Lego Portals

Technical Design Doc

By Luke Brown

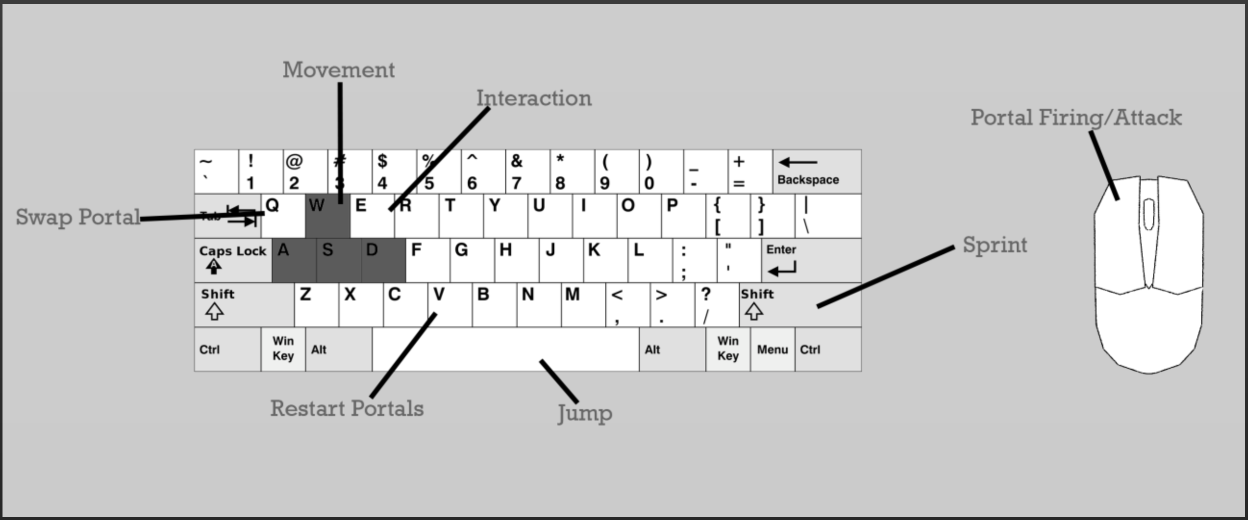
B017629L

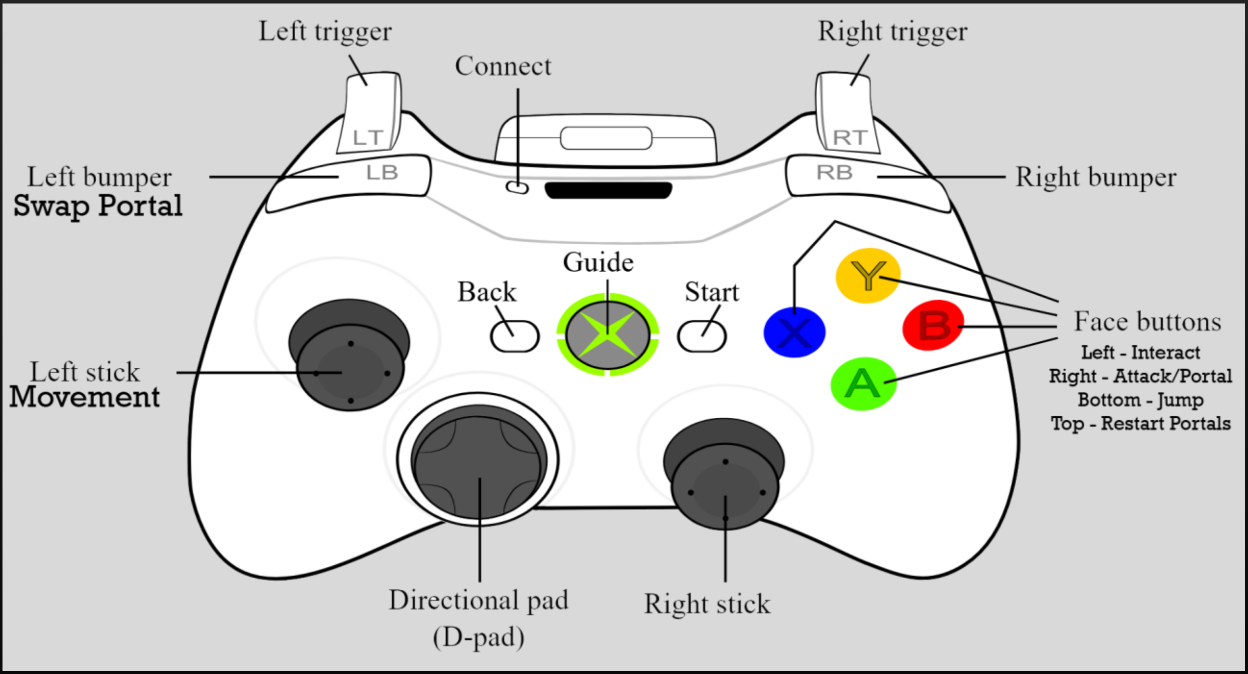
Context Page

To Access my YouTube playlist on the development of this project.

CLICK HERE

Controls


Movement
Interaction
Portal Firing/Attack
Swap Portal
Sprint
Restart Portals
Jump


Left trigger
Connect
Right trigger
Left bumper
Swap Portal
Right bumper
Guide
Back
Start
Face buttons
Left - Interact
Right - Attack/Portal
Bottom - Jump
Top - Restart Portals
Left stick
Movement
Directional pad
(D-pad)
Right stick

Project Introduction

Project Goals

The goal of this demo is to create a Prototype of a Lego game featuring most of the essential mechanics/features that a Lego game would feature. I wish to create a Lego game that combines aspects of both the classic Lego Games, and Valve's Portal Series. I want the main character to have a Portal Gun Ability that allows them to fire portals in the scene to complete puzzles, as to how I will go about that, I will learn along the course of this module.

Challenges and Risks

There are many possible things that I want to complete to learn how it was made but in Unreal. Lego Building has been a major mechanic in the Lego games, and I have always been curious as to how they are developed, whether it is that they are separate pieces that gets put together or if they exist as one object like a skeletal mesh and would animate being together. The Portal Gun is another area I feel would be a nice challenge to implement due to how the Lego games had their aiming system dealt in contrast to the Portal series which is in First Person. I also have a deep interest in modular systems and improving my knowledge in these areas and I feel the Lego games are the best method for me to learn this as they aften feature similar mechanic but open to be changed in ways, like how they likely featured a base Lego character that would be incredibly modular for adding different abilities and models.

Project Introduction

Hardware Requirements

In terms of hardware, this will be developed for the PC, and I will be using the Universities PC's but will also be using my own personal device and my device is weaker than the Universities with only a 2060 Nvidia Graphics card while theirs features a much more updated graphics card so it will be nice to see the difference in how my game will perform on different hardware.

In terms of the development side rather the build, Both PC's will be using Unreal Engine 5.2 and other external software. As to what software, they could vary during the course, but I will presume that I will end up using Photoshop, Premiere Pro, A modelling software of sorts (if I have the time), and other software that could possibly assist in developing this game.

