FACULTY OF SCIENCE, ENGINEERING AND COMPUTING

School of *Computer Science & Mathematics*

BSc DEGREE

IN

Computer Games Programming

PROJECT REPORT

Name: Tuchan Anthony

ID Number: K1917661

Project Title:
The Last Arena VR

Date:
08/04/2022

Supervisor:
Dr. Jarek Francik

Kingston University London

Plagiarism Declaration

The following declaration should be signed and dated and inserted directly after the title page of your report:

Declaration

I have read and understood the University regulations on plagiarism and I understand the meaning of the word *plagiarism*. I declare that this report is entirely my own work. Any other sources are duly acknowledged and referenced according to the requirements of the School of Computer Science and Mathematics. All verbatim citations are indicated by double quotation marks ("…"). Neither in part nor in its entirety have I made use of another student's work and pretended that it is my own. I have not asked anybody to contribute to this project in the form of code, text or drawings. I did not allow and will not allow anyone to copy my work with the intention of presenting it as their own work.

Date 11/04/2022

Signature T.Anthony

Contents

Introduction

Summary Of The Game:

The Last Arena VR is a Single player VR Action game that will allow the player to fight with a range of different fighting styles. The player will be able to use a variety of different weapons to help them survive the waves of enemies that are out to get them in the arena. Each weapon will give the player a unique way of playing for example, the dagger will allow them to do decent damage and allow the player to escape enemies by teleporting to the dagger whilst the two handed sword gives them a good amount of damage and allows the player to block enemy attacks.

Aim Of The Game:

The aim of the game is for the player to survive the wave of enemies for as long as they can if they manage to survive the wave of enemies for long enough then they will be crowned as champion of the arena which means they have completed the game.

Objective Of The Game:

The objective of the game is to gain victory by beating the wave of enemies.

Once the player go through the Start Game door in the hub world they will be put into an arena where they will have to fend off waves of enemies using a variety of different weapons that are placed on the floor around the arena. The player should try to use the weapons effectively, by effectively I mean using the weapons in the right situation for example, using a bow and arrow for when the enemy is far away and then when the enemy is close to the player switch to a close ranged weapon such as a dagger or a sword.

If the player survive the first few rounds then they will be able to unlock spells that will give them a heavy advantage in battle but they must use the spells wisely because they will only get one for each wave and depending on the spell it is recommended that they use them when enemies are grouped up together so you can weaken all of them and beat them easier.

Personal Aims And Objective

The reason for creating The Last Arena VR is because I wanted to expand on my game programming knowledge by trying to create a VR game something that is somewhat new and is still being improved in the gaming industry. I also wanted to see how well I can adapt to creating something that I have never touched upon before and I also wanted to improve on my researching skills by creating this game.

Summary Of What Is To Come

This report is divided up into different section that was vital to the creation of the Last Arena VR. The first section of the report is the state of the art review, in this section we have a look at all the things that influenced The Last Arena VR and it also discusses the tools that will be used to create the game. The next section is the Analysis section which discusses all the requirements that are going to need to be met to complete the game. The third section is the Design section where it will discuss the conceptual design for the levels, the UI design and the Technical Design of the game.

After that the Implementation section comes in where it will discuss all of the important mechanics of The Last Arena VR and they will be discussed in a way where the reader could get some ideas on how to implement these features themselves.

The next section is the Testing and Evaluation where all the testing results and the evaluation of the Moscow analysis along with the functional and non functional requirements will take place. The next section is the Legal, Social, Ethical Implications section where the discussion of who the target audience of the target audience of The Last Arena VR will take place.

The final section of this report is the Critical Review section where I give my thoughts on the project and discuss my achievements.

State Of The Art Review

Review Of Similar Games

Gorn

(Figure 1) Gorn - Feltham, J., 2021. Gorn Review: A Sublimely Silly Bit Of VR Violence. [online] UploadVR. Available at: <https://uploadvr.com/gorn-review/> [Accessed 20 October 2021].

A game that heavily inspired The Last Arena VR is a game named Gorn which is a violent VR gladiator simulator which was developed by Free Lives. In Gorn you are thrown into a coliseum where you will have to pick up a weapon that is on the floor, once you have picked up that weapon the gate that is dividing the player and the enemy will open. Gorn also has a wave system where once you defeat the current enemies that can be seen new ones will come in and new weapons also drop the longer you survive.

Gorn has very good physics mechanics but it is very violent, the player can chop off any body part of the enemy just like how someone would in real life .The player is able to break an enemies legs with a mace by repeatedly smashing the legs until it comes off and the most gruesome one of them all is that you can rip off your opponents head with your bare hands. Gore also uses its physics in a

comical way where the enemy will be wobbling around whilst walking towards the player and swinging its weapon in the air.

There are many features that make The Last Arena VR and Gorn similar games, the main one being how both games are a beat em up arena fighting game where the players need to beat the current waves of enemies to progress through the game. Another feature is how there are more than one weapon choices in the game, the only difference is that in Gorn the weapons are based on older weapons like a club with nails in whilst The Last Arena VR has more modern weapons such as daggers.

