Simple Portal

Assignment 2 - Technical Design Documentation

Tutor name: Jason Haasz

Author name: Xun He(30451248)

Table of Contents

Project Overview

Simple Portal is a First Person Action Game like “Portal”. Players can generate two portals to help him or her through the levels. By interacting with portals, players can experience some unique movements and actions due to the gravity. When the player enters a new level, he or she needs to move a cube and place it on the pressure pad to open the exit. If there is nothing on the pressure pad, the exit door will close. The details about the puzzles will be described in Game Logic.

Delivery Platform

- PC
  Reason:
  The PC platform pays more attention to the keyboard and mouse operation. The mouse is more accurate than the gamepad in placing the portal. Therefore, the basic controls of the game are based on the keyboard and mouse.

Core Mechanic Outline

Players can press the left and right mouse buttons to generate two portals on the surface of specific walls respectively. Players can stand before one portal to get a view of the other one. Also, they can enter one portal to reach the position of the other one so that they can move to areas that are inaccessible to normal movement.

About the basic design of each level, the player needs to place the companion cube onto the exit pad to open the sliding door. The exit pad will be arranged in a particular place. The player needs to change the positions of portals to help them reach the location of the exit pad. In the final level, the player will play against the AI. The details will be described in Game Logic and Artificial Intelligence.

Comprehensive list of all mechanics and features

- Movement of players and their cameras

1. Players can use WASD on the keyboard to do normal 3D movements, such as forward, backward, left and right.
2. Also, they can control the cameras by moving the mouse in order to look in different directions. Because the game is a First Person Game, the camera of the player will always be in the eye position of the player model and follow the movements of the player without delay.

- Portal

1. Players can use the left and right mouse buttons to generate two portals.
2. Portals can only be placed on the surface of specific walls. At the same time, only two portals can exist simultaneously in the game world.
3. Players can stand before one portal to get a view of the other one. The view will change according to the relative position between the player and the portal.
4. Also, they can enter one portal to reach the position of the other one. The inertia will remain consistent during the transition.

- Companion Cubes

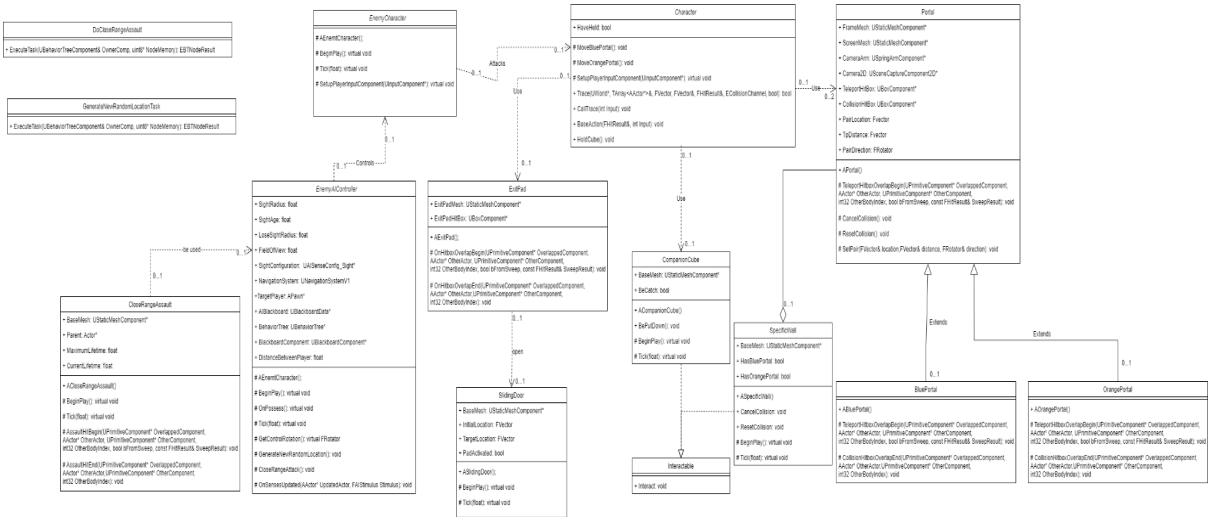
1. Players can press E on the keyboard to grab those cubes. And they can press E again to put down the cubes.
2. At the same time, the player can only grab one cube.
3. When grabbing, the cube will be placed in front of the player.