	Minimum Requirement	Overkill Requirement
OS Version	Windows 10	Windows 10
CPU	Quad-Core Intel/AMD (2.5GHz)	8-Core Intel/AMD
Memory	8GB	16GB
GPU	DirectX 11/12 - Nvidia Geforce RTX 2060 6GB	DirectX 11/12 - Nvidia RTX 3090 24GB
Storage	600MB	1GB

Platforms

Target Platform

This will be developed for PC as it is the easiest way to test when I am building my project, and it allows a safe set for creating a build that works and leaves it open to add controller settings for further testing.

In the future I would love to attempt a working console version though this only would work if I had a way to test with console, as of now I have tested with controller input, and it works relatively well from my test.

Engine Summary

The Engine version of my Unreal Engine project will be in 5.2 as it is the most updated version that is also available on the University PC’s. I currently do not have any plans to use a plugin on my project as I feel the version has everything, I need to develop the demo and other external software can assist me.

In comparison to the other Engines available, Unreal offers the best for what I wish to accomplish, not only this but I feel much stronger with Blueprinting these ideas than coding in C# or C++. Using an engine Like Unity has shown in past experiences that it is better for usage on 2D games, whereas Unreal is perfect for 3D development as it offers many tools for creating 3D experiences with improved lighting options than Unity would have.

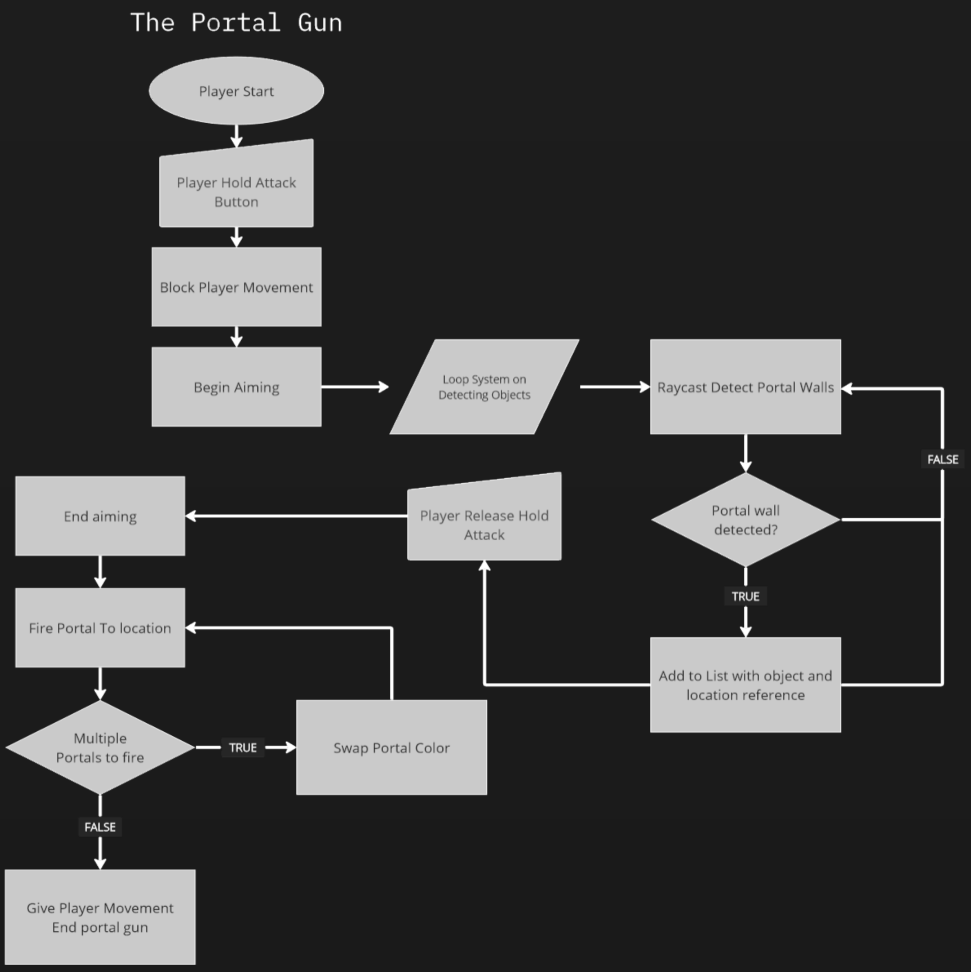
Debugging is another area I feel is great in Unreal though Unity has also got a great method for debugging content. I feel that Unreal is better for the number of options that I know of to help debug blueprints that aren’t working.

Systems And Diagrams

These slides are dedicated to all the main mechanics/systems of my project and some from the games that I have researched into. There is an explanation to them and a diagram to help showcase how they all work. All Diagrams were made Using Miro ().

System 1 – The Portal Gun

The Portal Gun is the biggest area I wanted to be done right in my project, and while I did not manage to complete it with the UI method classic Lego games Possessed (Brown, L., 2024f), I so believe that this is still a great concept for the Portal Gun and with more time in the future, I can get the UI aiming system working.