Hellsplit: Arena

(Figure 2) Hellsplit: Arena - Arena, H., 2021. Hellsplit: Arena on Steam. [online] Store.steampowered.com. Available at: <https://store.steampowered.com/app/1039880/Hellsplit_Arena/> [Accessed 20 October 2021].

Hellsplit Arena is a VR horror slasher game developed by Deep Type Games where the player has a few unique maps to choose from where they can from there defeat the waves of enemies. Hellsplit Arena is very similar to Gorn in the sense it's a VR combat game where you are fighting waves of enemies but in this game, you can play in multiple maps such as a courtyard of a castle or in a snowy field, to move around the player is able to ride on a horse.

Before fighting the waves of enemies, you have a range of settings that you can adjust such as setting up the types of enemies they want to fight, the weapon they want equipped and the armour they want equipped. Hellsplit Arena also has an adventure aspect of it where you can explore the maps to find enemies to defeat, this adventure feature could be added to the proposed game idea and if this feature is added then it also adds the possibility of mounts being added.

What Makes The Game Fun?

The Last Arena VR will be fun because the player will be able to be fully immersed in battle since instead of pressing buttons on a controller to fight, they will actually have to do the fighting using their own body to do actions such as swinging their weapons. To be even more immersed the player

will also have a character body that will match there real life movement and there will be mirrors in the game so that the player can see their character.

Another thing that makes the game fun is that there is a bunch of different fighting styles for example, you will be able to use a bow and arrow for when an enemy is far away and then when the enemy is closer the player can switch to a closer ranged weapons such as a sword or daggers.

If different types of enemies are added into the game then the player will also sometimes be forced to adjust their fighting styles in order to beat an enemy for example, when they are going up against an enemy that has a shield they will have to use an axe that can break down the shield before they can actually deal good amount of damage.

If spells are implemented into the game then that will be another fun aspect of the game because it means that players will be able to use spells in the game that will all have different effects such as the ice spell being able to freeze enemies and the fire spell which does a lot of AOE damage but to make it so that the spells aren't too strong some enemies will be resisted to certain spells which will reduce the damage taken to them but certain enemies can be weak to some spells resulting in more damage being done to them.

Tools and Technology (Game Engines, Programming Languages etc):

The Last Arena VR will be made on Unreal Engine 4 using a combination of Blueprints and C++ scripts although the other game engine Unity would be easier to create the game in the decision to go with Unreal Engine was decided because Unreal Engine is a powerful engine that is challenging but fun to use.

Unreal Engine also has features that help with making a VR Game such as setting up motion controllers and being able to easily view my game in VR by opening the VR Preview mode and viewing it through the headset. Also since Unreal Engine is used by some big games these days it is also a good idea to start getting used to it for when its time to work at a games company that uses it.

Artificial Intelligence

The Last Arena VR will have very little AI since the focus of the game is demonstrating the combat in VR. The Ai will mainly be for the enemies where they will be able to know where the position of the player is as soon as they spawn in. Once the Enemies have gotten close enough to the player they will attempt to attack them, the enemies will also react to being hit by the weapons of the Enemy and will lose health depending on the weapon they got hit by.

Analysis

The State Of The Art review mentions the different types of Game mechanics and Game engines that can be used to create The Last Arena VR. Keeping in mind the State of the Art Review section all the final decisions are decided in this Analysis section where a list of what is needed is created in the order of priority in a MOSCOW list (Must Have, Should Have, Could Have and Wont Have). In this section the Development Methodology that will be used for this project will also be decided here along with how the project will be approached.

Game Features:

- **VR Combat** – Combat will be in VR so to attack with the sword the player will actually have to swing their arms like they have a sword in their arms
- **Variety Of Unique Weapons** – Each weapon will be unique in their own way for example, An axe will require the player to use both hands to hold the axe but it can destroy the shields of enemies, whilst the bow and arrow requires 1 hand to hold the bow and the other hand is to be used to put the arrow on the bow and to launch the arrow.
- **Waves of Enemies** – Enemies will come in groups from somewhere in the arena and once the group of enemies have been defeated a new wave of enemies will appear.
- **Picking up Weapons with VR** – Wield weapons of your choice by picking them up from the floor with the VR controllers.
- **Different types of Enemies** – Some enemies will have shields that need to be broken down by an axe, some enemies are faster than normal, etc.
- **Can Block with certain Weapons** – You can block with weapons such as a sword to reduce the damage taken.
- **Range of Unique Spells** - Fire spell that does AOE Damage, Ice Spell that freezes, etc.
- **Practice Mode** - Where you can practice using different weapons and spells.
- **Enemy Weaknesses** – Some Enemies will be weak to certain spells
- **Maps that you can explore more** – Bigger maps that the player can explore and go search for the enemies
- **Mounts** – If maps are made bigger to explore depending on how big they are, they could have mounts to ride on to make exploring quicker

MOSCOW Analysis

Must Have	**Should Have**	**Could Have**	**Wont Have**
- VR Combat - Variety Of Unique Weapons - Picking up Weapons with VR	- Different types of Enemies (Some enemies will have shields that can only be broken down by an axe) - Can Block with certain Weapons - Full body Immersion - Waves of Enemies	- Range of Unique Spells - Practice Mode - Enemy Weaknesses - Maps that you can explore more -DLC Architecture	- Multiplayer - DLC

Not everything in this MOSCOW Analysis will be added apart from everything in Must Haves and most of the things in Should Haves, the items inside the Could Have list are optional and are not really needed.