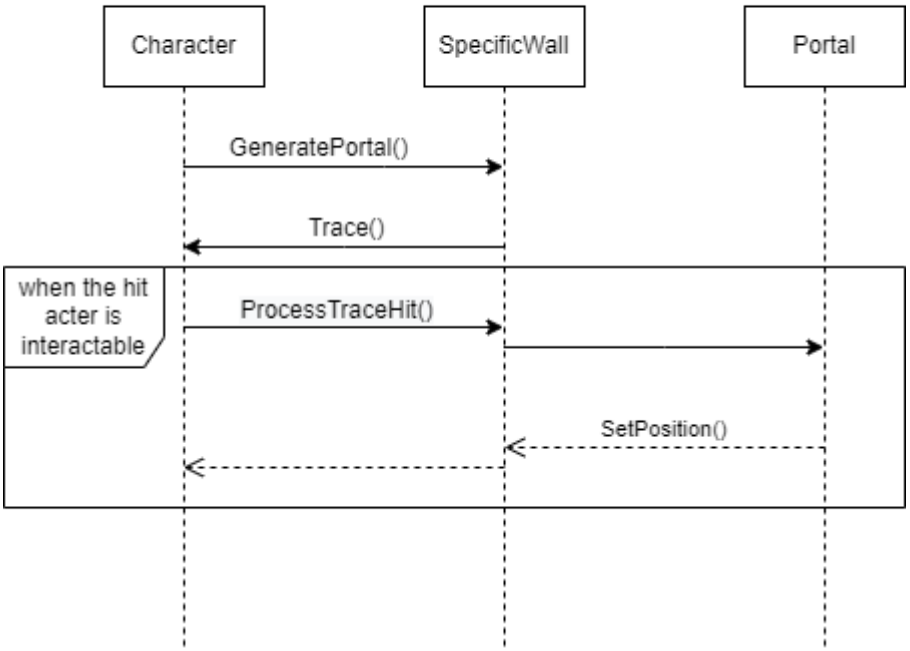
- Exit Pad

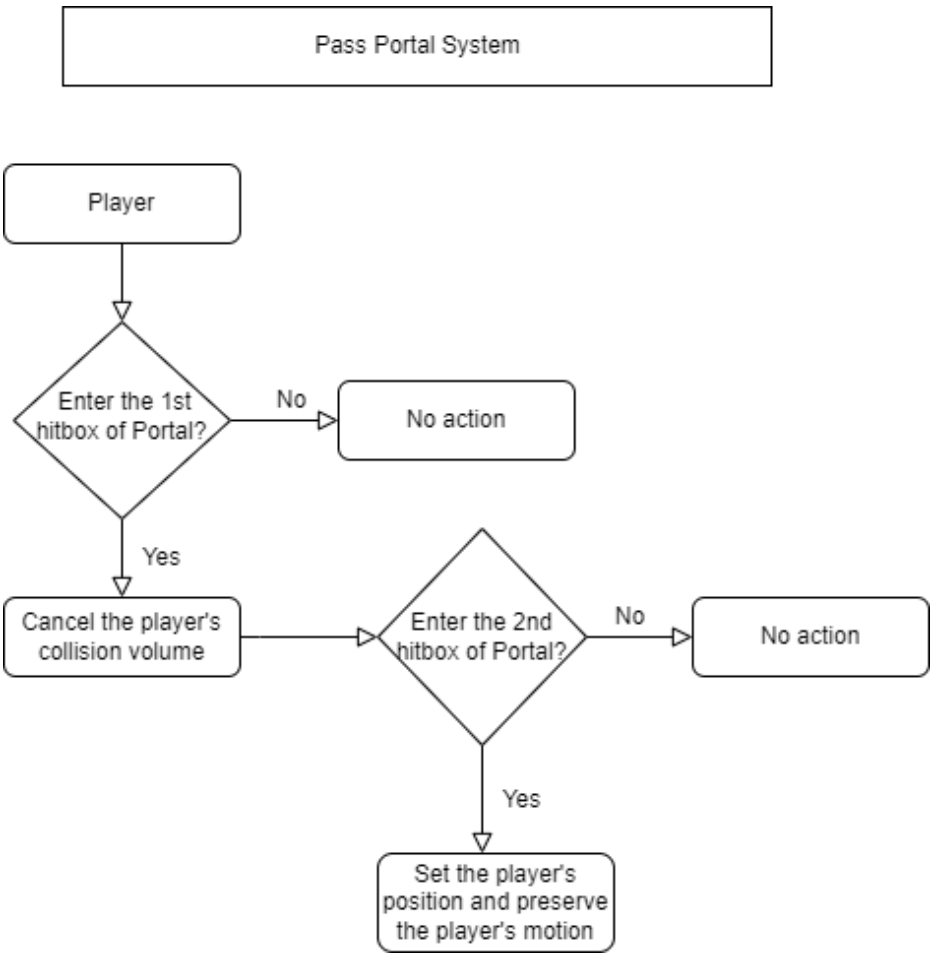
1. The pressure pad will be connected to the entrance of the next level.
2. When the player or the interactive cube steps on the pad, the entrance will open.

High Level Diagrams of topics

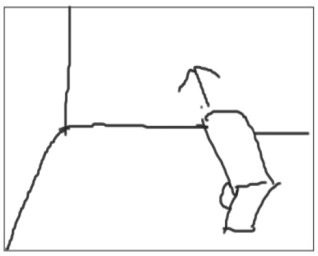
UML Diagram Link:
https://drive.google.com/file/d/1vBqxSSc8pIim6EIVQ6ImDZup0jFjzcgu/view?usp=sharing



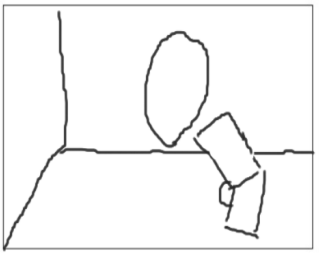




USER STORY/SCENARIO: Generate portal



The player can press left and right mouse buttons to click the wall.



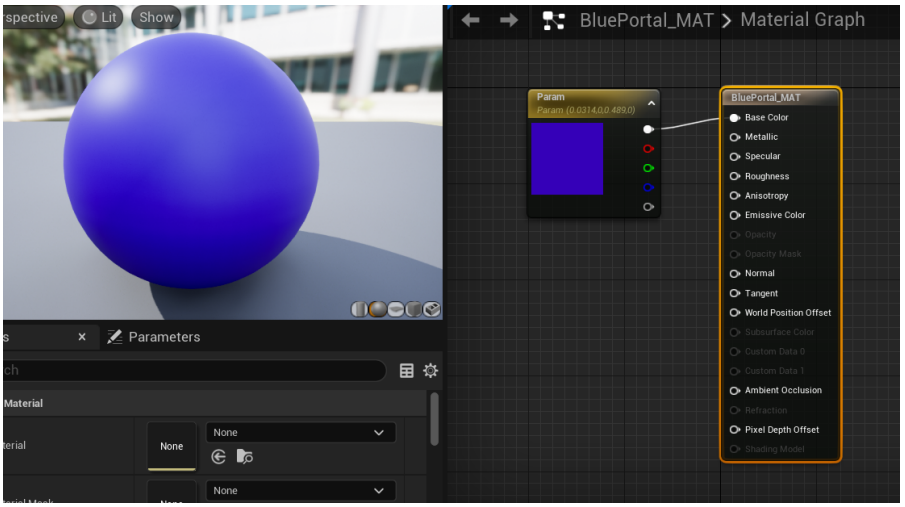
If the wall is interactable, there will a portal on the wall. And the player can see the view from the other portal.

If the wall is not interactable, there will nothing happen on the wall.

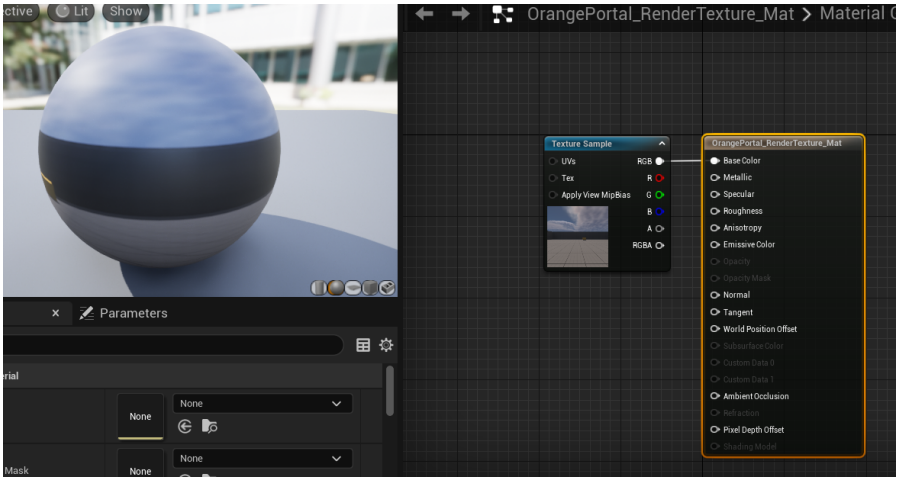
Materials and Shaders

- Portal

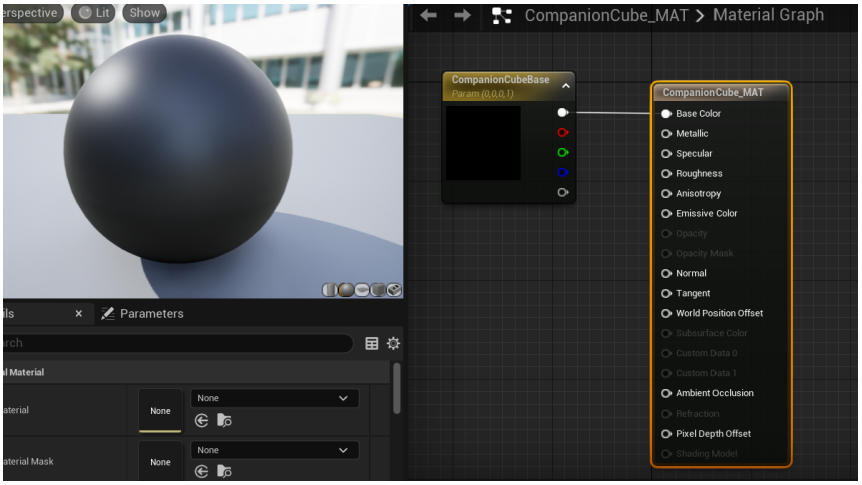
1. Frame:
   In the previous design, the frame will not be made of any material. However, I think it is necessary to help players recognize which portal it is. So I add a simple material to give colours to each portal.

