornado niagara and adds an upward impulse to the ball. The time dilation node is used to speed up the playtime of the niagara to make it seem like the tornado is lifting the ball away.

2.7.9 Missile Strike

The missile power-up spawns missiles above all players except for the player calling the missile strike (instigator). Upon impact, the missile explodes launching any players, excluding the instigator, who are within the blast range in a random direction. This is achieved by the following blueprint.

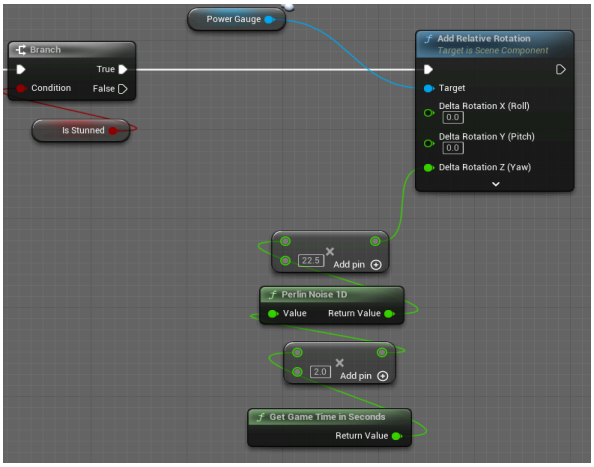


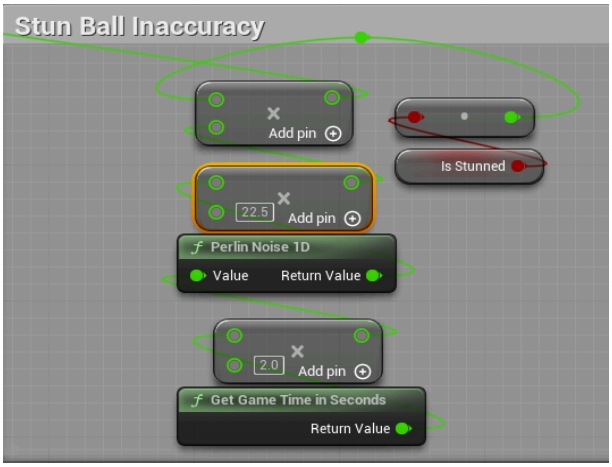


2.7.10 Stun Ball

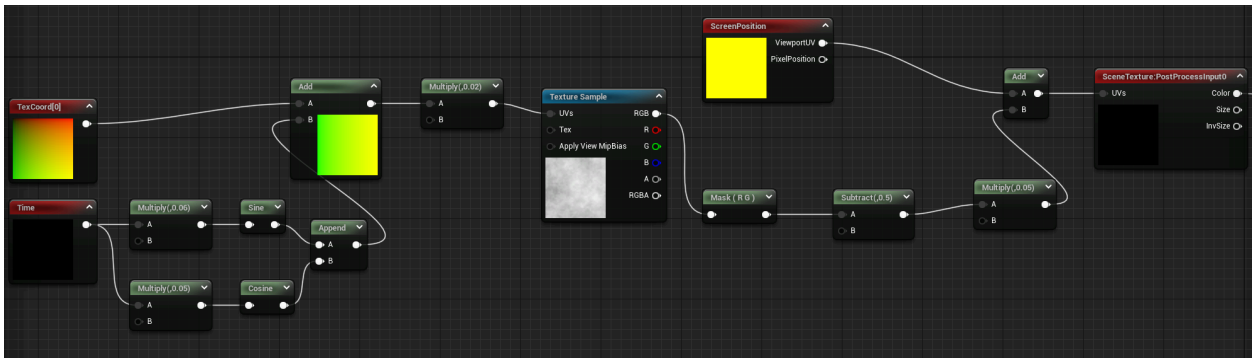
The stun ball power-up causes all players except the instigator to incur a massive shooting accuracy penalty whilst the screen becomes distorted and becomes more monochromatic.

The accuracy penalty is administered via the following code.





And the screen distortion is achieved via a post processing effect.



For both distortion and inaccuracy calculations, perlin noise is the preferred source of randomness as it allows for smooth transitions which will greatly enhance user experience.

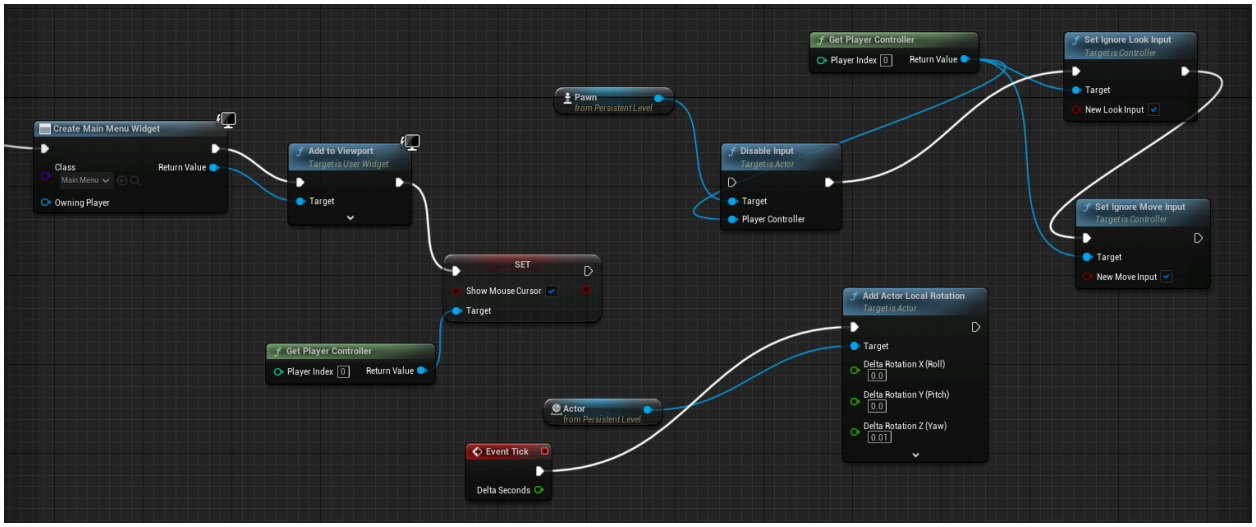
2.8 Planned Features that weren't implemented

In our project proposal there were features that we planned on implementing in our system. However due this being the first time group members have interacted with Unreal Engine we severely underestimated the time, effort and complexity regarding implementation of features. One such feature was the implementation of the golf ball which took a significant amount of time and effort. Furthermore it was also decided that some features did not fit the "goofy" aspect of our system. As a result other features that were better suited to our circumstance were chosen to be implemented. A list of features that were mentioned in the project proposal that were not implemented are

- Co-op mode (multiple users trying to putt one ball)
- Heavy Ball
- Blind putt (Effect merged with stun power-up)

2.9 Regular menu system

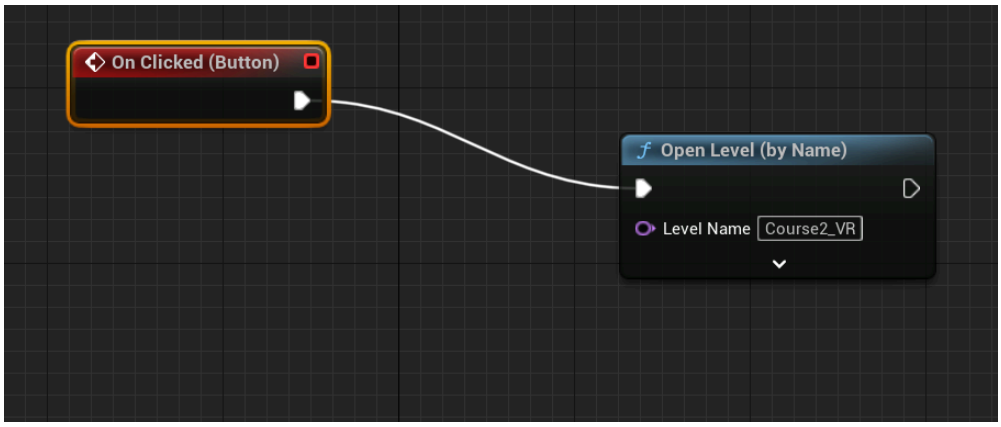
There are 3 menus in the game that help the user navigate between stages. The first of these menus is the main menu, essentially the main menu has the first level in the background with a custom main menu widget overlaid on the players viewport. We also ensured the user could see their cursor so we set "Show Mouse Cursor" to true. We also had to make sure that when the user moved their mouse cursor this didn't move the background so we ignored look input. All the other menus in the game use the exact same functionality to allow users to use their cursor to select options.



The main menu widget itself is quite simple, it has three buttons, "Singleplayer", "Multiplayer" and "Quit". The singleplayer button loads the stage selection screen, where the user can select between the three stages available by clicking a corresponding button that loads the level. If the

user selects multiplayer the multiplayer create session screen is displayed (which will be further discussed in the Multiplayer section of this report). If the user selects the quit button the game is closed.

The functionality for loading other levels simply uses the open level by name blueprint module. One such example from the stage select screen can be seen below:

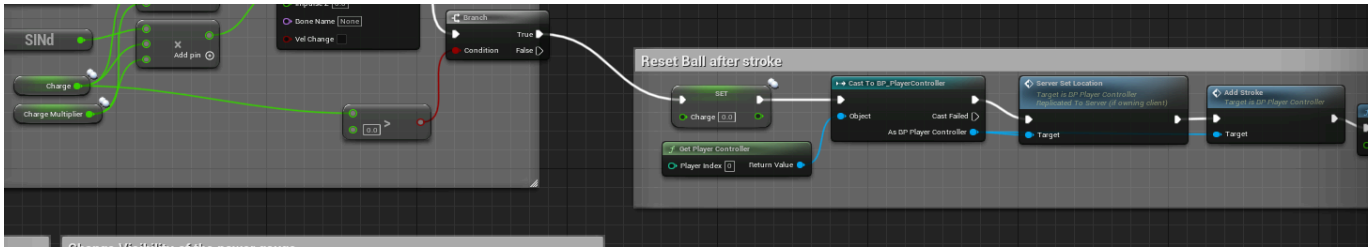


The pause menu is activated upon pressing the escape key and has the options “continue” and “quit” and works similarly to the buttons in the main menu.