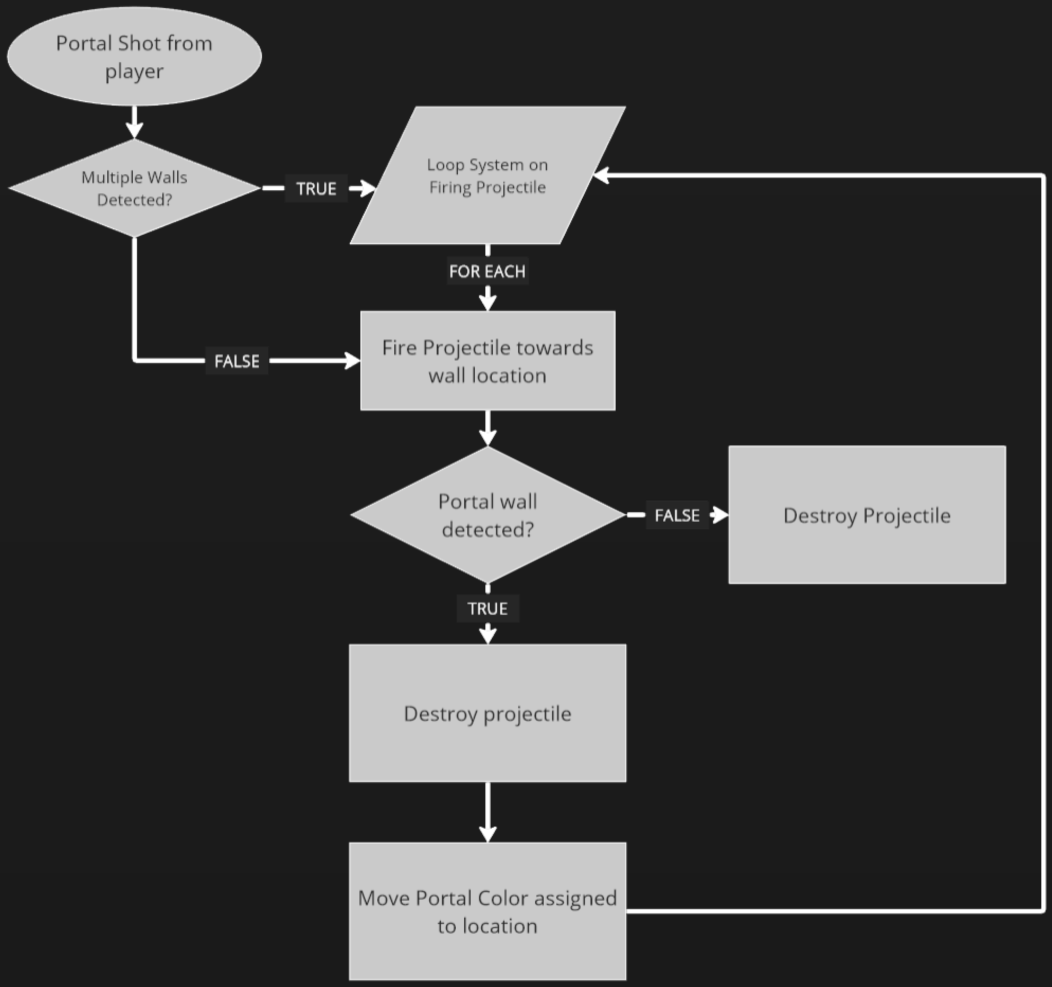

The Portal Gun
Player Start
Player Hold Attack Button
Block Player Movement
Begin Aiming
Loop System on Detecting Objects
Raycast Detect Portal Walls
Portal wall detected?
FALSE
TRUE
Add to List with object and location reference
Player Release Hold Attack
End aiming
Fire Portal To location
Multiple Portals to fire
TRUE
Swap Portal Color
FALSE
Give Player Movement End portal gun

Systems And Diagrams

System 2 – The Portal System

This system continues from the first where in this instance, once the gun has fired, will begin the projectile launch, if the projectile hits the portal walls it would then move one of the two portals to that location. This system loops if there are multiple portal walls that were detected in the Portal guns Aiming.

The Portal System

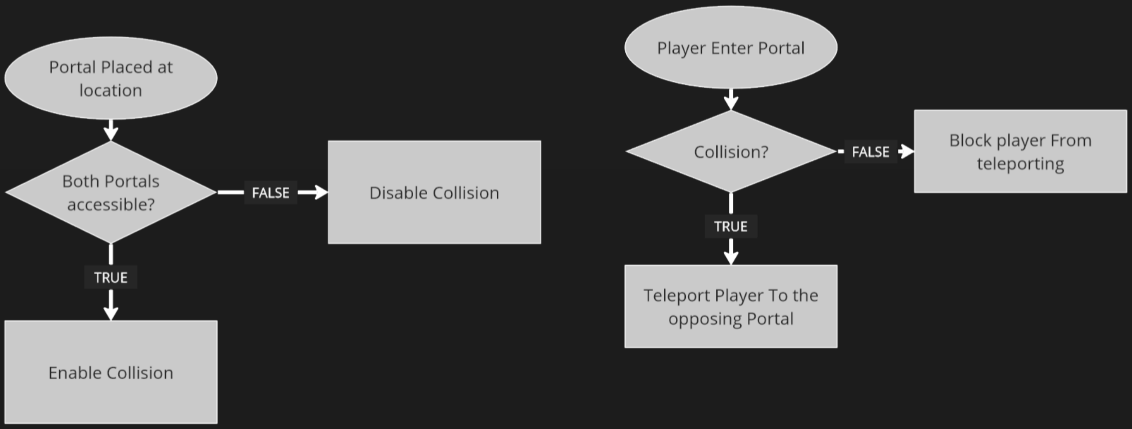


Systems And Diagrams

System 2 – The Portal Walls

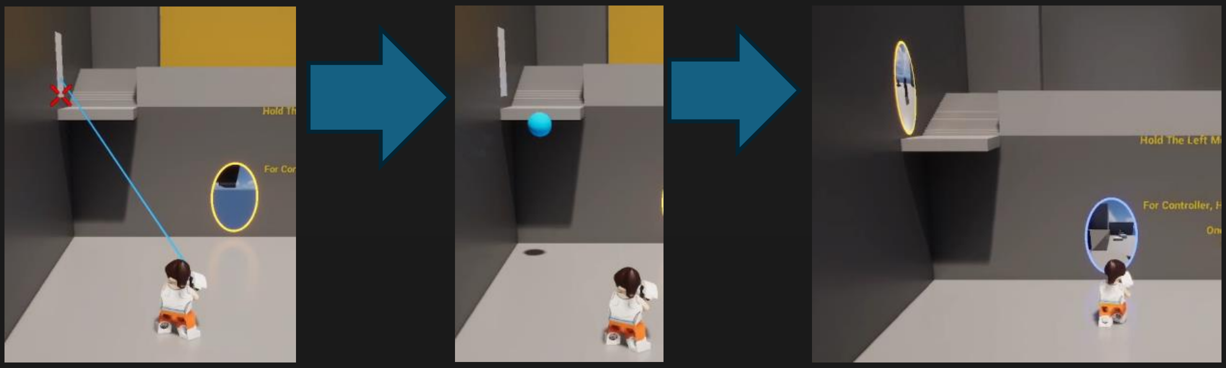
The Portal walls continue off the projectile and basically these are the check for the player and the check if the portals are accessible, by that it means that in my game, the portals already exist in the world and that when the player fires to a portal wall, the portal moves over to that wall, but the other one is still below the map. This system prevents the player from accessing that area.

The Portal System Continued



The Portal in Action

I have Listed a video Link () to showcase how everything works in game, and here are some images to demonstrate this system.

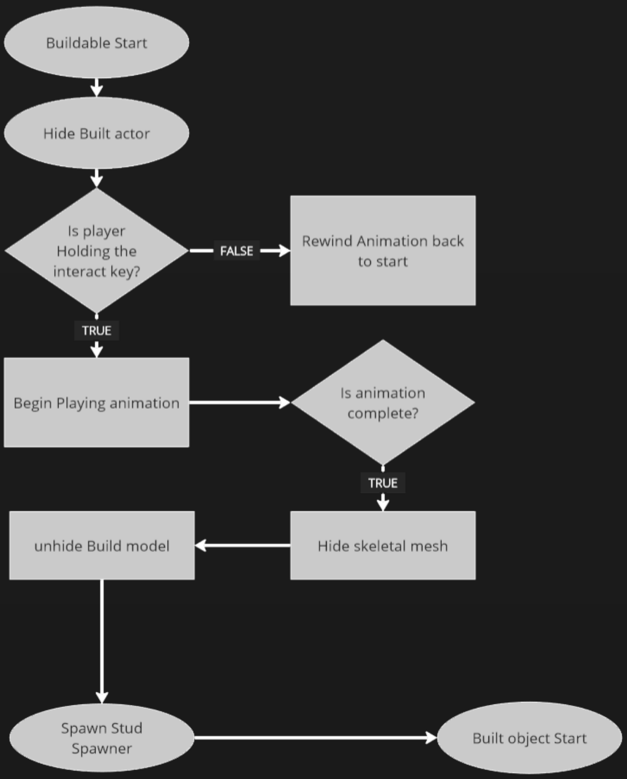


Systems And Diagrams

System 4 – The Build Mechanic

This system is a must have for Lego games or else it doesn't meet the purpose of being a Lego Game. In my game the Player would walk over the buildable object, either it being a Companion Cube or Pressure Plate. As long as the player is holding the interaction Key, the object would play the building animation. Once the animation is complete, it will remove the Skeletal Mesh that the animation is using and replace is with the built static mesh version. During my final time testing and documenting I had soon discovered that some Lego games had it so the animation would just pause when stopping in the middle of building whereas my one reverses everything back, I feel that I will change this to the other method because after testing it looks better that way.