The reasoning for not having **Multiplayer** in the Last Arena VR is because this is a single player VR game that focuses on combat so it is not really needed and there most likely wont be a lot of time to implement multiplayer. If the game is successful in the future there could be **DLC** such as new weapons, maps and new enemies to fight.

Functional & Non Functional Requirements

Functional Requirements	Non Functional Requirements
The Game will allow the Player to start the game whenever they want	The Game should not crash whilst the player is playing the game
The Game will allow the Player to pick up items	The Game should be somewhat challenging for the Player
The Game will allow the Player to deal damage to the Enemies	The Game should be a fun VR experience for the Player
The Game will allow Players to move their physical bodies around to complete the game	

Development Methodology

How The Game Will Progress

The Last Arena VR will progress by first going through the MOSCOW which shows the priorities of each game mechanic starting off with the mechanics in the **Must Have** section since these are the core mechanics that will get the game working if these mechanics aren't in the game then it would be a completely different game and wouldn't follow the proposed idea at all.

The Gantt chart will also be in use to help the project be finished on time along with a checklist on sticky notes (The app on Windows) to help easily identify which tasks have already been done. The project will be divided into sprints so that one feature will be programmed into the game and then tested to make sure it all works.

After all of the features have been programmed and tested then that is when they will be implemented into the final build of the game which is also where the map will be made. Once everything has been implemented into the final build the full game will be tested to make sure everything is running fine and if a problem arises then a solution will be found.

Methodology That Will Be Used

After looking at how the Development of the game will go It would make sense to use the **Agile Methodology**, the reason for this is because according to Atlassian the Agile Methodology is faster than using the waterfall methodology due to the fact that in Agile the project is broken up into different sections. With the Agile approach the project is being evaluated all the time which means that it is possible to respond to changes as quickly as possible whilst with the Waterfall Methodology everything must happen and apart from the planning stage of the project no changes are allowed to happen, if a change is needed then they must restart the project from scratch.

Design

Conceptual Design

The Last Arena VR is going to be set in an old fantasy era where the most popular sport is gladiator battles where humans have to fight for their lives against monsters. There is no storyline to the Last

Arena VR since the basis of the game focuses on using the VR headsets for the combat and focuses on the different types of weapons. The core mechanic of The Last Arena VR will be the combat system in VR so the levels will not be too detailed since it is not a game where the player can explore.

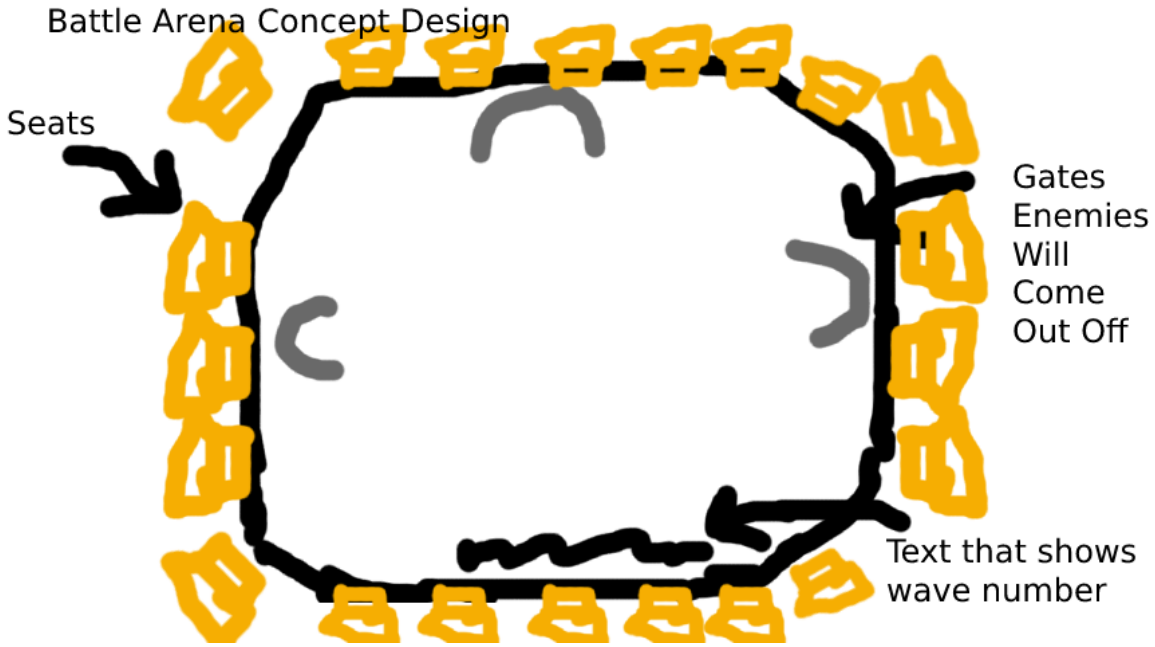
Level Design

Appearance

Since The Last Arena VR is set in an old fantasy world where gladiator battles are still popular the materials for the Arena will be quite simple for example, the walls will be made out of brick and the seats would be made out of some sort of stone, there will be no metal structures at all.