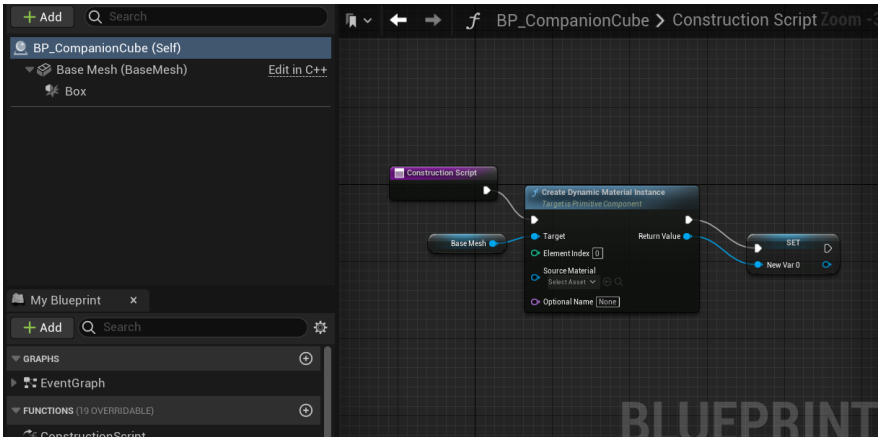


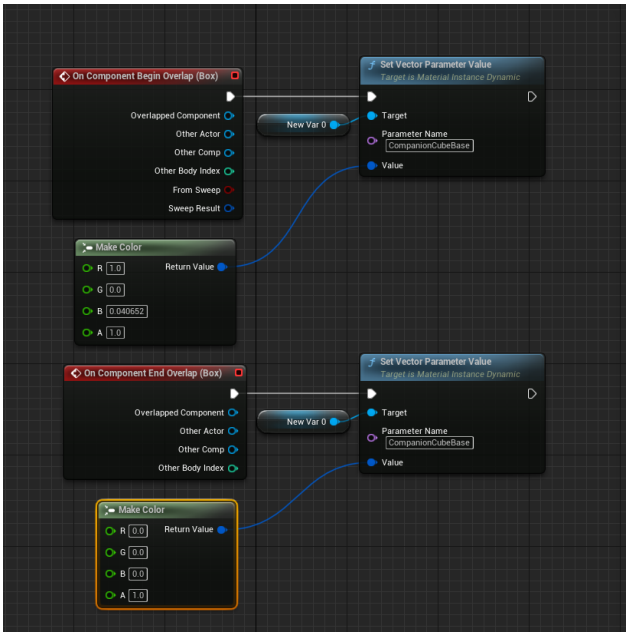
2. Screen:
   The material of the screen will be converted from the render texture through the camera 2D of the other portal. Therefore, the material can change depending on the location of the portal. Although this method will decrease the resolution of the screen display, it will be convenient for the later movement of the camera 2D.



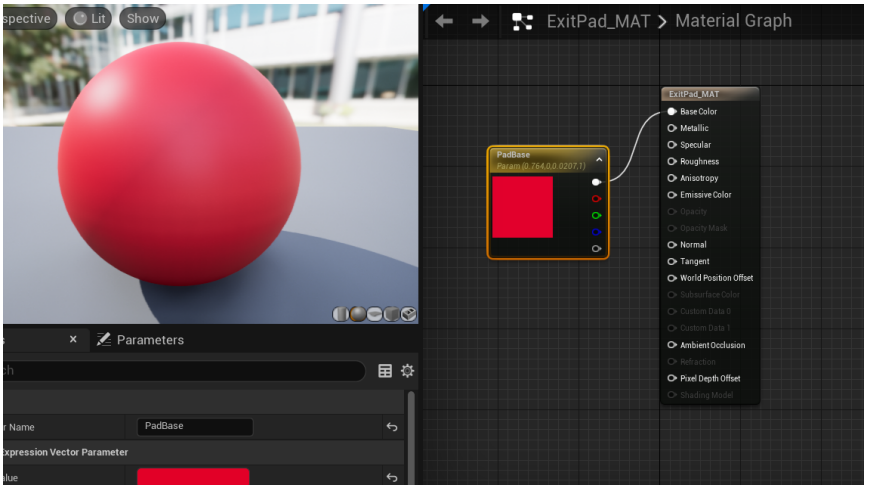
- Companion Cube
  To help players pay more attention to the companion cube and distinguish it from other common cubes, I add a dynamic material to it. The material base will be pure black. There will be a box collision around the cube. When the player is near to the cube, the cube will become red to notice the player.