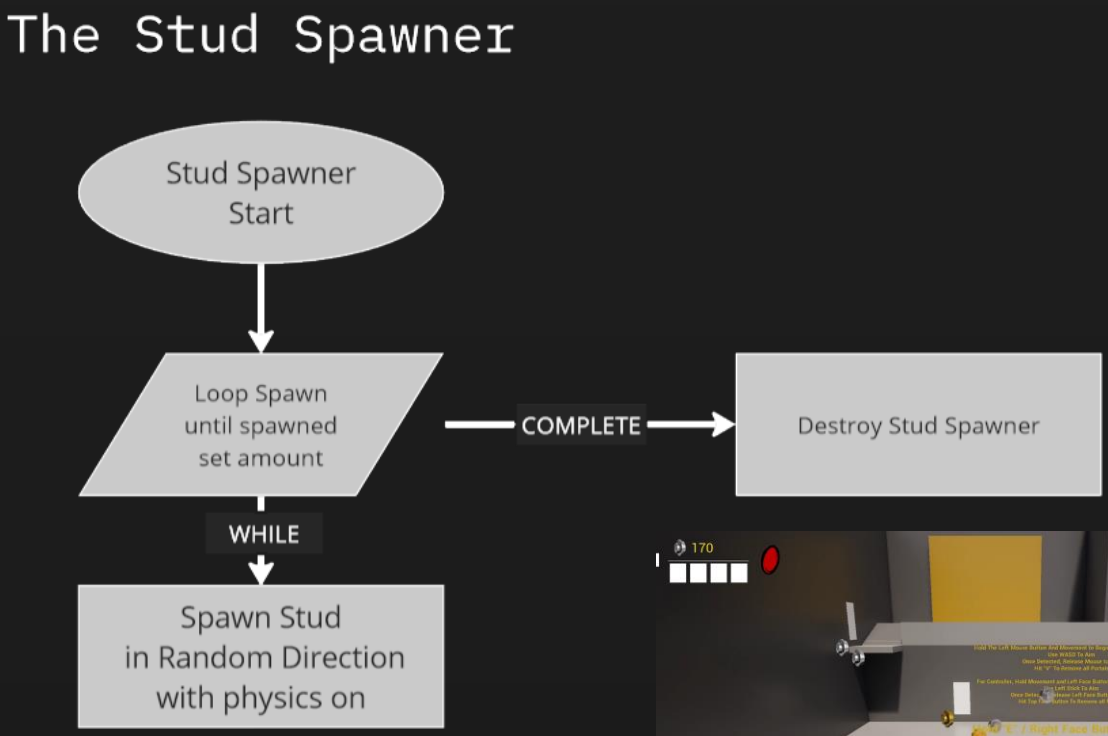
The Build Mechanic (The Pressure Plate & Companion Cube)

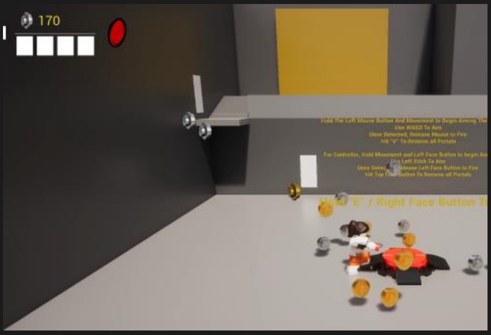


Systems And Diagrams

System 5 – The Stud Spawner

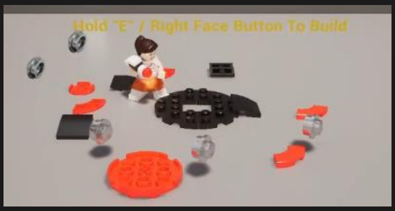
The Stud Spawner is used when the buildable objects are complete. In the future when I implement destructible objects, I can incorporate this stud spawner with it to make studs appear each time you hit the object, until it explodes.

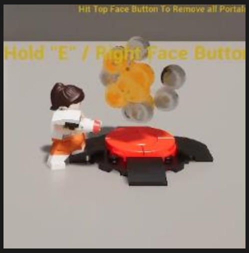

The Stud Spawner
Stud Spawner Start
Loop Spawn until spawned set amount
COMPLETE
Destroy Stud Spawner
WHILE
Spawn Stud in Random Direction with physics on



Here we can see it in Action:

The Studs will spawn outward and when the player collides, it will move to the UI



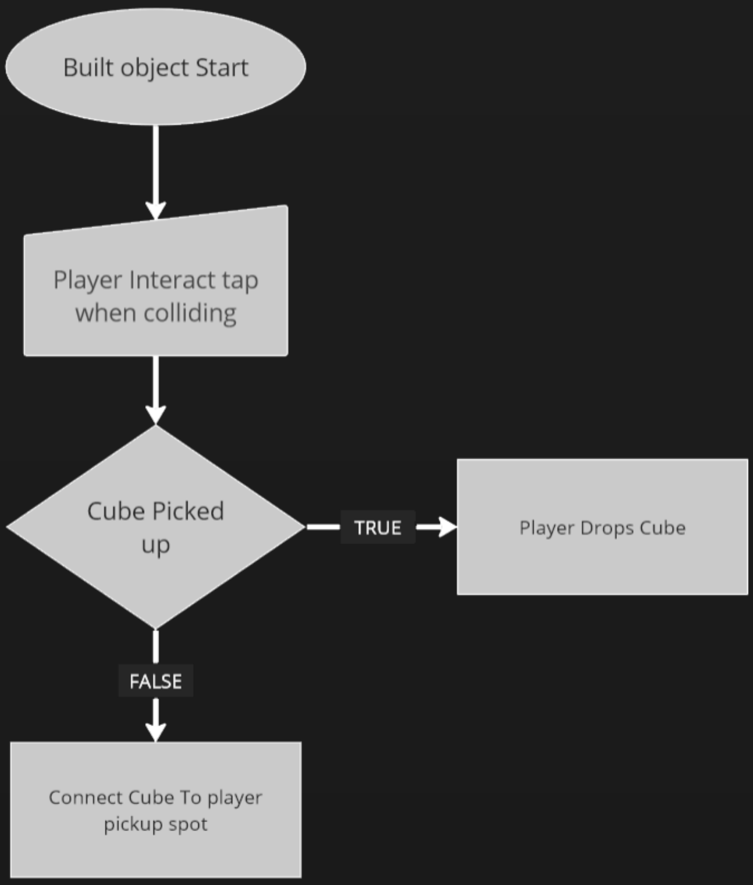

Hit Top Face Button To Remove all Portals
Hold "E" / Right Face Button

Systems And Diagrams

System 6 – The Built Companion Cube

Connecting to the Build Parent, the Companion Cube has a system in play for when it is built or else there would not be much a reason to build the object. The cube can be picked up by the player and dropped, this allows the player to move the cube to locations. In the future, I plan to add systems to prevent the cube from going through walls while it is picked up as the player could mistakenly pick it up and drop it through a wall and it would be gone.