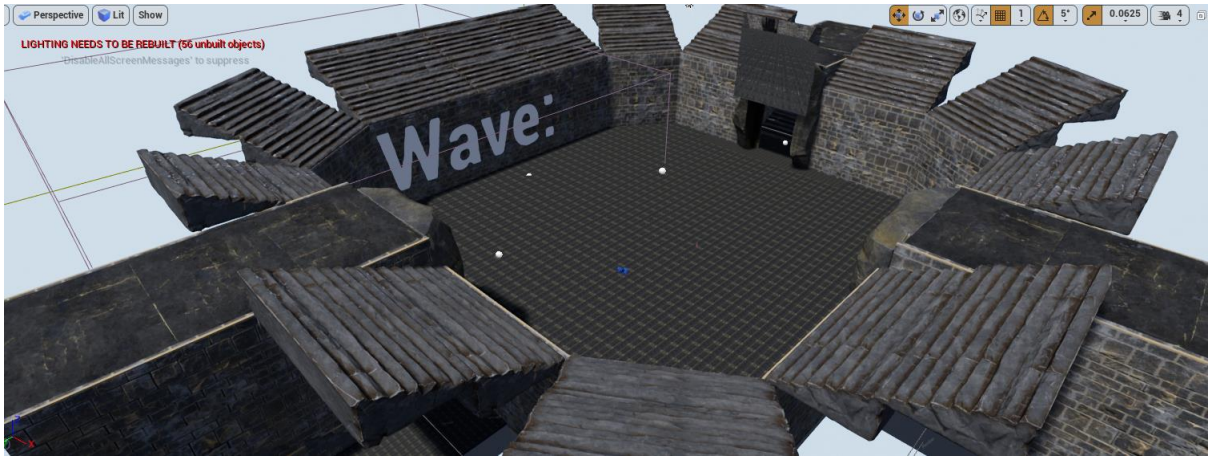
Battle Level

Since the game is all about fighting hordes of enemies in an arena the level will be similar to a stadium where there are seats for spectators, doorways the enemies will come out of and the terrain will be flat. The size of the level will be quite small due to the game being a VR horde fighting game so there isn't really any point of having the map being big where the focus of the game is to fight enemies.



(Figure 3) Concept Art of the Battle Arena For The Last Arena VR

Final Battle Arena Level



(Figure 4) Final Level Design of the Battle Arena For The Last Arena VR

Lobby Level

There will be a lobby before entering the arena which will match the appearance of the arena with the walls and the floors, there will also be a mirror where you can see the players character model which is to mainly showcase the full body immersion that will most likely be in the game. The lobby level will also have a portal where if you go through it then you are taken to the arena where the game will truly begin

(Figure 5) Concept Art of the Lobby Arena For The Last Arena VR

Final Lobby Level

(Figure 6) Final Level Design of the Lobby Level For The Last Arena VR

UI Design

In The Last Arena VR there will be quite a few UI elements most of them not being on screen. The only on screen UI element that will be in the game will be when the player gets hit by the Enemy the screen will blink red, this is so the player knows when they have taken damage. The reason for there being very little on screen Elements Is because the VR games they don't really use UI that much since it ruins the immersion that VR gives the player.

For the UI elements that aren't on the screen it will mostly consists of text that will give the player information about the game for example, in the lobby level it will explain what makes each weapons unique and if they have some sort of special ability this will be told. There will also be text in the Battle Level that will tell the player the wave number that they are on, also for the portal that will take the player to the Battle level from the Lobby level there will be text above it that says "Start game" and for the portal that takes the player to the Lobby level from the battle level it will have text above it that says "back to lobby"

How The Design Fit With The Requirements

The Design of The Last Arena VR fits well with the requirements that were listed in the Analysis section of this report because since this game focuses on nothing else but the VR combat it would be wise to make the design of the Levels simple so that the player can effectively use their weapons without anything being in the way.