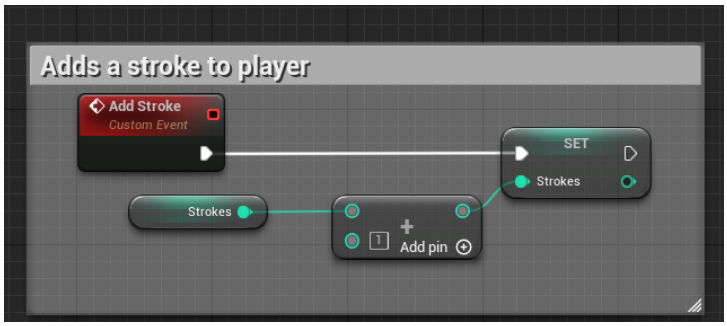
3.0 Stroke counter

To count strokes whenever the ball is about to be launched (when the power bar is on the screen), we check if the charge given to the ball is greater than 0. We need to check this because if the charge = 0 and we add a stroke this will add a stroke every time the user left clicks on the ball because this activates the charging of the power gauge and when they release the button that launches the ball, although if the charge given is zero since the user did not pull back the mouse we don't want this to count as a stroke as the ball will not move.

If the charge is greater than 0 we get the current player controller and call the add stroke custom event which adds 1 to a variable in the player controller which keeps track of the strokes. This is used by the HUD and course complete screens to keep the user informed of the number of strokes they used.



Inside bp_ball blueprint

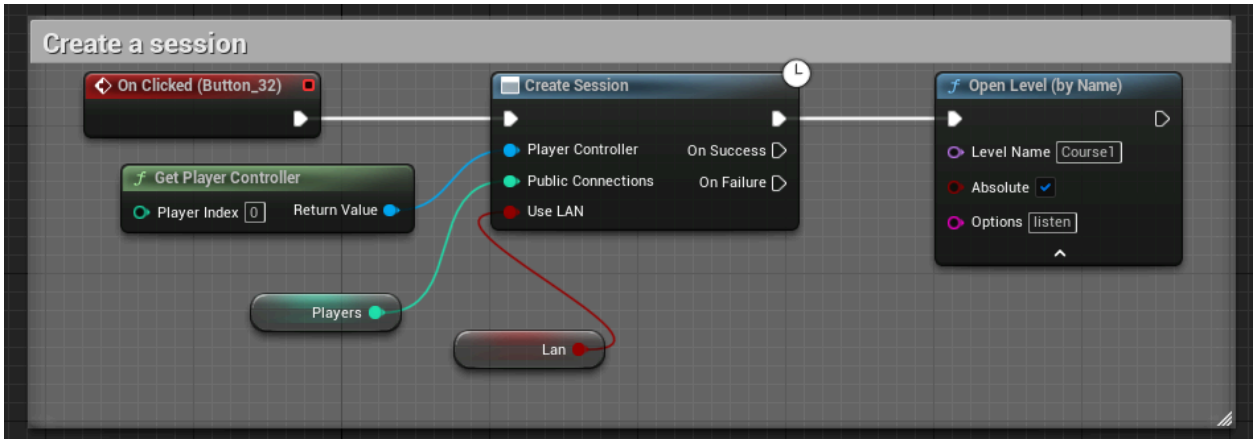


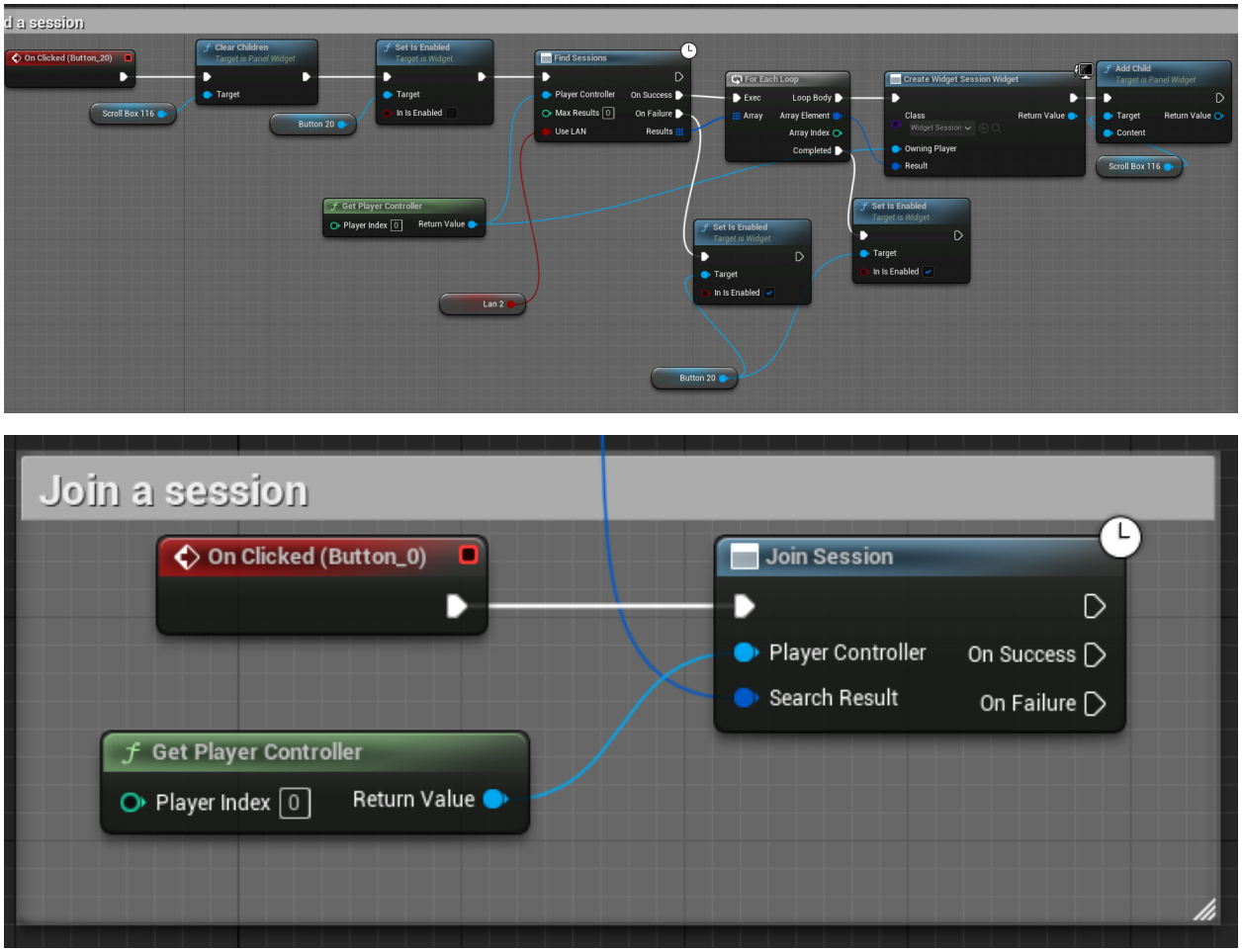
Inside bp_PlayerController blueprint

3.1 Multiplayer

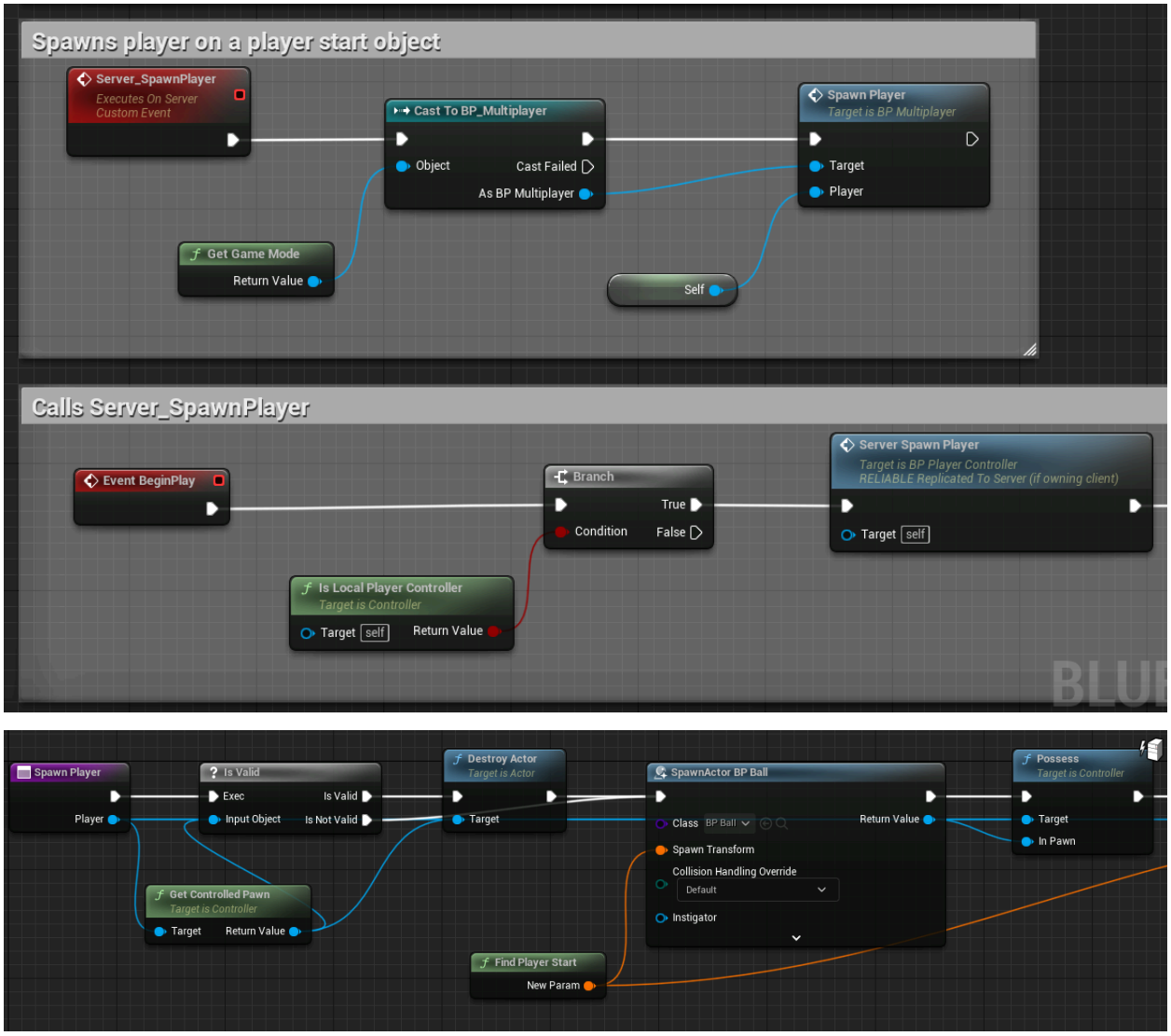
Multiplayer was probably the most challenging feature we implemented into our system. Syncing up balls and physics from the host and the clients were difficult and required a lot of self-learning on how to handle server and client side replication.