The Built object (Companion Cube)

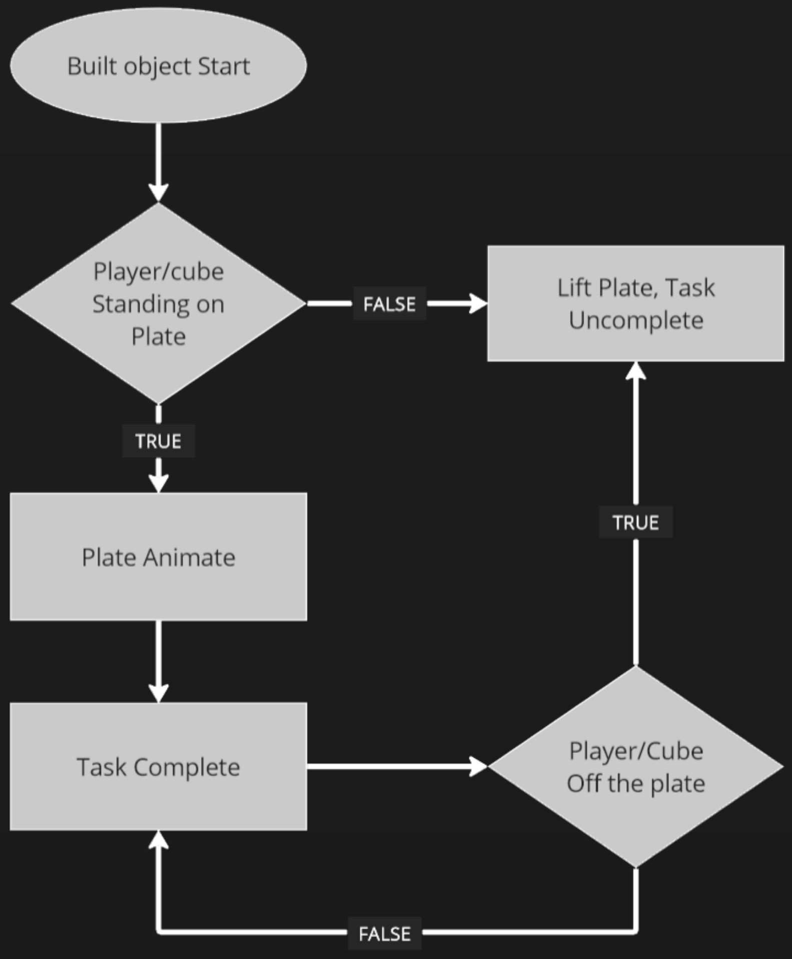

Built object Start
Player Interact tap when colliding
Cube Picked up
TRUE
Player Drops Cube
FALSE
Connect Cube To player pickup spot

Systems And Diagrams

System 7 – The Built Pressure Plate

Similar to the Companion Cube, this object would also follow after the build system and once built the plate would detect the player and the cube and if it detects them standing on the red centre, the plate will drop down and activate any connected actor it is linked to. If the player or companion cube is not on the plate, the plate will return back up and deactivate the object.

The Built object (The Pressure Plate)

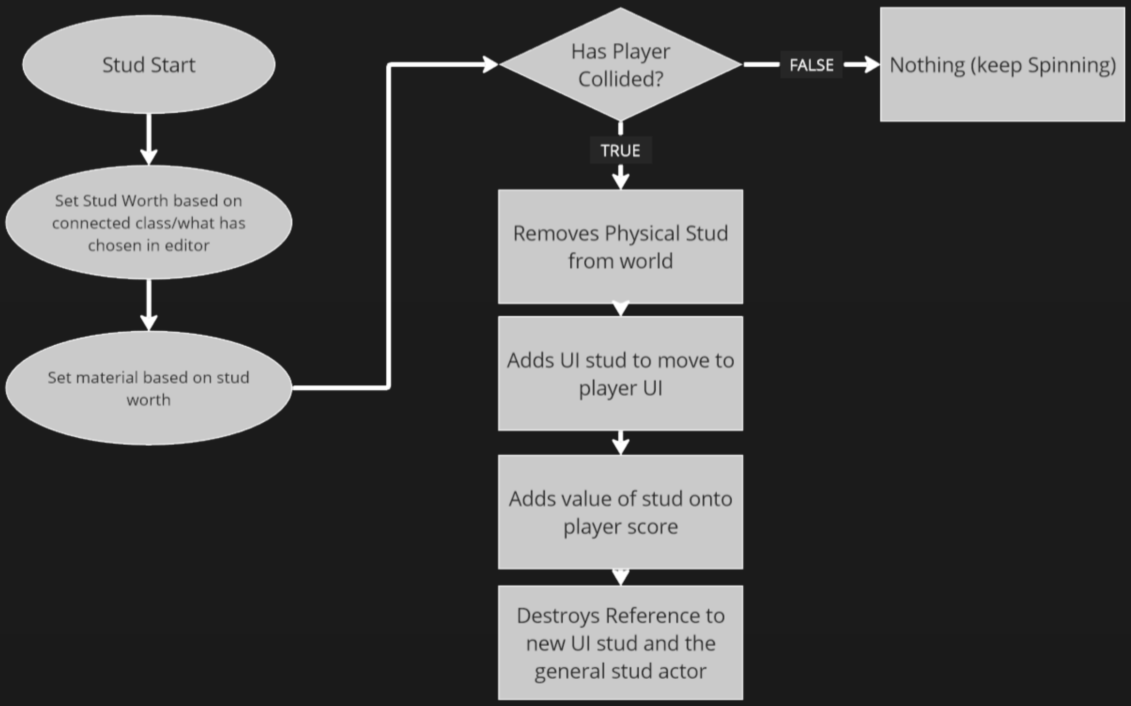

Built object Start
Player/cube Standing on Plate
FALSE
Lift Plate, Task Uncomplete
TRUE
Plate Animate
TRUE
Task Complete
Player/Cube Off the plate
FALSE

Systems And Diagrams

System 8 – The Studs

Going back from the Stud Spawner, The Studs are the object that spawns from them or are in the level thanks to splines. The studs have a rotating movement component that will allow the actor to spin like the Lego games/UI. The Stud gets its worth from either the editor if it was to be spawning from the spline or in singles, or the Stud Spawner. Once the worth has been set it would set the material of the stud to align with its worth and when collected, would create a UI popup of the stud you created into the players screen and would move over to the players score count. While that is happening, it will get rid of the initial 3D stud in the world to naturally make them move over to the UI.