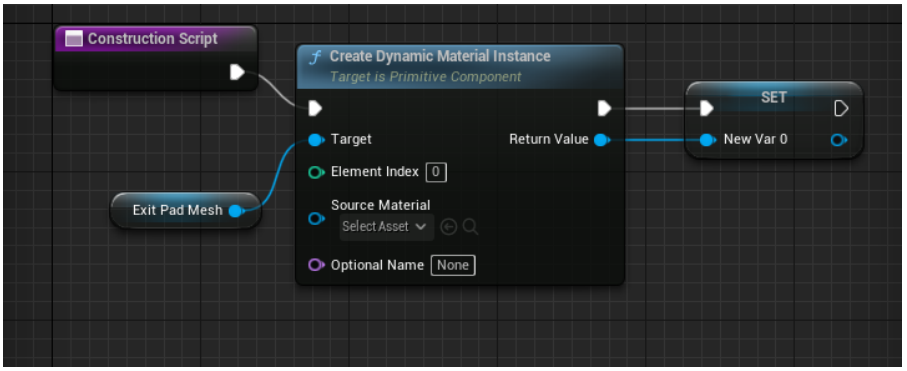


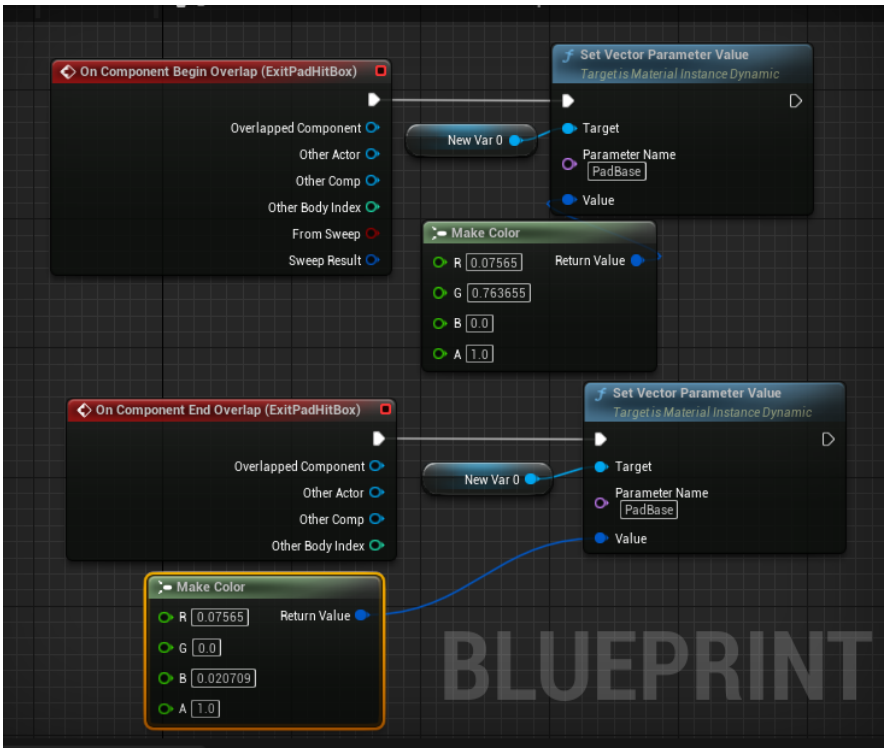




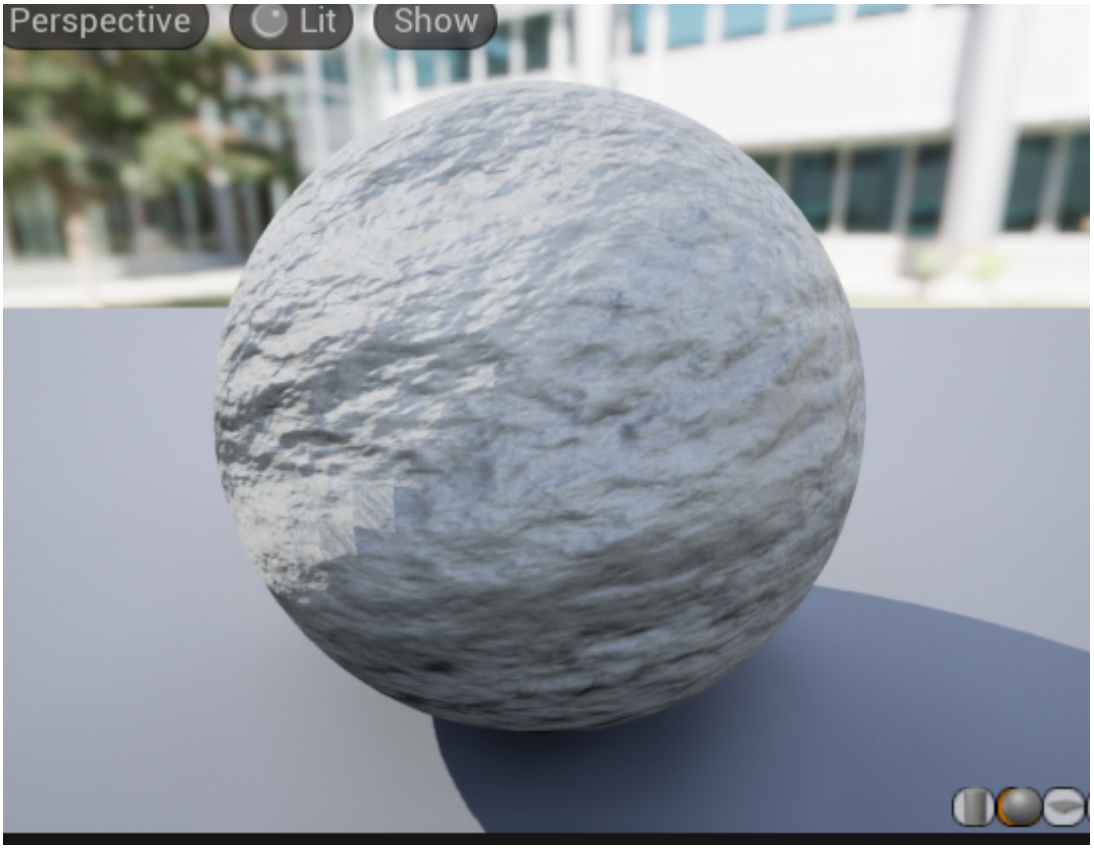
- Exit Pad
  The material of the pad will be similar to the companion cube. When the player steps on the pad, it will become green from red. It can help the player recognize whether the pad is working without watching the sliding door.

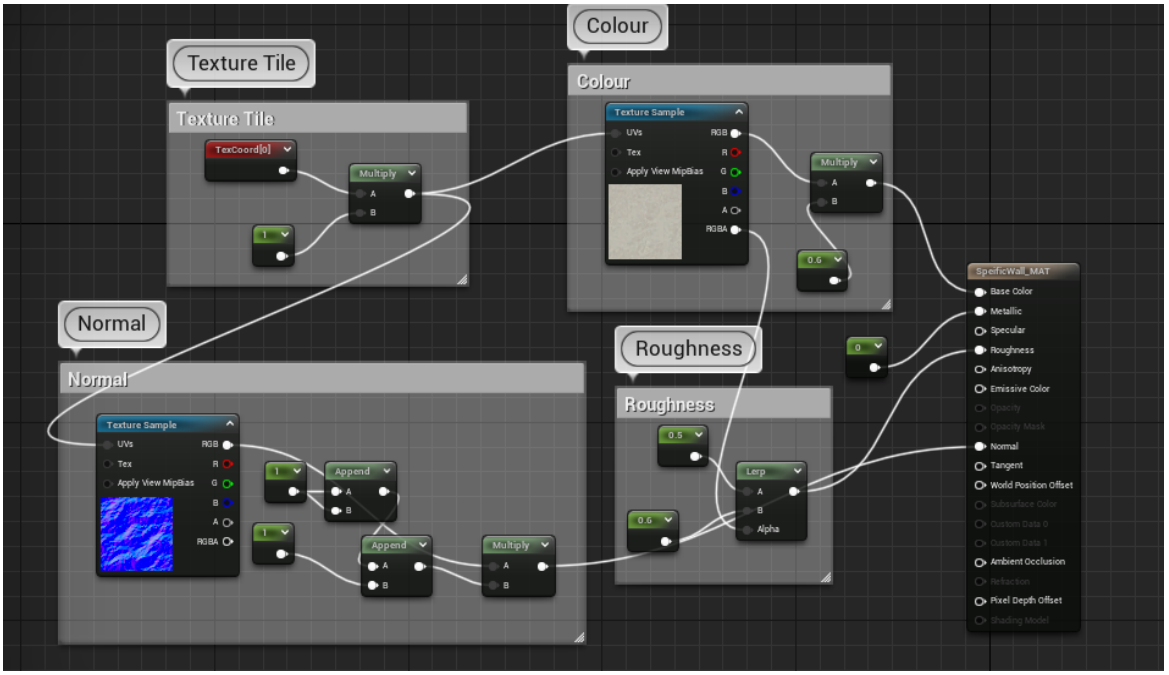






- Specific Wall
  To help the player distinguish between the common wall and specific wall, I use a rock material onto the specific wall. In the shader, I focus more on the base colour, metallic, roughness and normal. The detail of shaders is shown in the following screenshot.