Technical Design

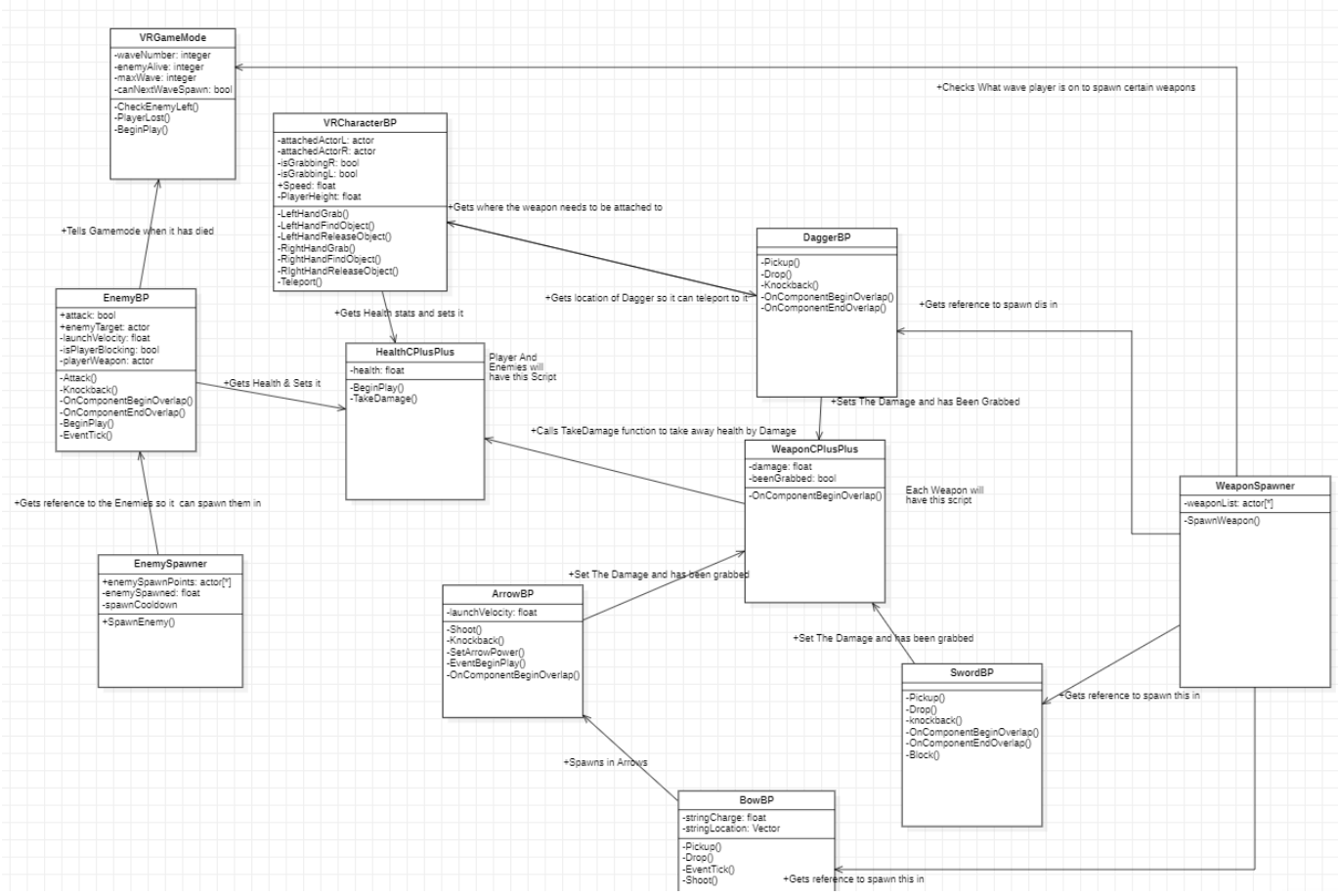
Architecture Of The Game

For The Last Arena VR most of the blueprints are going to be linked with the blueprint of the VR Character. The reason for this is because the VR Character Blueprint is what most objects in the scene will be interacting with for example, all the weapons in the game will need to attach themselves to hands of the VR player so they must have some sort of reference to the player. The blueprint for the Enemy will also need to get reference of the player so that it can deal damage to them.

Since the Enemies are AI that will track and attack the player they will have Behaviour Trees and blackboards that is provided with Unreal Engine, the blackboard will provide the variables for the

Behaviour tree and the Behaviour tree will deal with all the tasks and services that will allow the Enemy to move towards the Player and attack the Player.

UML Diagram



(Figure 7) UML Diagram For The Last Arena VR

This UML diagram showcases the relationships of the different classes that will be added to The Last Arena VR.

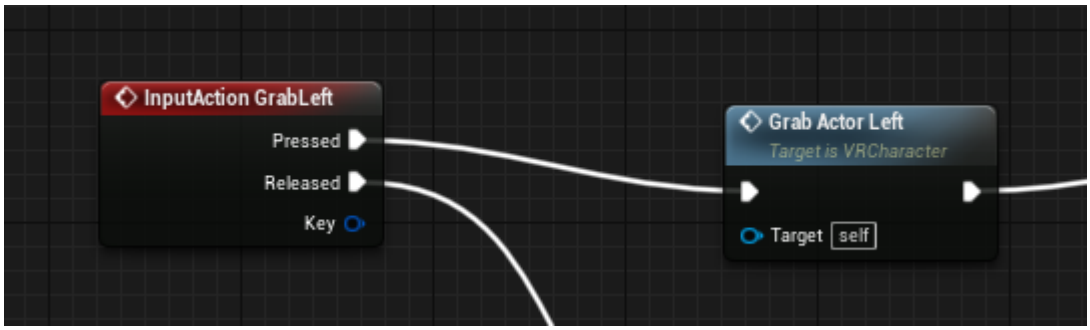
Implementation

The Last Arena VR focuses on the combat in VR so the most important mechanics that were implemented are the ones that are related to the player being in VR and how the weapons work. The Implementation section of this report will demonstrate the main features that makes The Last Arena VR what it is.