The Lego Stud

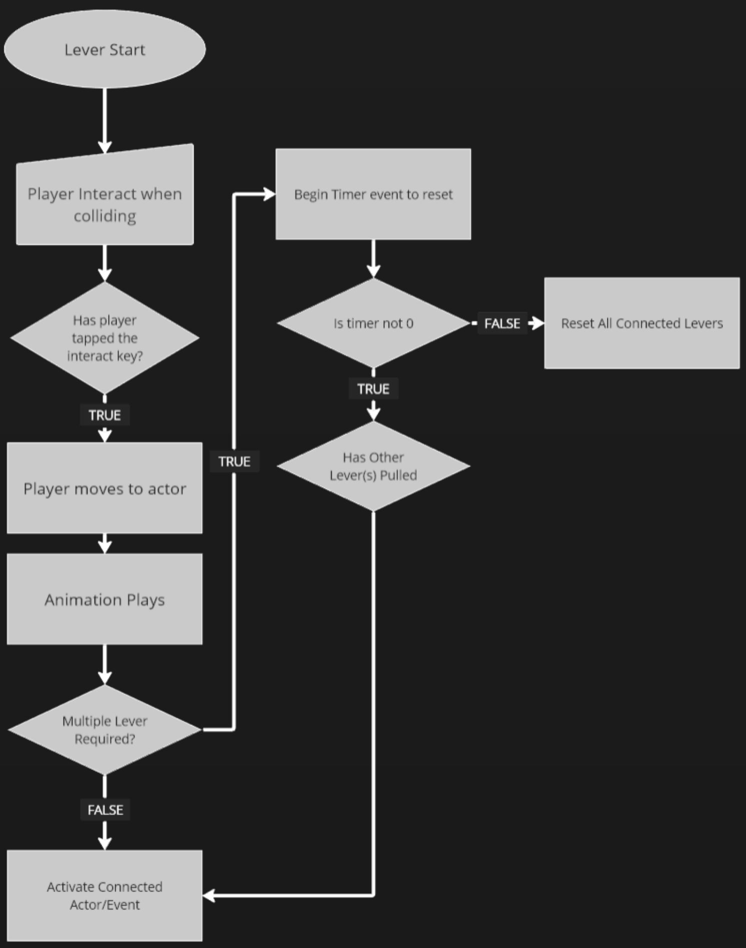

Stud Start
Set Stud Worth based on connected class/what has chosen in editor
Set material based on stud worth
Has Player Collided?
FALSE
Nothing (keep Spinning)
TRUE
Removes Physical Stud from world
Adds UI stud to move to player UI
Adds value of stud onto player score
Destroys Reference to new UI stud and the general stud actor

Systems And Diagrams

System 9 – The Lever

The Lever link many other objects like the cube and plate, follows a parent class of the Interaction System, which Is explained in the Next Slide. The level is designed to be modular for being singular use levers and multiple use levers, with possibility in the future for character specific levers. When the player taps the Interaction key, the player will move to the lever and pull it, if the lever only requires one to be pulled the task will complete to activate its connected object, if there were a need for multiple levers, a timer would begin when pulled and if you do not pull the other lever in time, it will reset all levers, but pull all levers and it will complete the task and remove the timer from happening.

The Lever

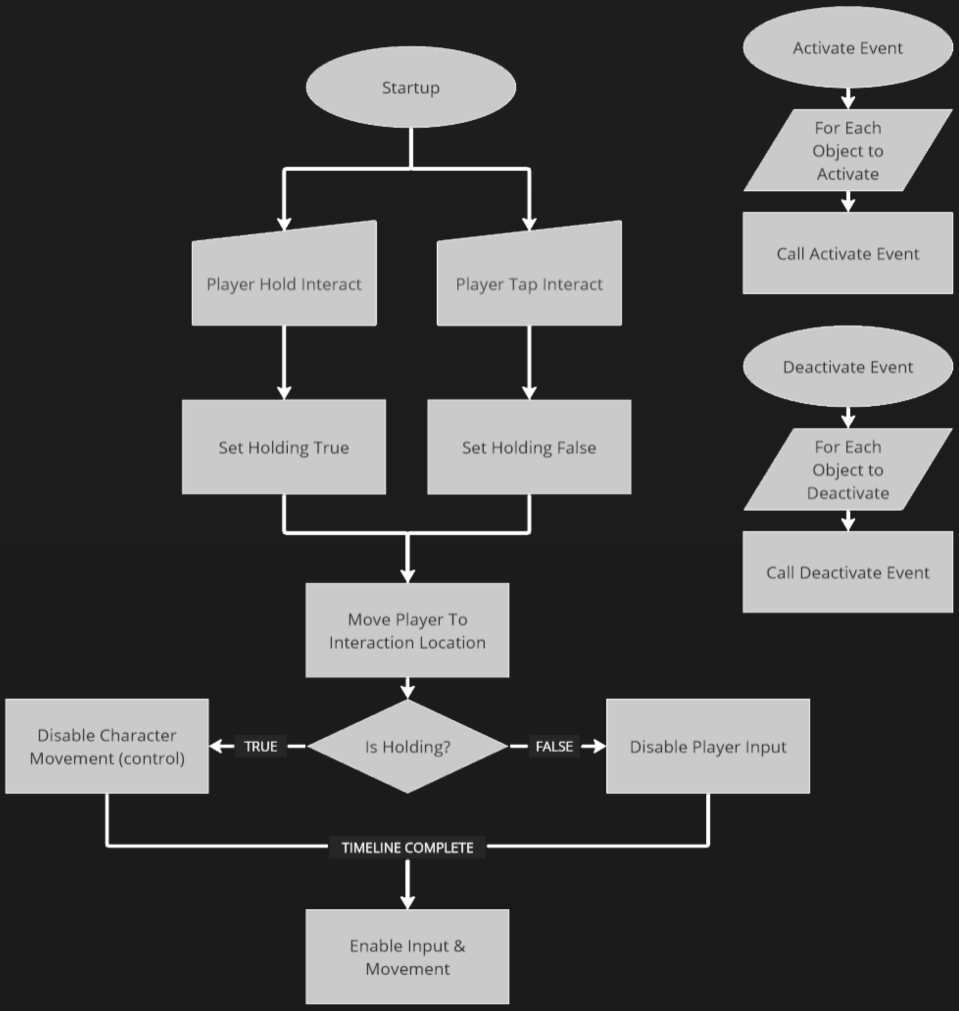

Lever Start
Player Interact when colliding
Has player tapped the interact key?
TRUE
Player moves to actor
Animation Plays
Multiple Lever Required?
FALSE
Activate Connected Actor/Event
TRUE
Begin Timer event to reset
Is timer not 0
FALSE
Reset All Connected Levers
TRUE
Has Other Lever(s) Pulled

Systems And Diagrams

System 10 – The Interaction System

As mentioned in the previous slide, most of the objects share a parent Interactable class, this is to save time when adding events for moving the player over or whether to detect when the player taps or holds the interaction key. If the player were to tap or hold on a child class that uses either event or even both, it would normally move the player to a target spot, in the instance of the build mechanic, this follow spot would be spinning around the buildable recreating the classic building system (Brown, L., 2024f) of the older series.