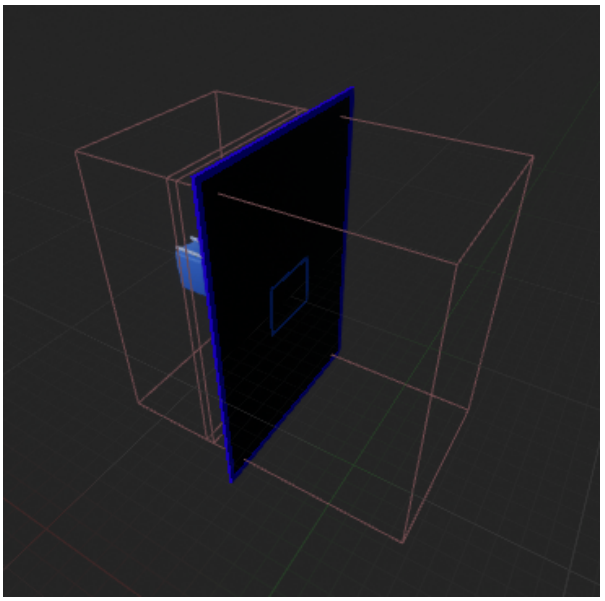




Details about 3D Objects, Terrain & Scenes

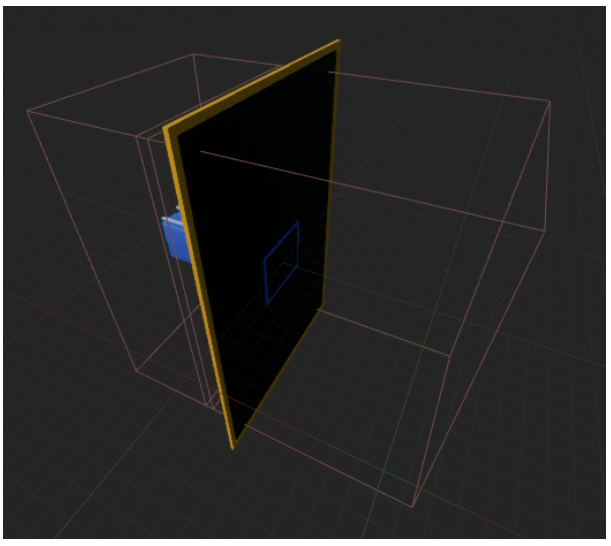
- Blue Portal

1. Purpose: To help the player move to the other portal for reaching the exit
2. Class Type: Custom Actor
3. Size: The same height and width as the player
4. Models



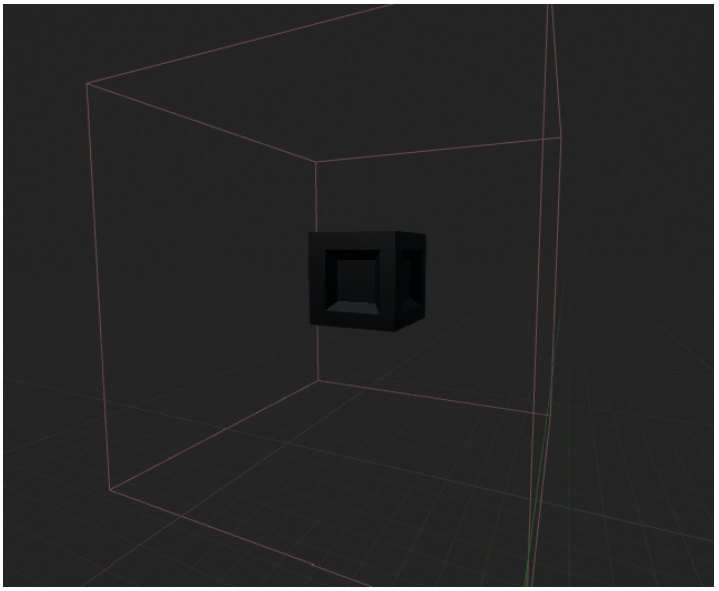
- Orange Portal

1. Purpose: To help the player move to the other portal for reaching the exit
2. Class Type: Custom Actor
3. Size: The same height and width as the player
4. Models



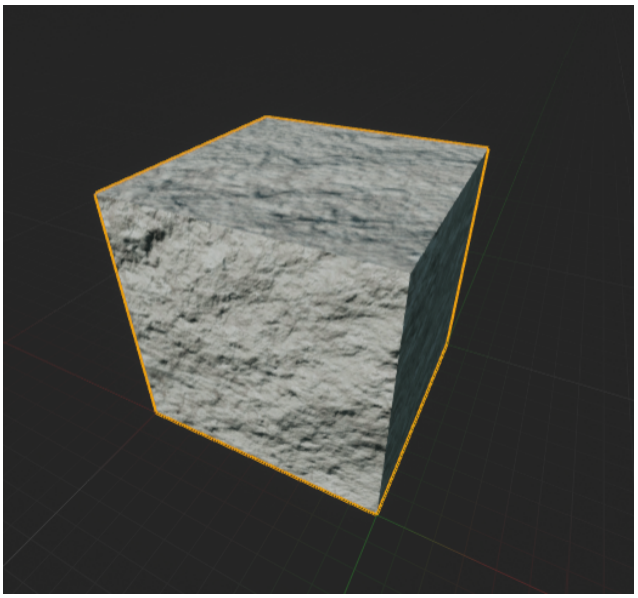
- Companion Cube

1. Purpose: To place on the pressure pad to open the exit door
2. Class Type: Custom Actor
3. Size: The half height and the same width as the player
4. Models



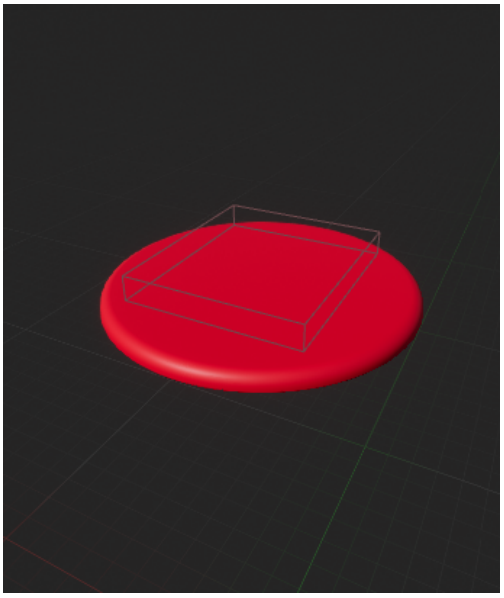
- Specific wall

1. Purpose: To restrict the player's ability to place portals anywhere
2. Class Type: Custom Actor
3. Size: Variable as the level demands