To begin, in our main menu, we used Unreal Engine's built in session creation nodes to create and join sessions. The create session function requires the number of players and a boolean for LAN, which when provided creates a session for us. After that, we use the find session function to search for sessions. After a session is found, we use the join session function to join the host's session based on the results provided by find session.

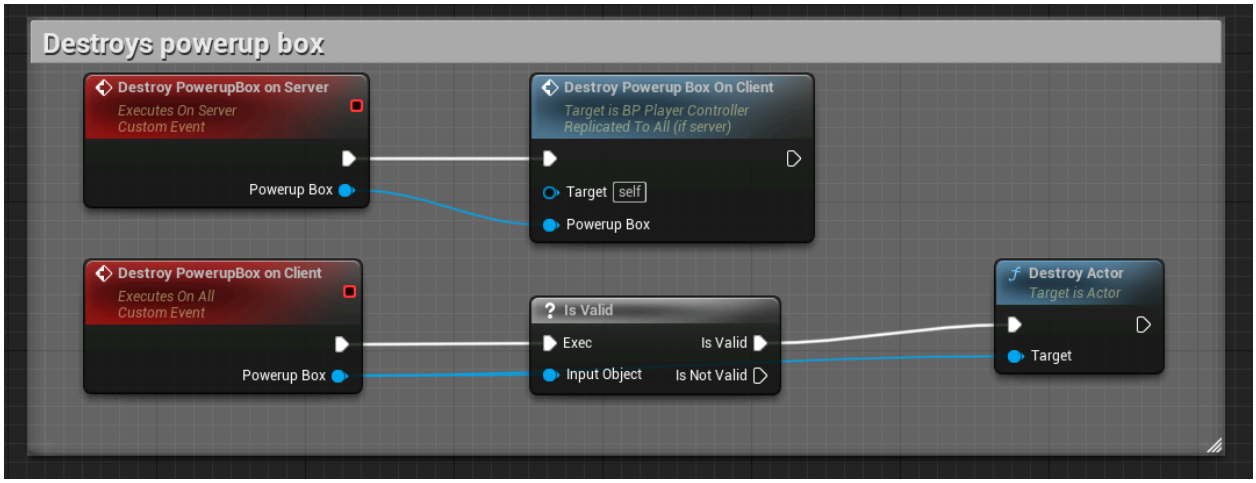


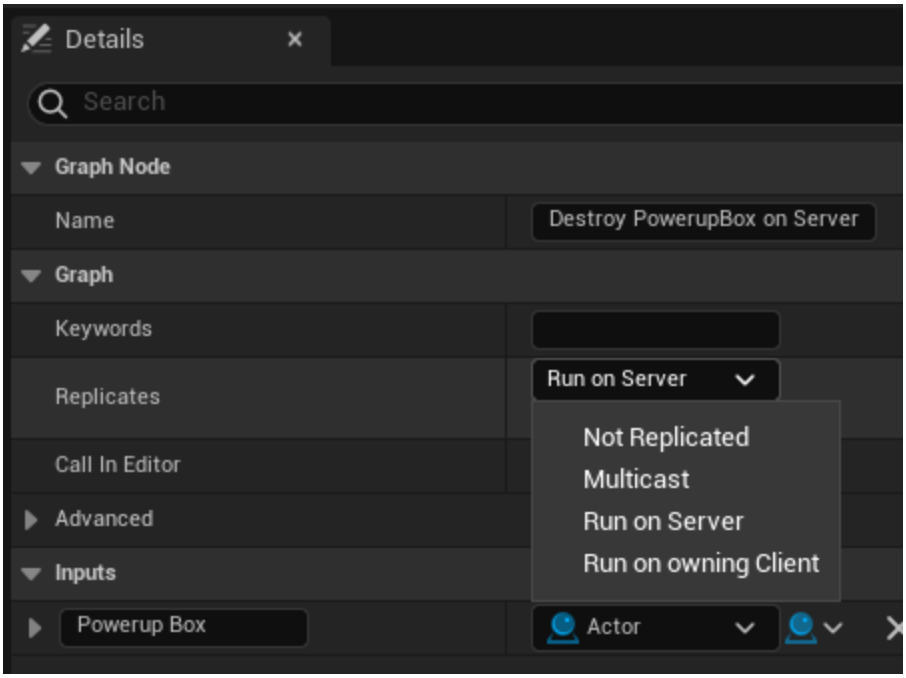


From implementing multiplayer, it is then natural to implement a way to spawn players dynamically rather than simply having a player pawn inside the level. To do this, we used player start objects, and had the server spawn each player in as they joined the session. Each player will be assigned a player controller class blueprint from the server, and from there, the player controller blueprint spawns the player on one of the player starts in the level.



Afterwards, we ran into a problem where the balls of each player would not be in sync with each other, causing balls in the client side to be unable to move. To fix this, we downloaded a plugin called physicsSync which synced the physics of the ball with the server.
More problems then arose such as power-up boxes not breaking on client side, and animations/niagaras not being played client side and vice versa. To solve this, we learnt about replication inside Unreal Engine, and applied it to every mechanic which required server and client interaction. Going forward, we had to make sure to call events on the server side first, before telling the server to call the same event on every client. To do this, we simply have to change the replication method on custom events, for example.

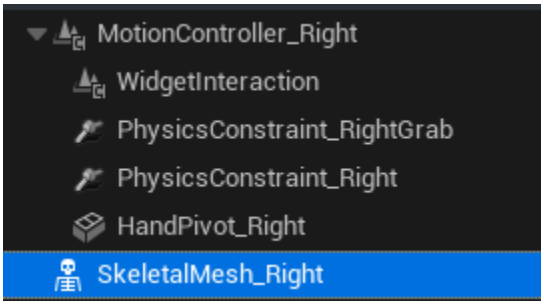




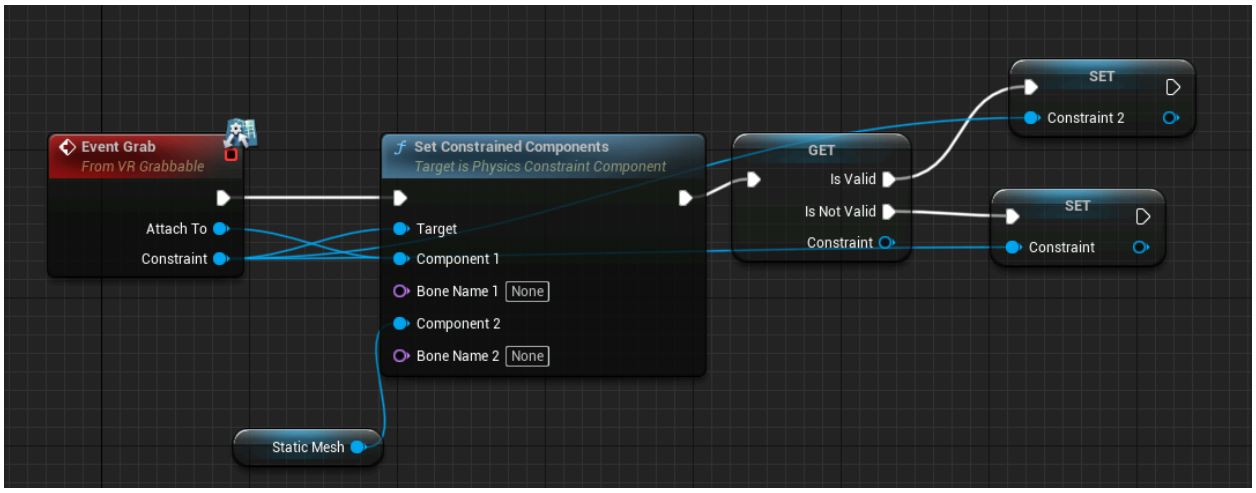
3.2 VR

Implementing VR was not so easy as remapping the player's movements to motion controllers. We first had to redesign the core movement of the player as the perspective is shifted away from that of a third person ball, to a first person perspective of a human. The premise of VR was to allow the player to walk around and hit the ball with a golf club, rather than simply controlling the ball with controls. We did this by giving the ability for the player to walk around

in VR, as well as moving their hands and grabbing objects such as the golf club. However, this was not enough, as the hands along with grabbed objects do not simulate physics, causing a loss of immersion as the golf swing and hit do not feel like it should in the real world. To solve this problem, we implemented physics based hands and object grabbing in our system. For this, instead of having our hand skeletal mesh as a child of the motion controllers, we instead use a physics constraint component to attach the motion controllers with the skeletal mesh. This allows the skeletal mesh to simulate the feel of having weight on your hands when grabbing objects.