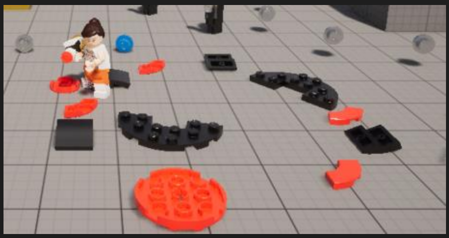
The Interaction Parent

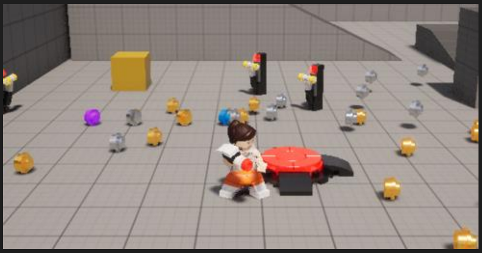


Diagrams in Action

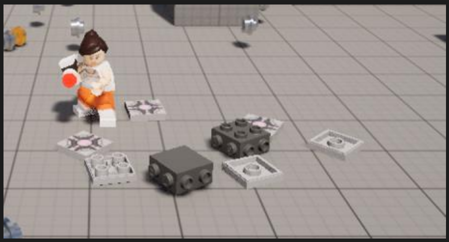
The Diagrams' Showcase screenshots that I could not fit in their slides are present below:

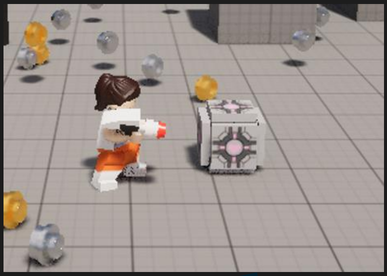
System 7 – The Pressure Plate

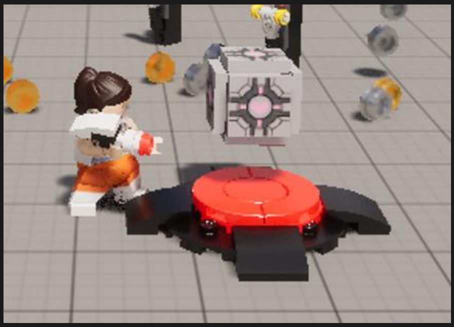




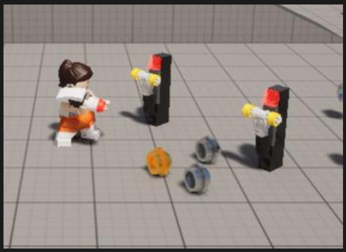
System 6 – The Companion Cube

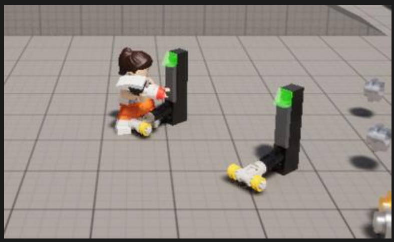






System 9 – The Lever





Optimisation and Profiling

Profiling Systems

Ensuring that systems are all working comes thanks to the Breakpoint system that Unreal presents. I will always use these if I want to understand what is happening in my scripts and ensuring that what needs to be called is called.

I will often also use Debugs such as Print string or Debug draw for when I wish to incorporate collision checks, as my project does feature this, in the build these do not get shown as when I build the game, I set it to a mode where Debugs would not normally appear.

There was an issue throughout my project where debug draws would not display causing me to find other outlets for testing collision detections. The Aiming system had been one of the major struggles to implement due to this and it is why the final build features the Arrow when aiming, initially it was planned to be a UI based aiming system though problems with the draw debug and other perspective screen size-based problems cause several delays causing me to find another method for aiming the portal gun, which a suggestion from a lecture was what I followed and had a raycast line from the player which has that arrow to see where around would the raycast be detecting. This method worked though I still feel that in the future I will be changing it to the UI method as it was very close to working the way I wanted it to.

Profiling Graphics

When it comes to the Graphics side of things, it more commonly is used when I have any form of assets placed within my game, or that I use particles and such. I personally still need to learn this area more for my future as it is still an area that I do not work much on causing games of mine to feature performance drops when I have any assets in the game, the project does feature this minor problem though it is not as offensive as my previous projects and can run well enough on a mid-grade PC.

Coding Standards

Programming Standards

To ensure the project is easy to understand and easy to locate everything, I ensure that I keep to a naming convention for the project's variables and actors. Actors that are creating in Blueprint would have the name structure of “BP_Example” whereas Interfaces would have “BPI_Example”. Other naming conventions would be the UI “WBP_Example”, or my variables within these would have Floats have a capital ‘F’ in the start of each float variable and this would follow through with the other components and variables. Folders would also follow a structure to allow me to find out what actors/UI/maps etc. would go where.

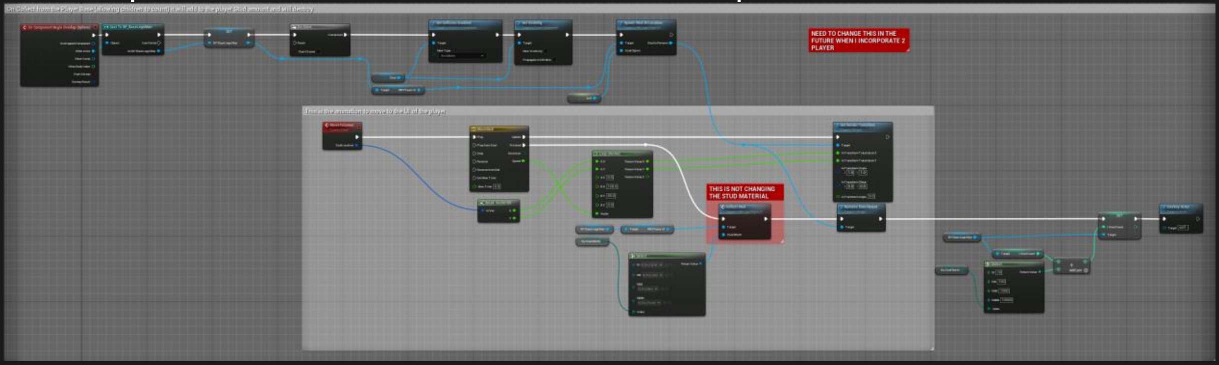
Style Guide

How I make my scripts clean and easy to read

I often have two methods to ensuring the blueprints are clean to my perspective. I used Comments to help describe what these blueprints are doing, sometimes making multiple comments to explain the path of what happens.

I also ensure I layout my blueprints in a clean manner ensuring everything is lined up correctly and relatively spread to let the blueprint breath and not overcrowd any reader.

Following these two methods ensures that my blueprints can easily explain what they do, are easy to locate and read clearly. Below is an example of what I consider a clean blueprint.