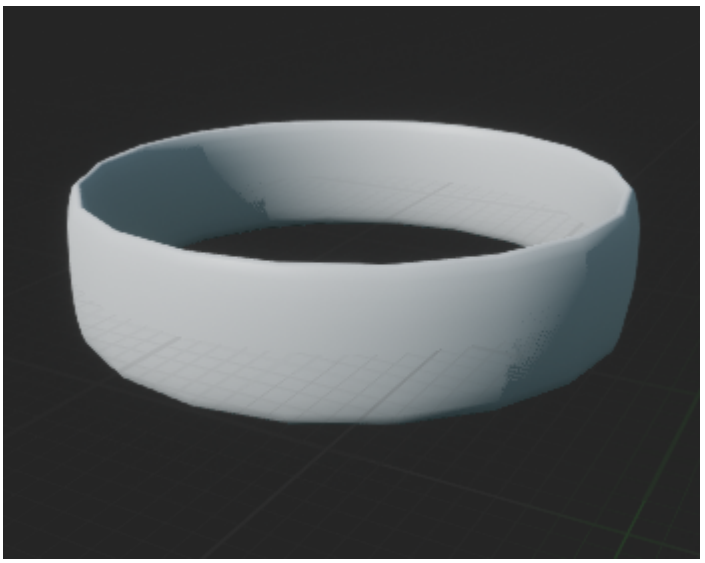
- Pressure pad

1. Purpose: To open the entrance of the next level
2. Class Type: Custom Actor
3. Size: The same width as interactive cubes
4. Model



- Close range assault

1. Purpose: Hurt the player
2. Class Type: Custom Actor
3. Size: A bit larger than enemy character
4. Model

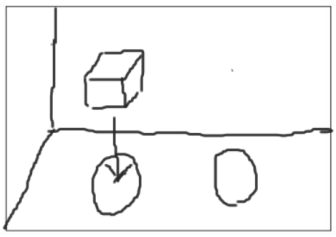


Use of Physics Engine

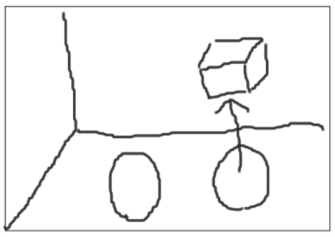
When the player or the interactive cube passes through the portal, the inertia remains the same as before the portal. Also, the direction of the player's motion will be changed to the direction of the portal exit's normal. For example, when the player jumps into the portal from a high place, he or she will leave the portal exit with a brief acceleration.

Example for storyboards:

USER STORY/SCENARIO: Physics expression



The interactive cube falls into the portal.



After the cube enters the portal, the cube will jump to the same height when it fell.

Interaction Elements

- Interactive cubes and specific walls

Both of them will be in **Interactable** class, because it can help the system recognize whether the actor belongs to the interaction element when the player wants to grab or use ray casting.

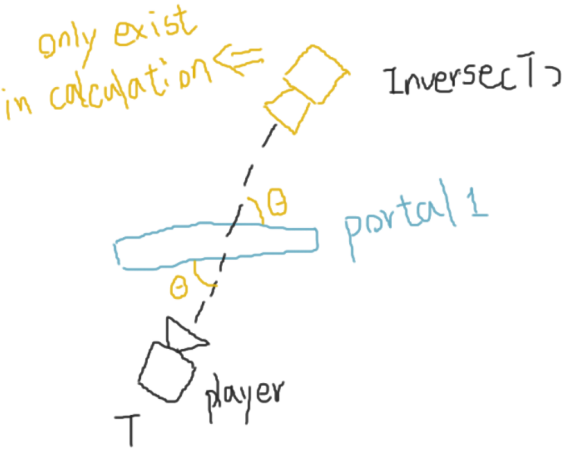
- Camera 2D of the portal

This camera belongs to **UsceneCaptureComponent2D**.

About the projection of the screen, the screen from this camera will be converted to **Texture Target**. Due to a total of two portals, we get two **Render Textures**. Then, I will convert those two **Render Textures** to **Materials**. Next, the screen mesh of one portal will adopt the material converted by the other portal's texture.

About the movement, the camera 2D can move in Tick() according to the relative position between the player and the portal. The distance and angle between the player and portal1 equal to the distance and angle between portal2's camera 2d and portal2 respectively. Therefore, I can use Inverse(T) to get the position of the hypothetical camera 2D in Portal1. Then, I will rotate and move this camera 2D to the direction and the location of portal2.

There is a following picture to describe the calculation of the camera 2D.



Game Logic

- Generate portals

In this part, I will use ray-casting to move the portal.

Firstly, I will check whether the hit actor is the specific wall. If it is, I will use **FHitResult.Location** and **FHitResult.Normal** to get the location and direction of the wall.

Then, I will set the location of the portal equals the location of the wall plus a few distances according to the wall's direction to avoid overlapping the portal with the wall.

Also, the direction of the portal will be rotated to the direction of the wall by **FRotator**. The **PortalLocation** and **PortalDirection** will be updated.

- Teleport

In this part, I will use 2 hit boxes to realise teleporting.

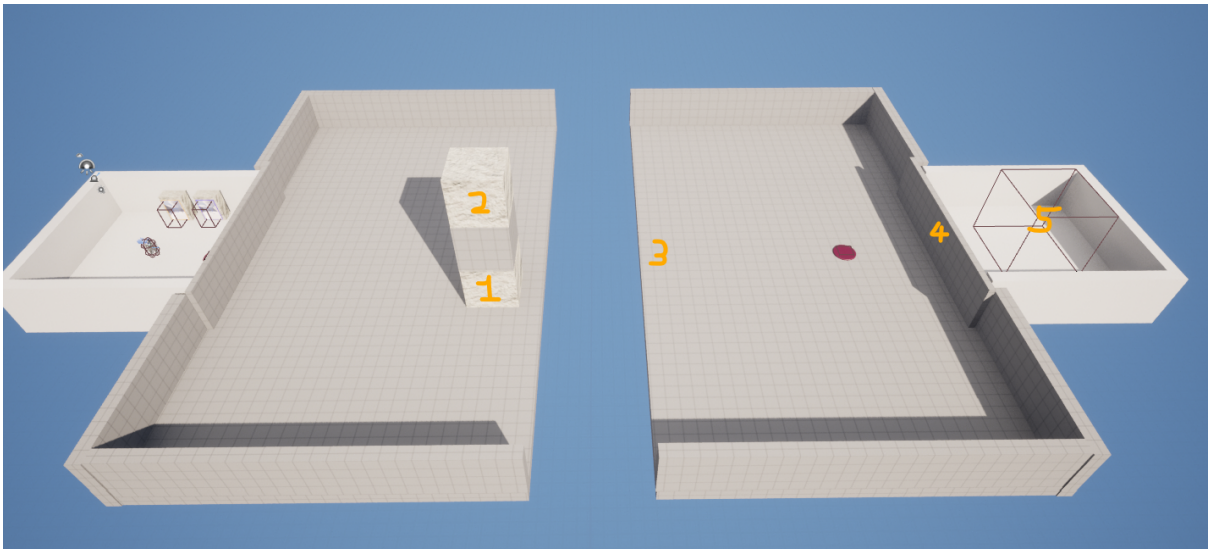
The first hit box will be placed in front of the portal. When the player enters this hit box, he or she will have no collision volume. This can help him or her not be blocked by the portal and the wall.