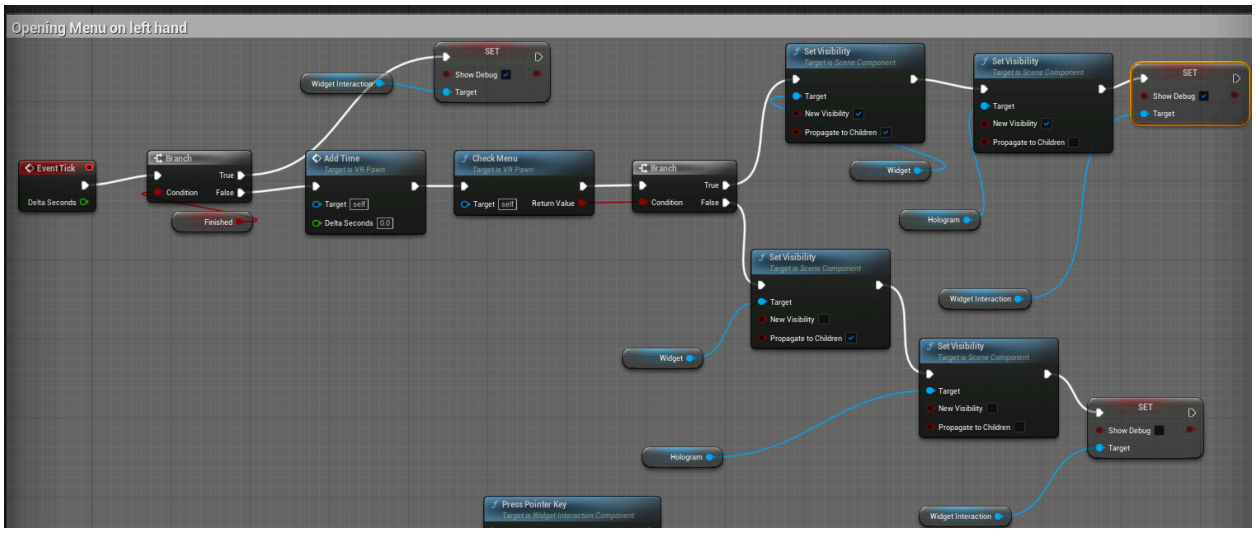


Grabbing objects works in a similar way, by attaching another physics constraint from the motion controller to the object that is being grabbed. This grabbed object has weight and thus is added onto the weight of your hands, causing your skeletal mesh to move not 1 to 1 with your motion controllers but slower depending on the weight you have grabbed. This along with the weight of the golf club, simulates a more accurate feel and swing of the golf club, as you can now feel its weight as you move it around.



In addition to the movement getting a rework for VR. The HUD elements such as your score and power-ups also had to be adapted for VR. As it is generally bad practice for elements to be on

the screen and in your face in VR, we had to place the score and power-up elements into a more subtle position. We designed a holographic screen that shows up when the player turns their palms facing toward the ground and looking slightly above the hand. The holographic screen shows the player's score and power-up, and is manually turned off when the player is not looking. This provides a more immersive way for the player to view their score, while it being non-intrusive to their vision at all times.



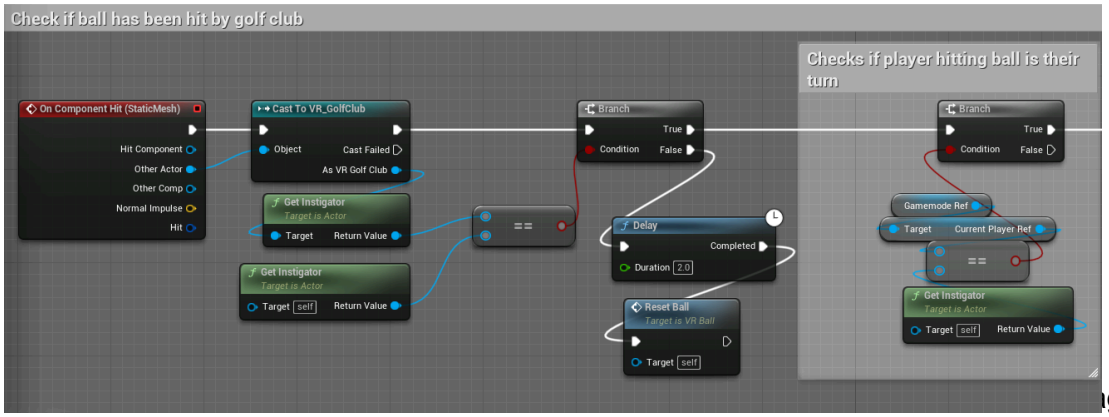
3.2.1 Main Menu

We also had to make a new main menu exclusive for VR as the player is no longer just staring at a screen, we had to create an entirely new environment for the player to immerse themself in before starting a game. We did this by simply reusing assets used in our courses and creating a

small level for players to start in. In the middle of the islands are stationary buttons that players can point at with their virtual hands and click on. They are the main menu buttons such as play and level select, but just in a 3D environment.

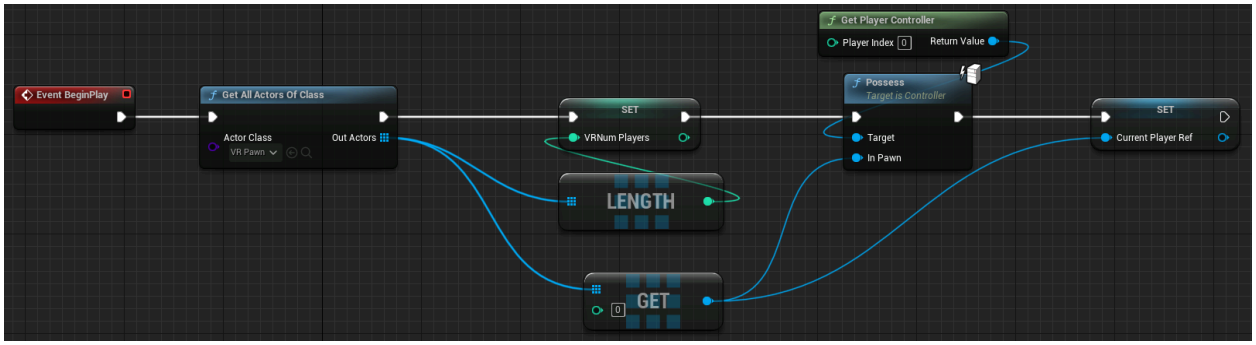
3.2.2 Changes to the Ball blueprint

In order to accommodate VR, many aspects of the ball that only makes sense when viewed from a monitor have been removed such as the power gauge greatly simplifying the blueprint. Instead of the charging mechanic, we now test if the golf club that is hitting the golf ball belongs to the player currently taking their turn and if it isn't we simply respawn the ball that was accidentally hit at its original location after 2 seconds.

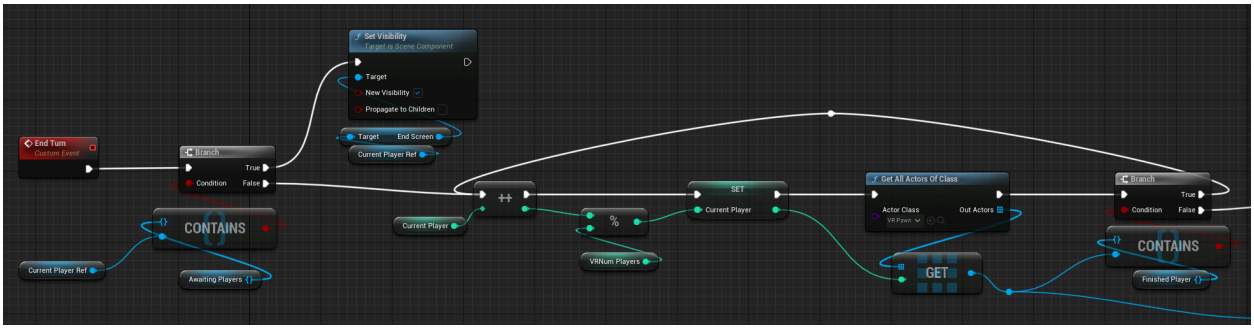


3.2.3 Passing the headset mechanic

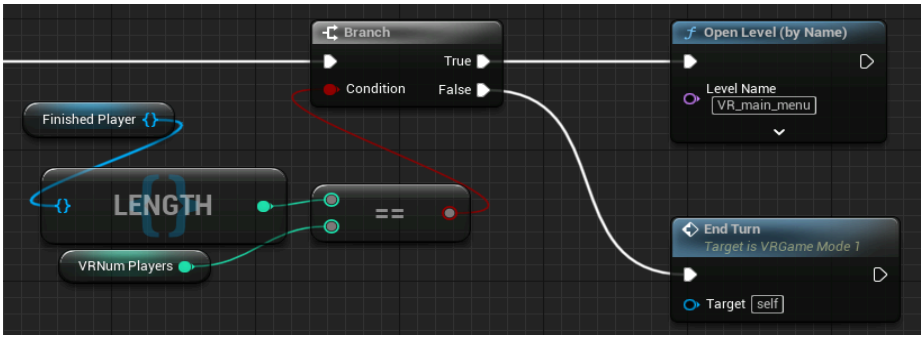
After much deliberation, we have decided that passing the headset will be the preferred method of implementing multiplayer in VR as LAN VR is entirely out of the scope of this course. Upon starting the level we obtain an array of references of all the players that are participating in the game.



We then continuously loop through all players that are still in the game until they have all potted their balls and skip any players that have already completed the course.
Upon completion of the level a level complete will display on the headset prompting the player to click continue and passing the headset on to the next player.