On Collect from the Player Base (allowing children to count) it will add to the player Stud amount and will destroy
NEED TO CHANGE THIS IN THE FUTURE WHEN I INCORPORATE 2 PLAYER
This is the animation to move to the UI of the player
THIS IS NOT CHANGING THE STUD MATERIAL

Coding Standards

Commenting Rules

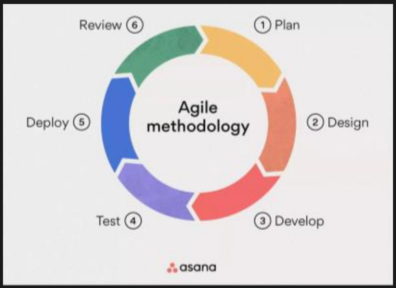
I mentioned before how I comment on my blueprints to help the designers, and such to understand what does what in my blueprints and to help them understand if there is anything in the blueprints that they feel are unfinished, broken or working.

I colour code any comment that I feel would be unfinished or broken, often Yellow being the sign that something is incomplete and Red being that there is a problem with text that helps explain what I mean. White is commonly used by me to showcase that a script is working as I normally explain in that comment what it does and not any issues and such it may have. The image in the previous slide showcases my commenting method with an inclusion to my red problem-based comment.

Code Review Procedures

Following a similar structure to the Agile method of development (Laoyan, S., 2022), Programming mechanics falls into a similar structure of planning, programming, testing, and review. When a blueprint does not work I would Breakpoint check and Blueprint Debug my way through to finding the problems that are present in my actors. After I find a probable fix, I test again and see if it improves or get worse.

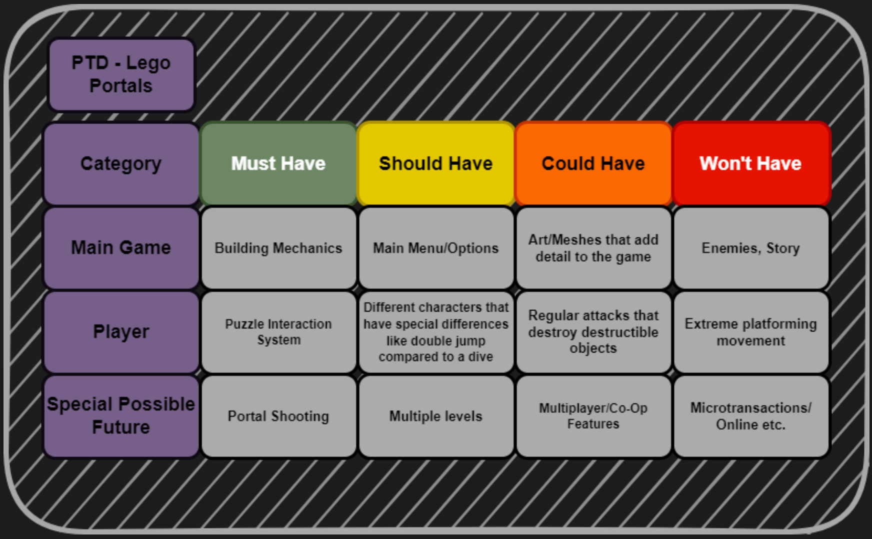
One thing I had also learnt with this project is that each build can play differently if not optimised correctly or restricted in ways, as for example, my build system had the player rotate perfectly on my own device but when I moved to a much more advanced device, it flew around the build, adding frame rate caps and restricting that spin to not be called every frame allowed a fix to prevent future problems in the final build.



Production Overview

Moscow

The Moscow is a great method to plan out a project as it perfectly lets the creator think what is the most necessary thing to add for different subjects and even down the line of should, could and won't have for the project. It is great at showcasing to someone who hasn't seen the project understand exactly the type of content your project would have or not.



PTD - Lego Portals

Category	Must Have	Should Have	Could Have	Won't Have
Main Game	Building Mechanics	Main Menu/Options	Art/Meshes that add detail to the game	Enemies, Story
Player	Puzzle Interaction System	Different characters that have special differences like double jump compared to a dive	Regular attacks that destroy destructible objects	Extreme platforming movement
Special Possible Future	Portal Shooting	Multiple levels	Multiplayer/Co-Op Features	Microtransactions/ Online etc.

Timeline

For extra guidance in keeping on track for tasks at hand, I use two other tools to guide me. I use Trello (Brown, L., 2024c) to help create a task list of what mechanics/art/animation etc. my game would feature and not always do they have to be stuff that is guaranteed to be added as I can always leave them for future purposes. A Gantt chart (Brown, L., 2024a) is different because it is more specifically used to keep a time frame of when these tasks should be completed. Trello would allow me to add subtasks and explanations to them whereas to keep space clean, my Gantt chart would be used to plan out the task but only keep the title of a task.

Budgeting

Budgeting for the project would be to see the length of the project and create a plan, the Gantt chart and Trello would be the start of that, and you would follow up with finding out how well you believe you could create all the tasks from most important to least within the time frame you are given.

My project does not feature anything that cost me financially, but it does feature assets that were made Using multiple Tools, First, there was Mecabricks (Mecabricks, 2024) which was a tool used to help create Lego Models for my game, I followed guides on how it all worked and eventually had to also learn how Blender works so that I may create animations and export these Lego builds into Unreal Engine.

Reference List

Brown, L., 2024a. Gantt Chart. [online] login.microsoftonline.com. Available at: <https://staffsuniversity-my.sharepoint.com/:x:/g/personal/b017629l_student_staffs_ac_uk/EZshrgCLZxpJq76aJrsIOwoBcGHF65GSvcPfjwR1-wJQdw> [Accessed 1 March 2024].

Brown, L., 2024b. Lego Portals Form. [online] forms.office.com. Available at: <https://forms.office.com/e/jiXVM6K7Qi> [Accessed 1 March 2024].

Brown, L., 2024c. Lego Portals Trello Board. [online] trello.com. Available at: <https://trello.com/b/K4Bt24sn/lego-portals-ptd> [Accessed 1 March 2024].

Brown, L., 2024d. Project Lego Portals Miro. [online] miro.com. Available at: <https://miro.com/app/board/uXjVN5Z4-bI=/?share_link_id=716218909060> [Accessed 1 March 2024].

Brown, L., 2024e. PTD - Final Gameplay Video. [online] www.youtube.com. Available at: <https://youtu.be/zoDBCeyKoiI?si=QvU916J_j8k6FIlA> [Accessed 1 March 2024].

Brown, L., 2024f. PTD - Lego batman gameplay elements. [online] www.youtube.com. Available at: <https://www.youtube.com/watch?v=ylKbgG6vm_s> [Accessed 1 March 2024].

Laoyan, S., 2022. What Is Agile Methodology? (A Beginner’s Guide). [online] Asana. Available at: <https://asana.com/resources/agile-methodology>.

Mecabricks, 2024. Mecabricks.com. [online] www.mecabricks.com. Available at: <https://www.mecabricks.com/>.

Wikipedia, 2010. File:360 controller.svg - Wikipedia. [online] commons.wikimedia.org. Available at: <https://en.m.wikipedia.org/wiki/File:360_controller.svg> [Accessed 1 March 2024].

Wikipedia, 2024. QWERTY. [online] Wikipedia. Available at: <https://en.wikipedia.org/wiki/QWERTY#/media/File:KB_United_States.svg> [Accessed 1 March 2024].