Picking Up & Dropping Weapons Mechanic

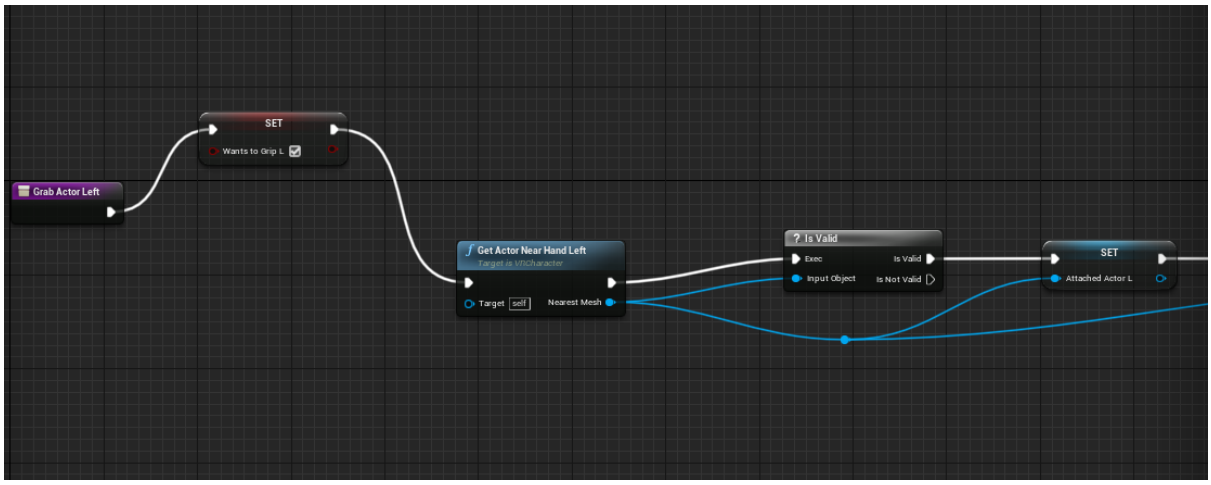
This section explains how the picking up and dropping weapons mechanic works and it will explain how each weapon is picked up since they are picked up differently.

Grabbing Function



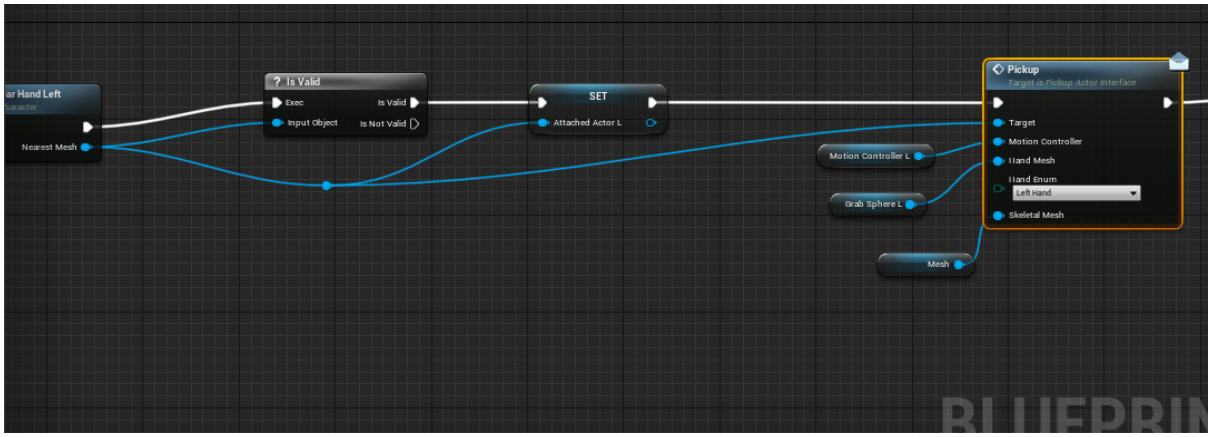
(Figure 8) Where the Grab function is called, inside VRCharacterBP

For Picking Up weapons there are quite a lot of steps that happens that first starts inside the blueprint for the VR Character, when you press the trigger on the VR controller depending on which controller you pressed it with (Left Or Right controller) a function called GrabLeft or GrabRight will be called.



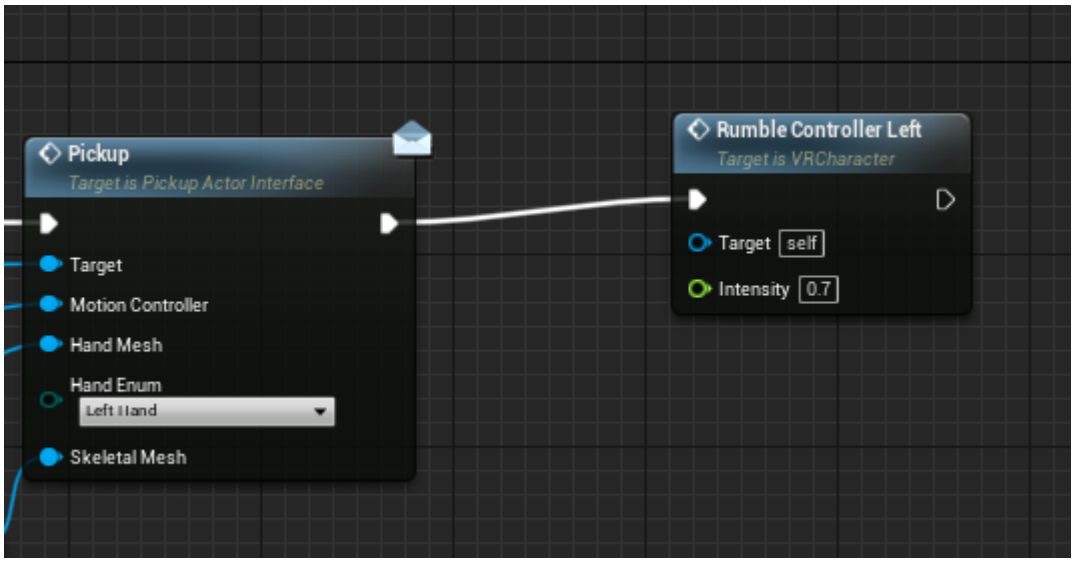
(Figure 9) Grab Actor Left Function, inside VRCharacterBP