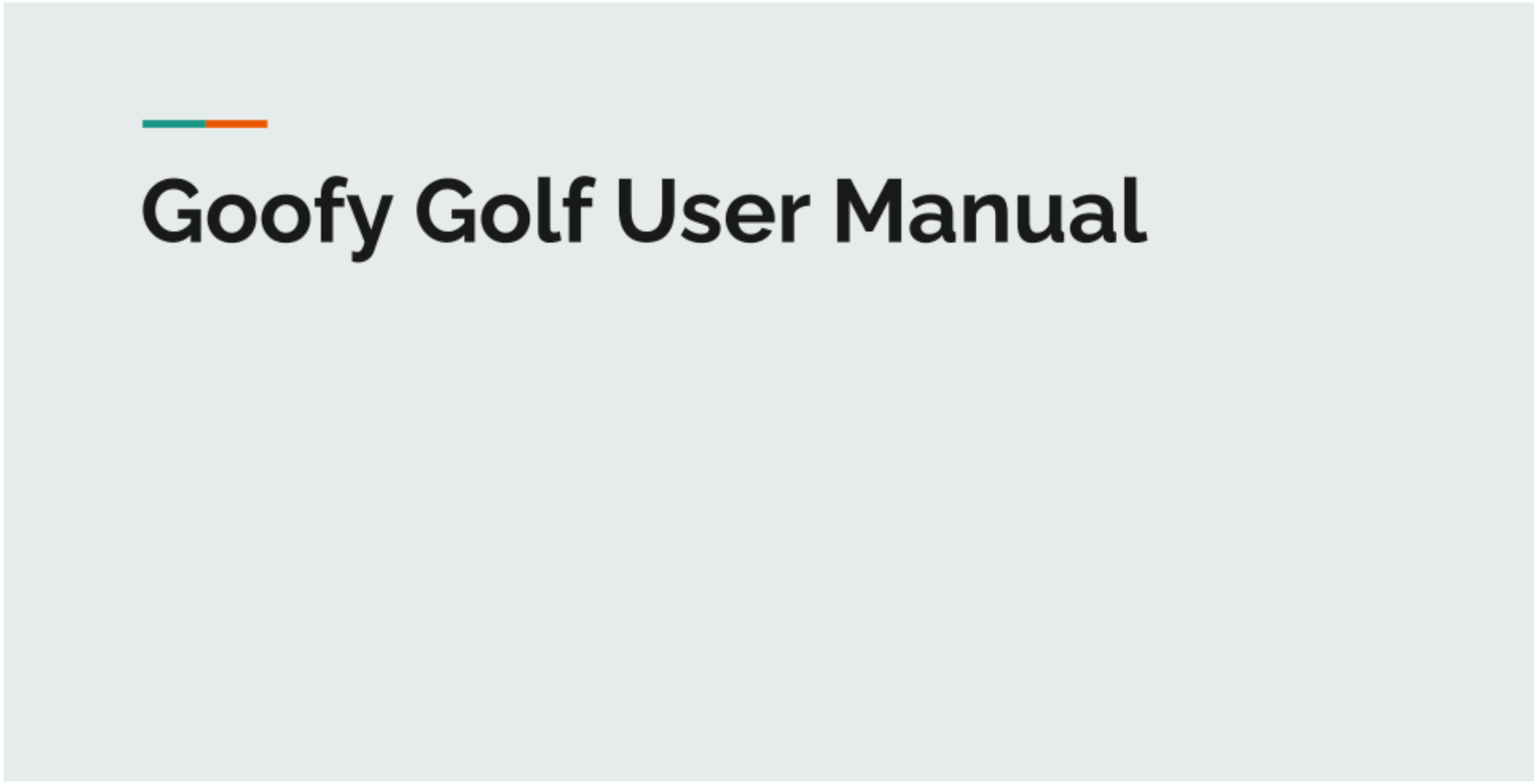


The game will return all players to the main menu after all players complete the course.



Section 4

User Manual



Important Demo notes

The following 3 slides are exclusively for demo purposes

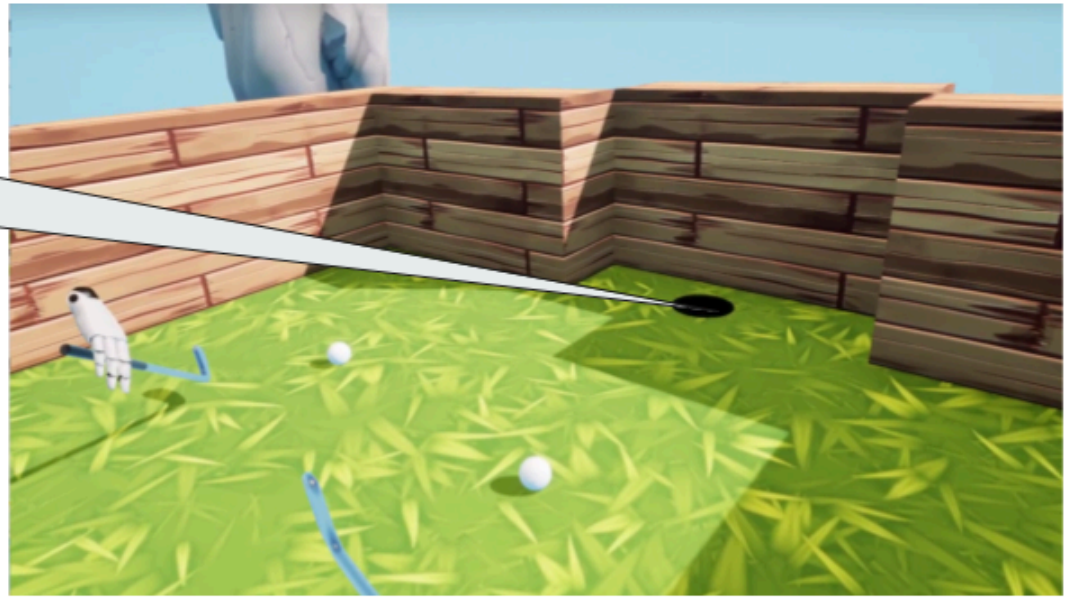
Cheats for Debugging and Demo

Press keys 1-9 on the keyboard to instantly give yourself a powerup

Cheats for Debugging and Demo

Hole is provided as a convenience for testing and demoing purposes.

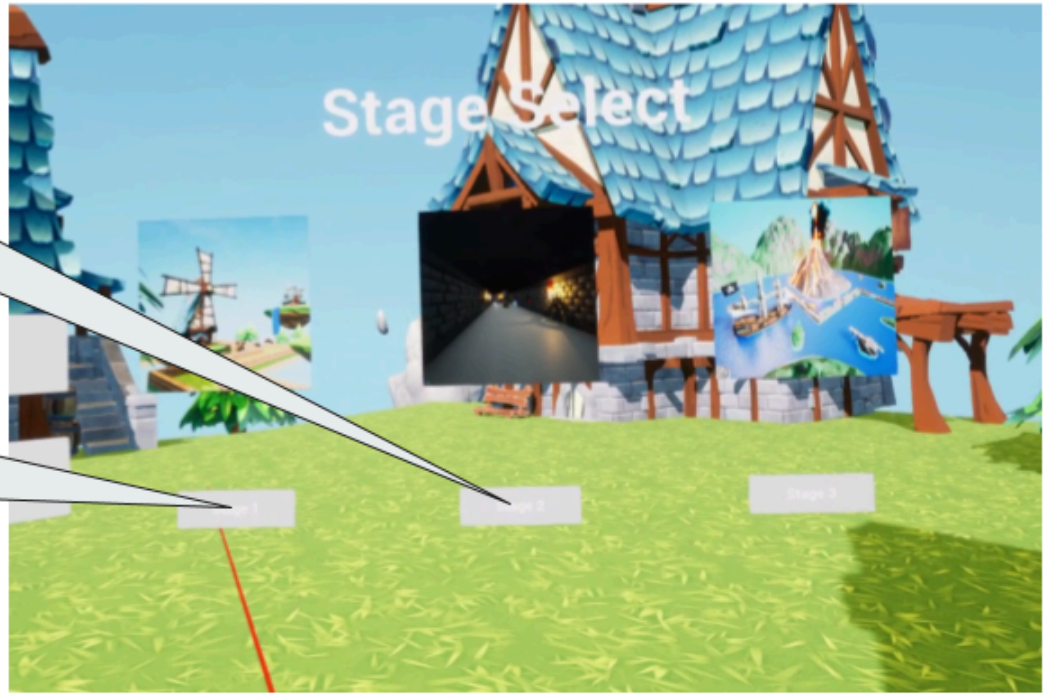
Please note that black hole powerup will only work on the “true” hole and not this one



Cheats for Debugging and Demo

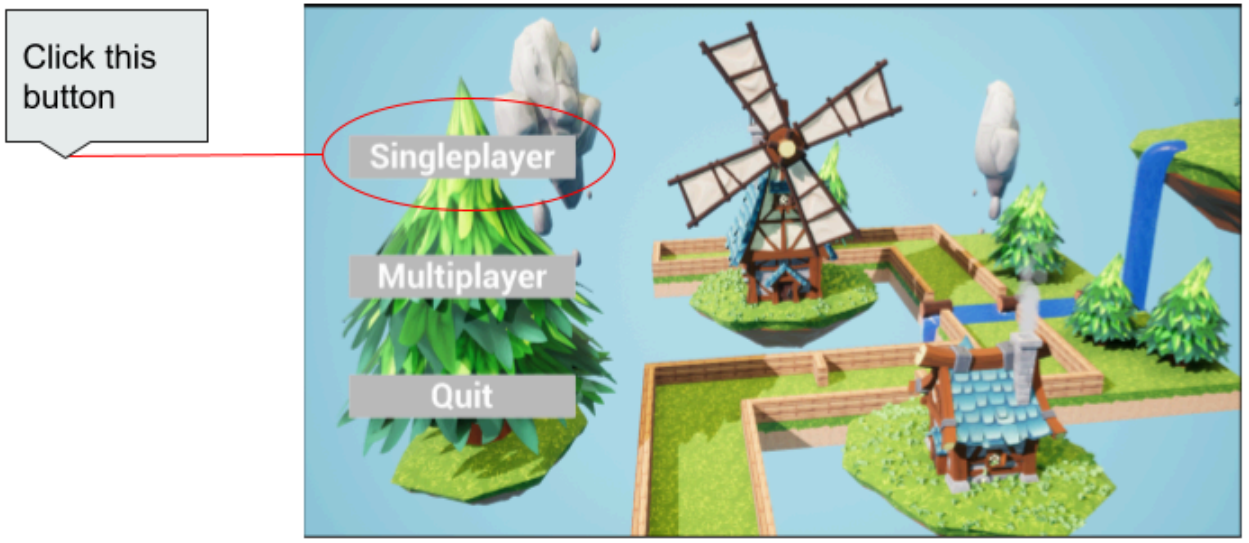
Play the second level to test for single player