The second hit box will be placed inside the portal. When the player enters this hit box, his or her motion, such as **FVector**, will be preserved.

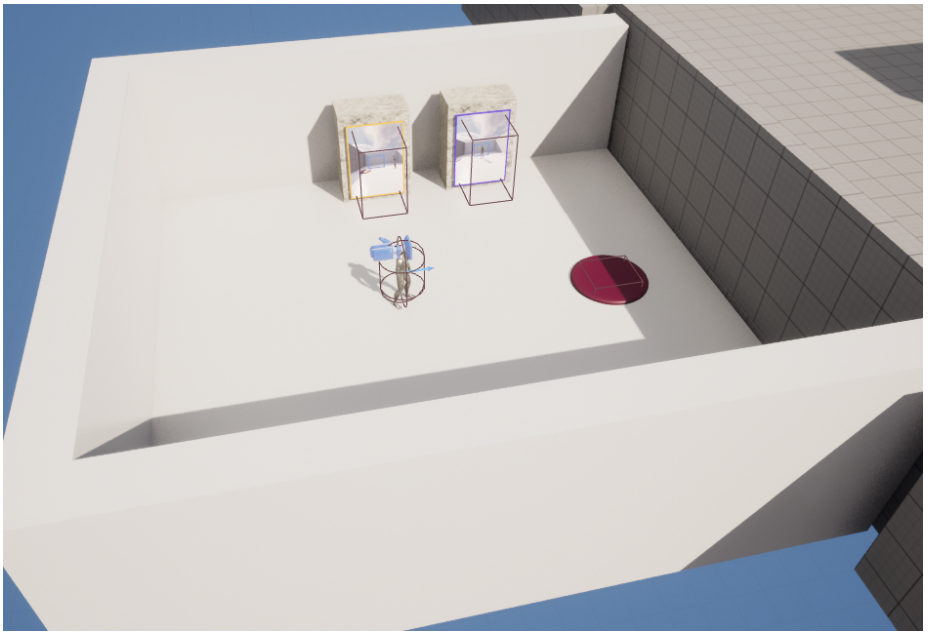
Then, he or she will be teleported to the position, which equals the location of the other portal plus a few distances according to the direction of the other one. The extra distance can prevent the player from being teleported repeatedly. Next, the preserved motion will be added to the player.

- Details of the easy level design

The following image (see figure1 and 2) is the level in the easy mode. In figure2, the player can quickly understand each interaction with the other elements. In figure1, there will be a big fissure that the player cannot jump directly to the other side. The player needs to generate the portals at locations 1 and 2 (highlighted “1” and “2” in the figure1). Then, he jumps through portal1. Due to Mechanics of the portal, his location will transform to location 2 and his forward vector will continue to help him reach location 3 (highlighted “3” in the figure1). Next, he can step onto the red exit pad. The pad will become green. And the sliding door (highlighted “4” in the figure1) will start to move to the left. Finally, the player can go to the next level through the box collision (highlighted “5” in the figure1).



(figure1)

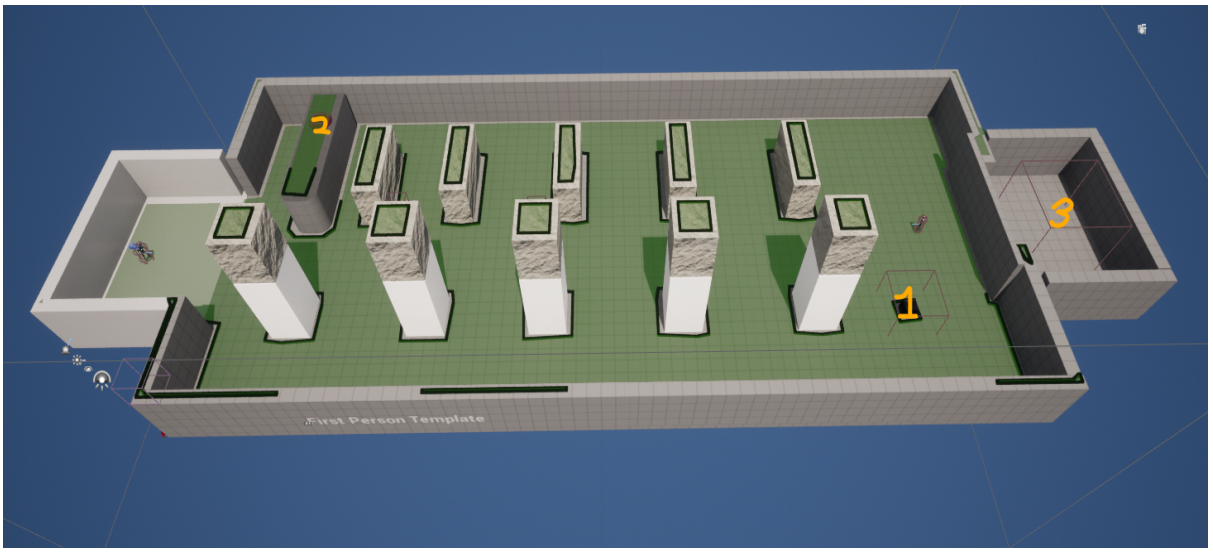


(figure2)

- Details of the hard level design

The following image (see figure3) is the level in the hard mode

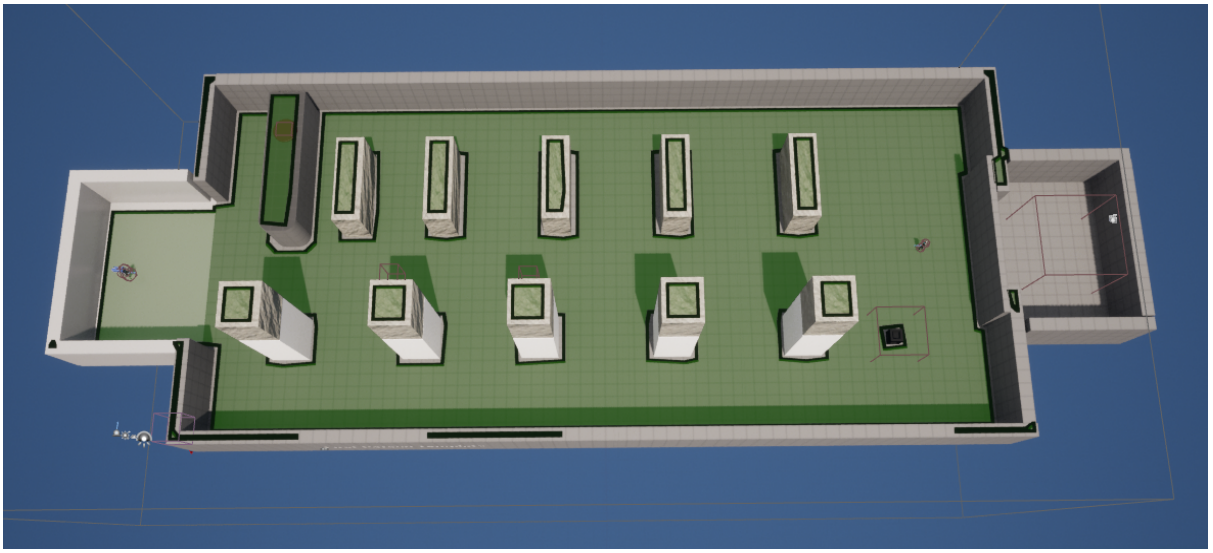
In the final level, the player will face the AI character. There is no method to hurt the AI. The player needs to elude the AI and take the companion cube (highlighted “1” in the figure3) to place it on the exit pad (highlighted “2” in the figure3). Then the sliding door will open and the player can go to the exit room where AI cannot enter (highlighted “3” in the figure3).