Inside this function the first thing that happens is that a Boolean variable (Called WantsToGrip) is set to true which will basically tells the game that the player is trying to grab something, after that a function called “GetActorNearHand” will be called that will basically check what is touching the hand and it will then get reference to that object (which is a weapon in this case) and a reference to that object will be saved in a variable called AttachedActor.



(Figure 10) Grab Actor Left Function, inside VRCharacterBP

After that a function called Pickup will be called from within the Blueprint of the object and the motion controller variable of the hand that is trying to grab will be passed into this function along with the Hand mesh, the skeletal mesh and the Hand Enum which determines which hand is performing the action so that the object knows what hand to attach themselves to, once again the Pickup function will be explained later on.

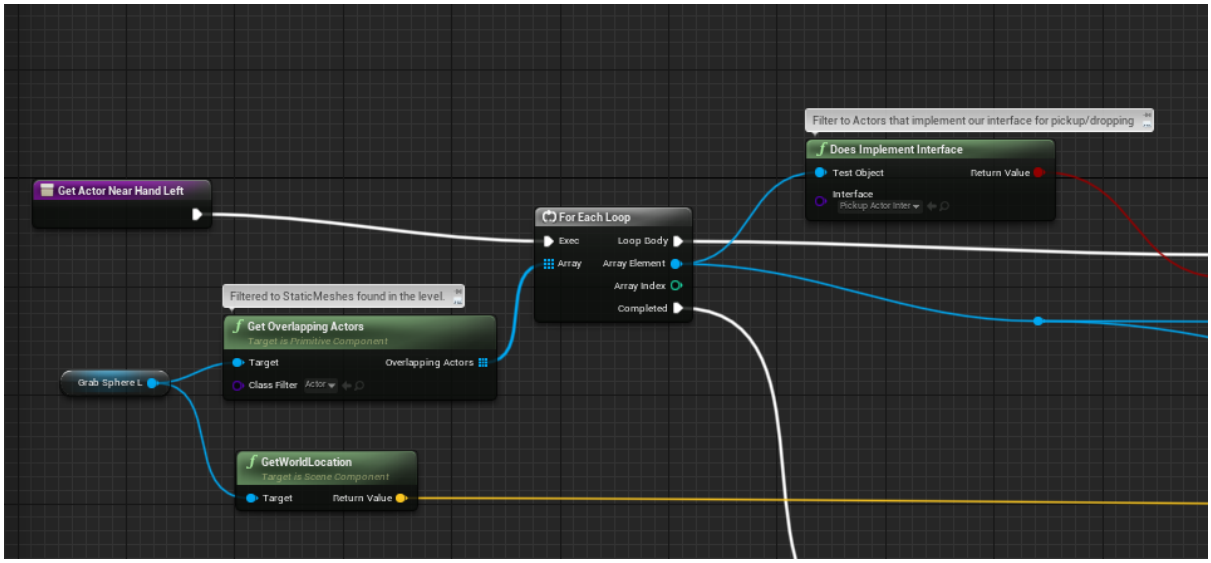


(Figure 11) Grab Actor Left Function, inside VRCharacterBP

Finally after the weapon is picked up the controller that picked up the item will vibrate.