Play the first level to test for multiplayer

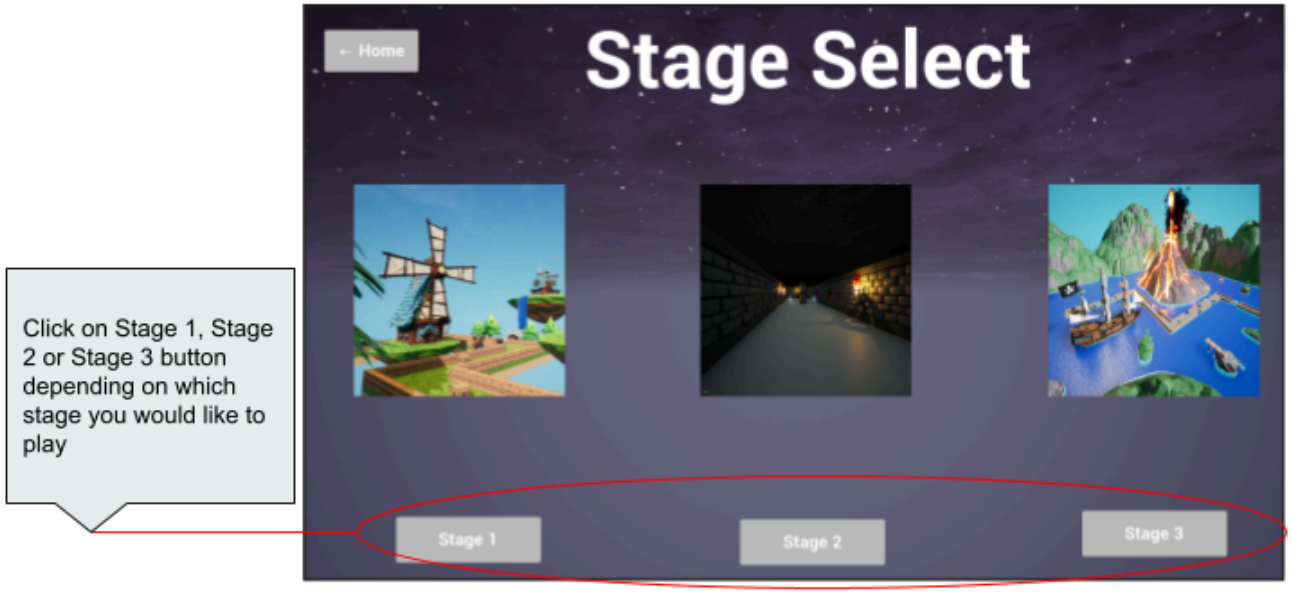



Desktop Version

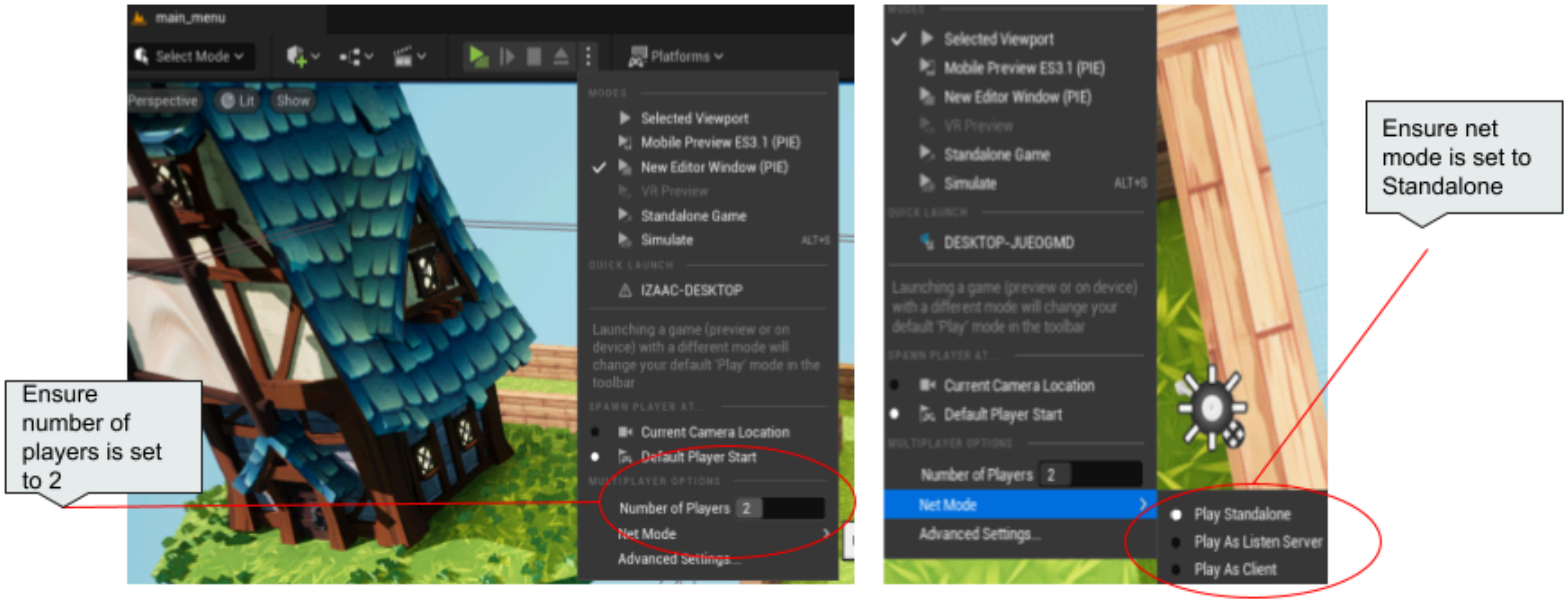
Starting a singleplayer game



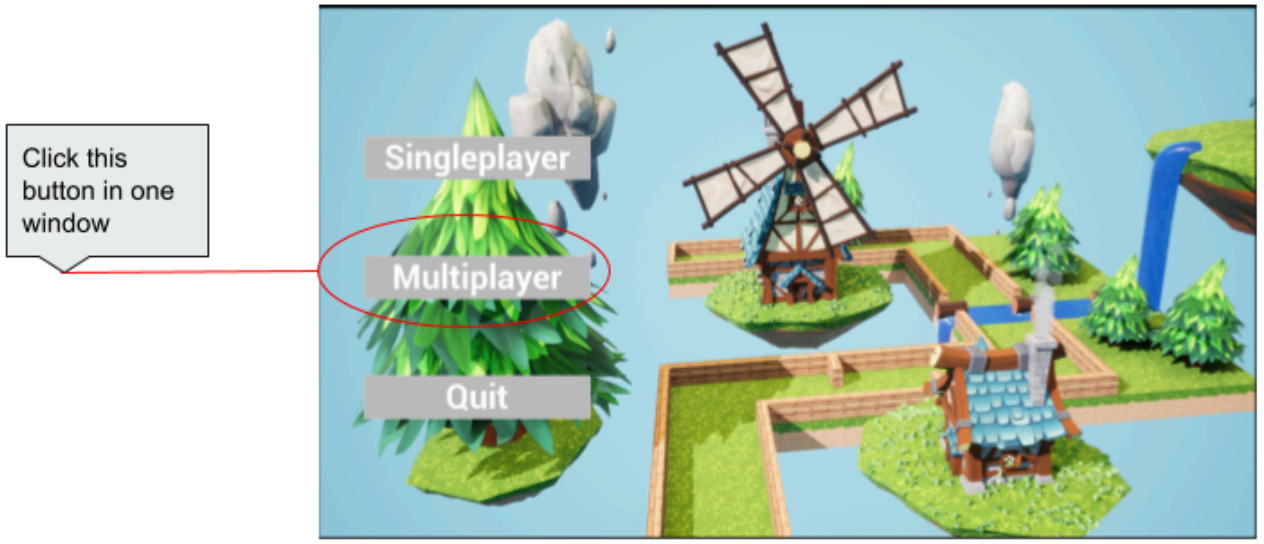
Starting a singleplayer game



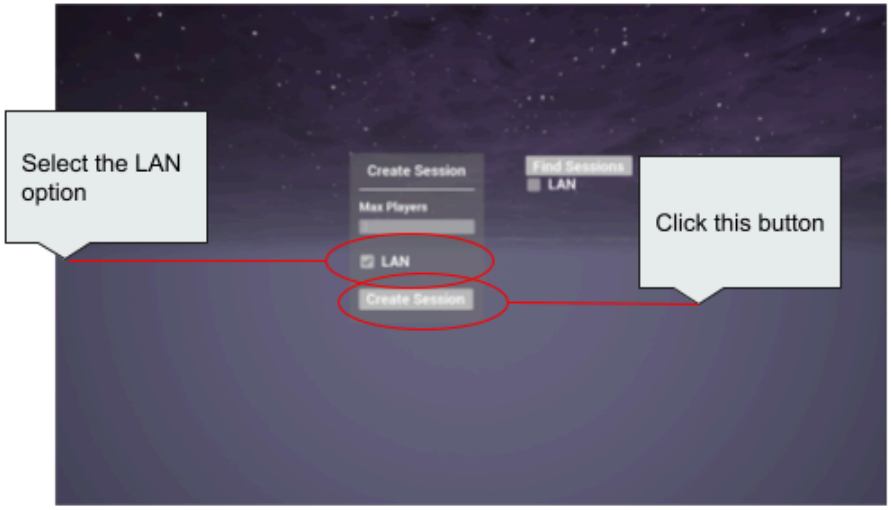
Starting a multiplayer game



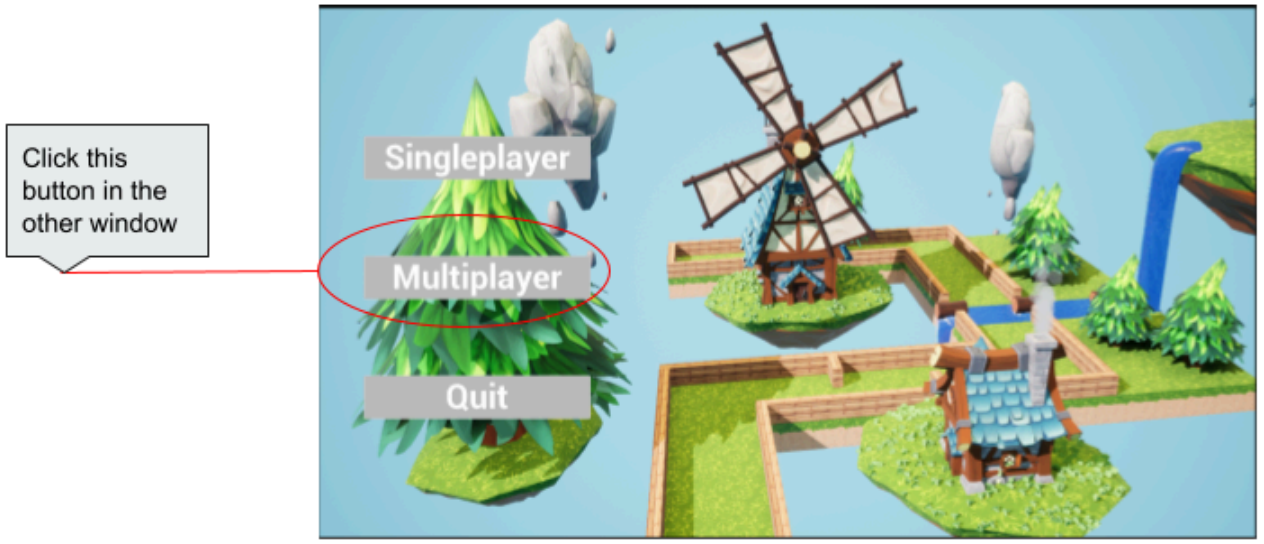
Starting a multiplayer game



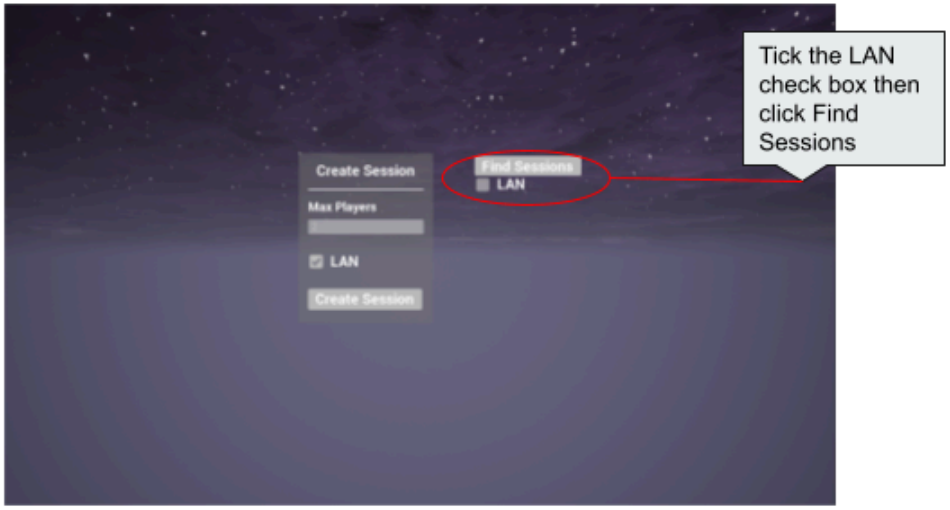