(figure3)

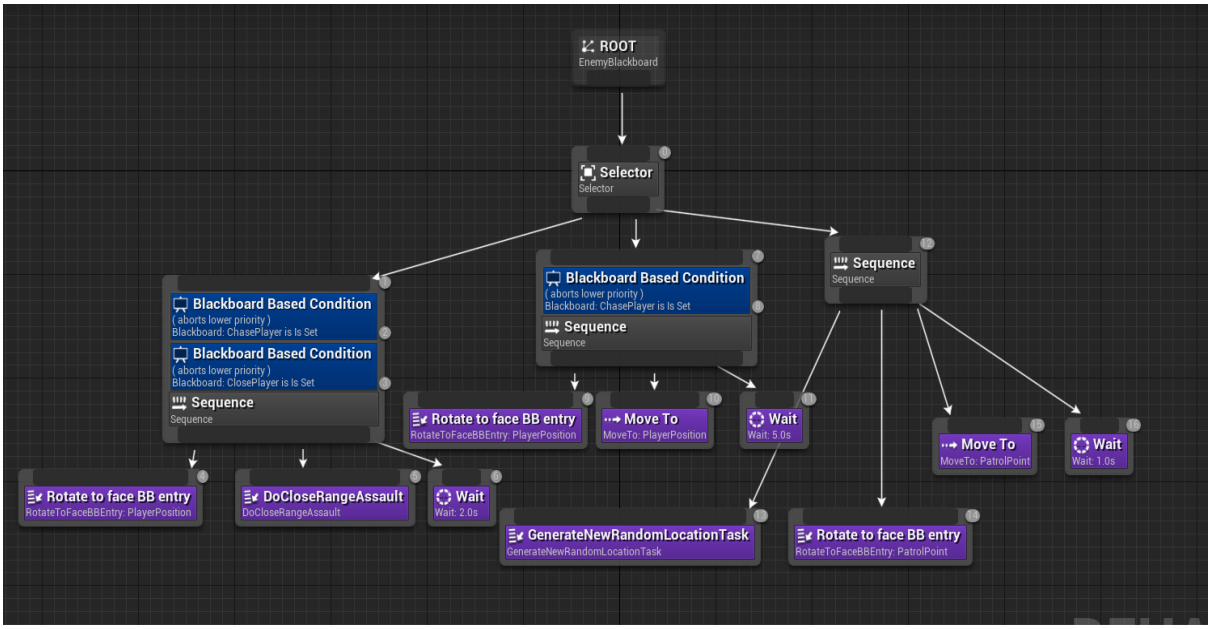
Artificial Intelligence

About the design of AI in the game, I plan to create this AI character as the boss which cannot hurted by the player in the final level, because I wish the player would focus more on the puzzles in the levels. The AI character only can do a close range attack. Close-range attacks will produce a ring collision around AI. This collision will exist for only 1 second. I also set the wait time as 2 seconds after attacking. This design can make the AI more balanced. The Sense used in AI will be sight.
The navigation meshes will cover the whole final level as shown in the figure4.



(figure 4)

The behaviour tree is shown in figure5.



(figure 5)

Particle Systems

The particle effects will be used in generating portals and close-range attacks.

- Generating portals

Before the portal is generated, there will be some particles of the corresponding colour, which will fade in on the wall. And they will fade out very quickly. This can give the player a visual feedback that generates the portal.

- Attack

When the AI character attacks, the hit box produced will have particle effects to notice the player dodge the attacks visually. The master colour will be red to make it more striking.

Audio Details and Specifications

The audio will be used in opening the sliding door and close-range attacks.

- Opening the sliding door

When the player steps or places the companion cubes onto the exit pad, there will be a mechanical sound to notify the player that the sliding door is opening. This can decrease the confusion of the player when he cannot see the sliding door directly.

- Attack

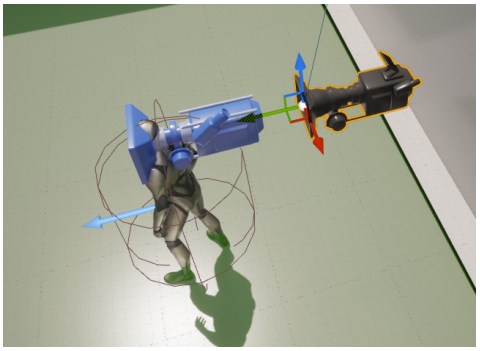
When the AI character attacks, there will be an explosion sound around the AI character, which can notice the player that the attack is running on hearing. Also, the explosion sound can make the attack more striking.

Cinematics Information

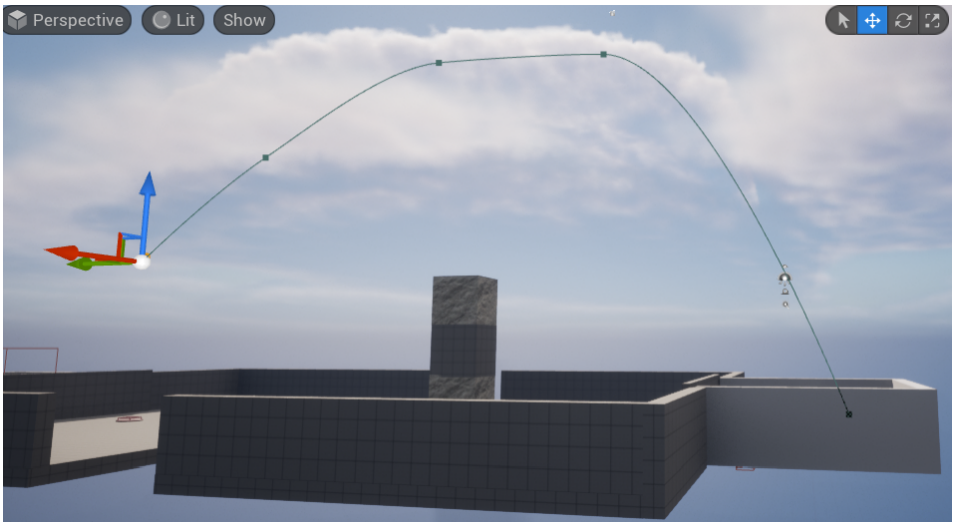
I set the path of Cine Camera Actor to overlook the whole level. This can help the player quickly understand the design and target of this level. The final location of the camera will be placed behind the character. This can help the movement from Cine Camera Actor to First Person Camera smoother.

The initial rotation is

The final location is



The path in easy level is



The path in easy level is