Starting a multiplayer game



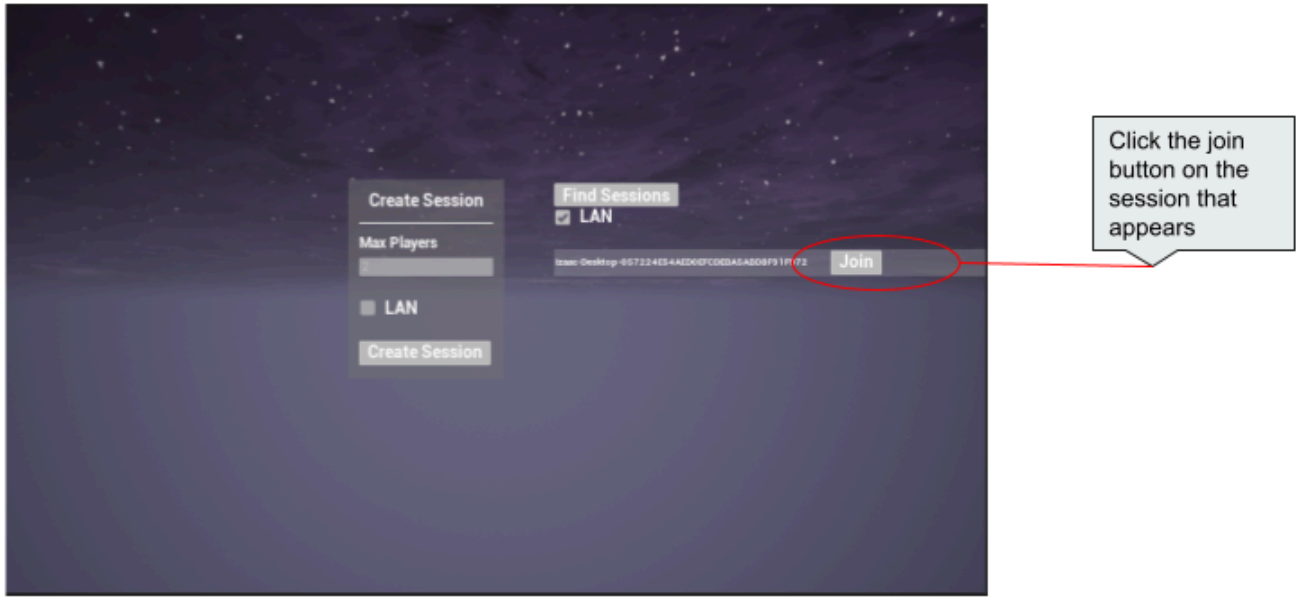
Starting a multiplayer game



Starting a multiplayer game

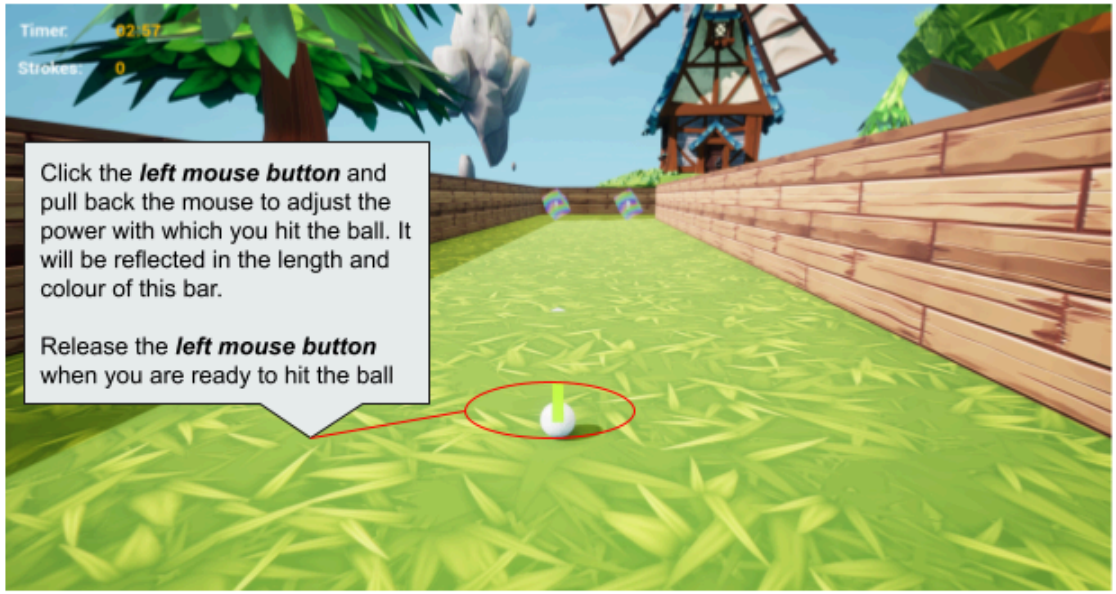


Starting a multiplayer game



Aiming the ball

Hitting the ball



Hitting the ball

After hitting the ball your stroke count will increase by 1, the aim of the game is to get the ball in the hole in the least number of strokes.

Getting a powerup

To get a powerup you should hit the ball into a “?” box

Pause the Game

To pause the game at any time press the ***escape*** key. Press continue to resume the game and quit to quit to the main menu

Getting a powerup

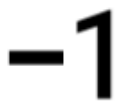
To get a powerup you should hit the ball into a “?” box

Getting a powerup

Types of power ups

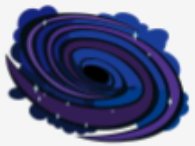
Ghost: Allows the player to move through walls and other obstacles

-1: Remove a stroke from the current players count



Power hit: increases the maximum strength of the players next stroke

Hole Magnet: Attracts the player to the hole



Types of power ups

Tornado (multiplayer only): Creates a tornado around the players ball that obstructs other Balls

Nuke(multiplayer only): Sends a missile to the other players ball to knock It from its current position

Jump Ball(Non-VR): Summons a tornado to make the players ball jump up

Sudden stop(Non-VR)): Instantly stop the players ball while it is moving

Types of power ups

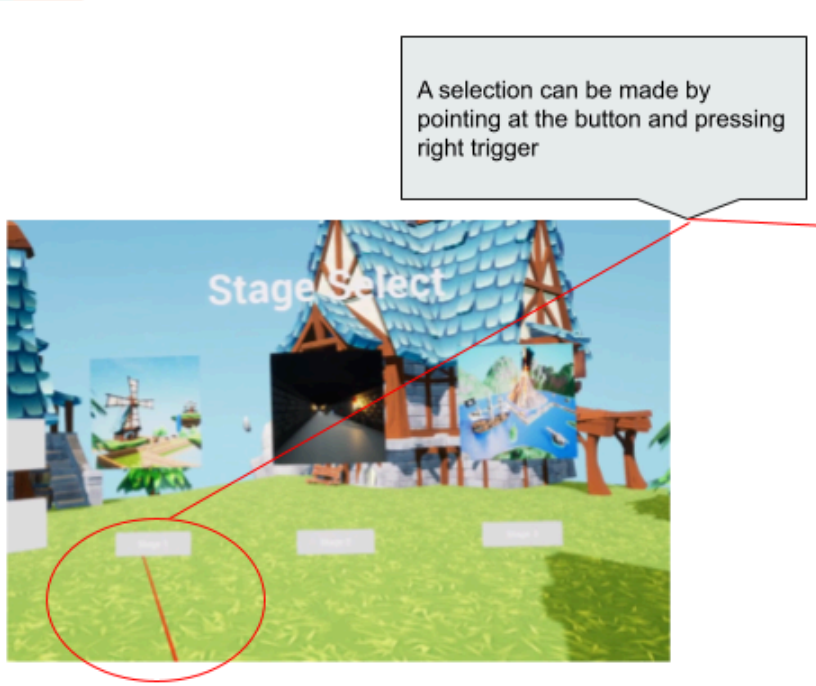
Stun Ball (Multiplayer only & Non-Vr): Causes the screen to become blurry and monochrome and also incurs massive accuracy debuff for all other players

VR Version

Opening VR version

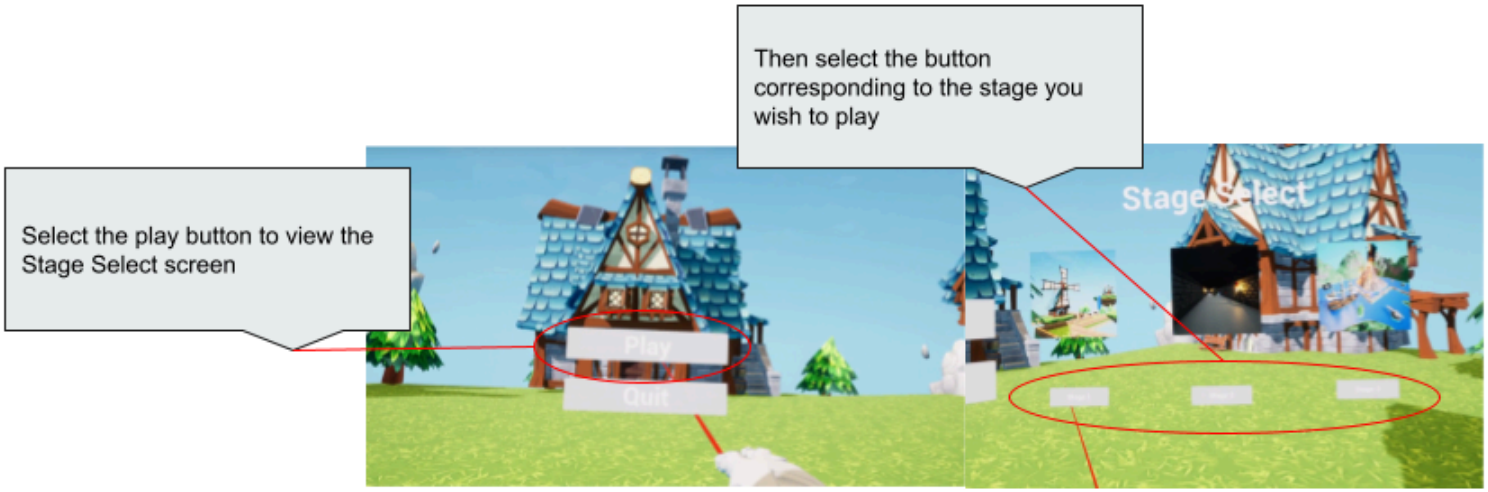
To start the VR version of the game please open the ***VR_main_menu*** level inside the Levels folder

Making a selection



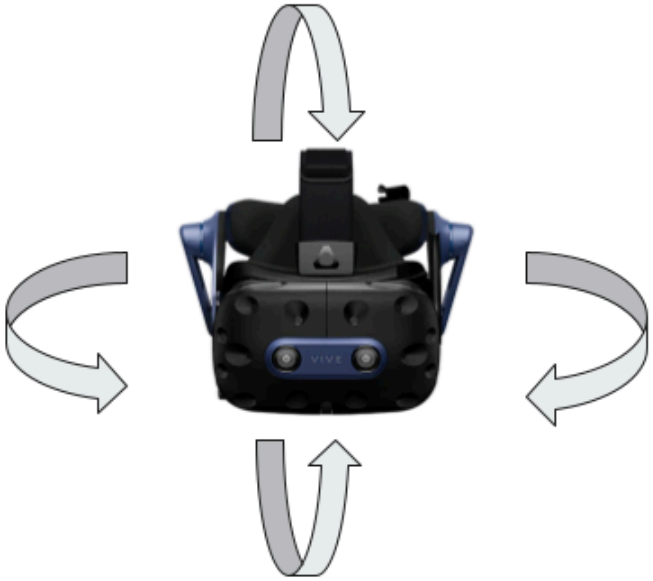
IMPORTANT: If you are having difficulty pressing the button, try to press the button while moving around in game

Level Selection



Looking around

Looking around can be achieved by physically turning your head while wearing the VR headset



Moving


Movement can be achieved via the moving the thumb on the left hand trackpad
VIVE

Picking up the club

Throwing the club

The golf club can be dropped/thrown by holding and releasing the right handed trigger

Using the holographic interface

Holographic interface can be opened holding your left hand in front of you and angling it such that the back of your hand is pointing towards the sky

Using the holographic interface

Point at the “Quit” Button to return to main menu.

Hitting the ball

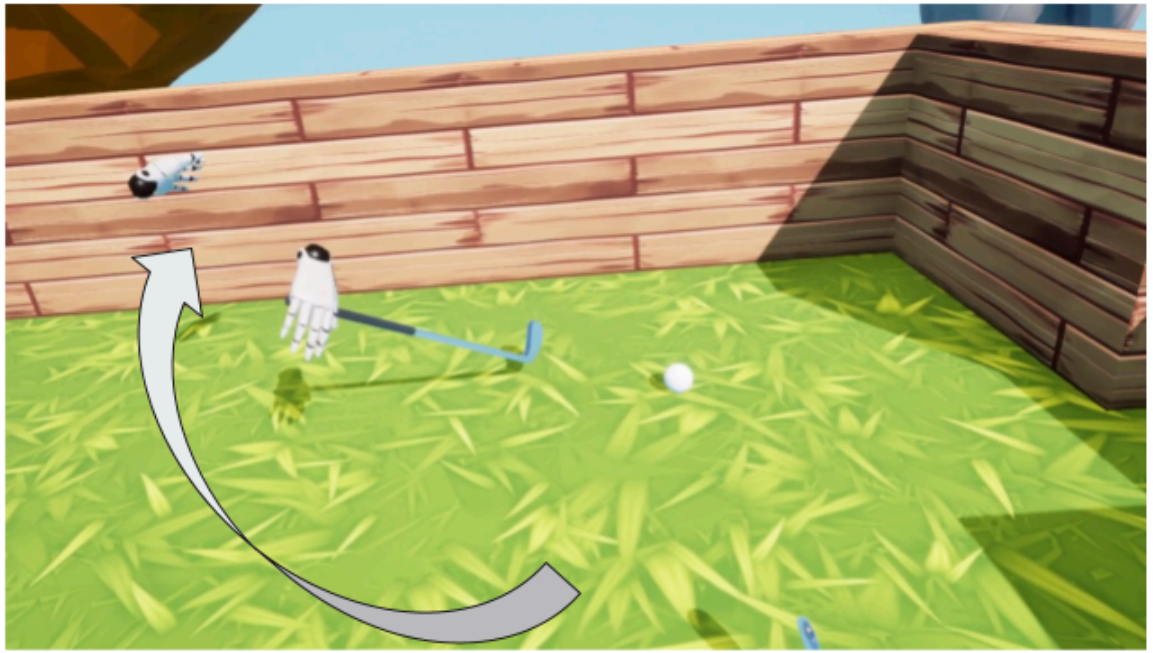
Hitting the ball with the golf will cause the ball to be launched. Note that a penalty system is in place to prevent a player from continuously pushing a ball instead of striking it

Activating Powerups

Powerups can be activated by clicking on the left handed trigger button

Ending a turn

The end of a turn is defined as when the ball ceases to roll. When this happens the camera will shift to the other player's perspective and participating players will need to pass the headset along to the next player.



Level Completion Screen

After you have completed the course press the continue button to wait for other player completion or if everybody is complete, return to main menu.