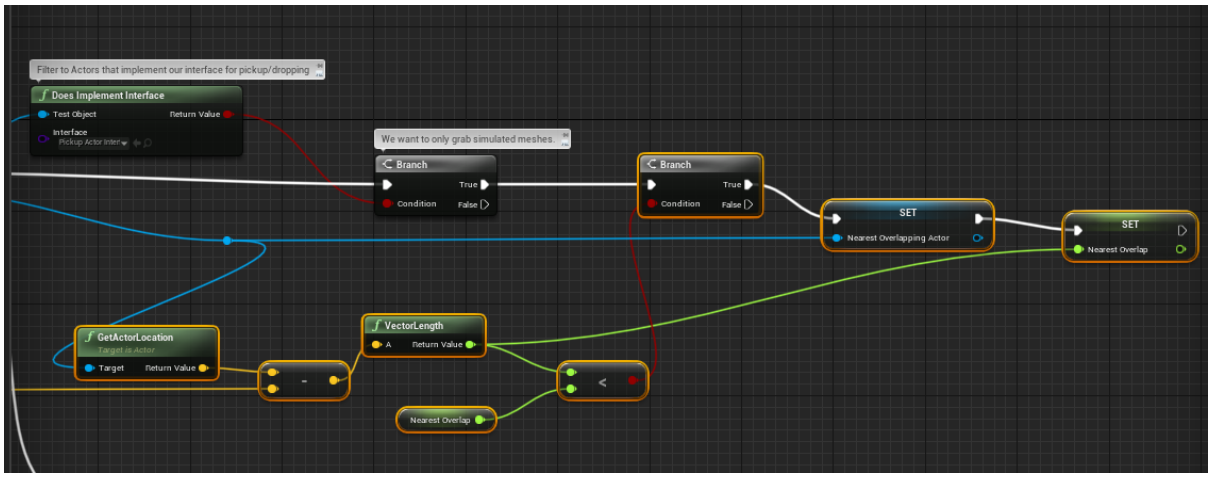
Finding The Nearest Actor

As previously stated the purpose of the “GetActorNearHand” function is to check what object the hand is touching and it will then output the reference if that object to the grab function that was explained before.



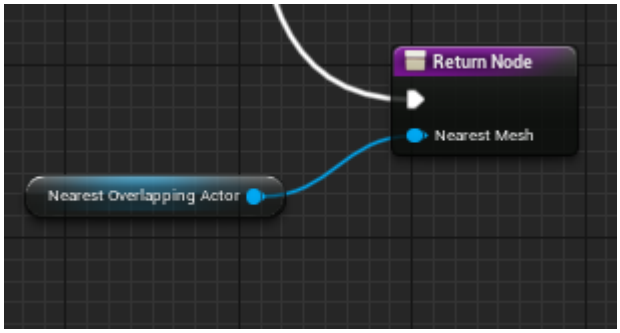
(Figure 11) Get Actor Near Left Function, inside VRCharacterBP

Inside this function the first thing that is done is a for each loop that will check all the objects of class Actor that is overlapping with the Grab Sphere which is the hitbox that detects what the player can pickup.



(Figure 12) Get Actor Near Left Function, inside VRCharacterBP

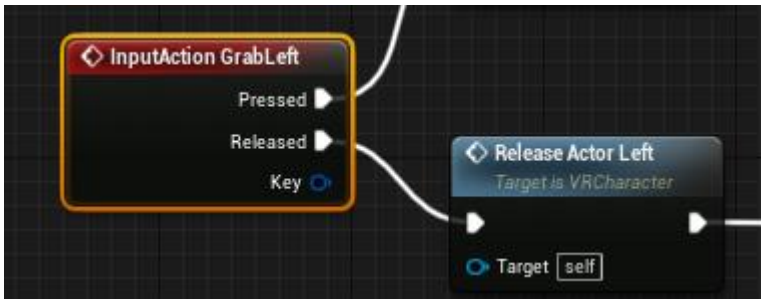
For each object it will check if the object uses the interface for picking up/ dropping (Will be explained later on) and if it does then that means that object can be picked up by the player, finally get reference to that object and set it into the “NearestOverlappingActor” variable and also set the variable known as “Nearest Overlap” that will keep hold of the distance between the object and the grab sphere.



(Figure 13) Get Actor Near Left Function, inside VRCharacterBP

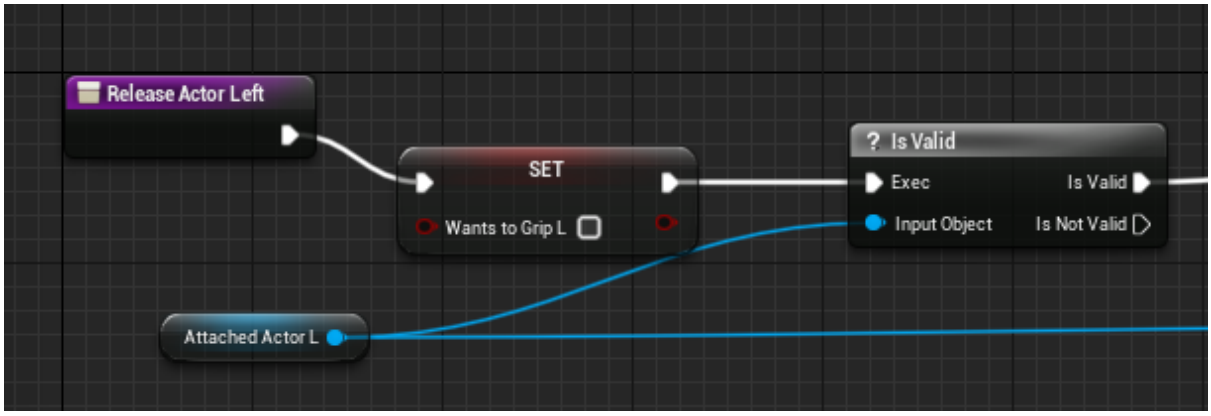
The very last thing involving this function is making sure that the "NearetsOverlappingActor" variable is returned so that the Grabbing function can use it for the Pickup function as explained previously.

Releasing Function



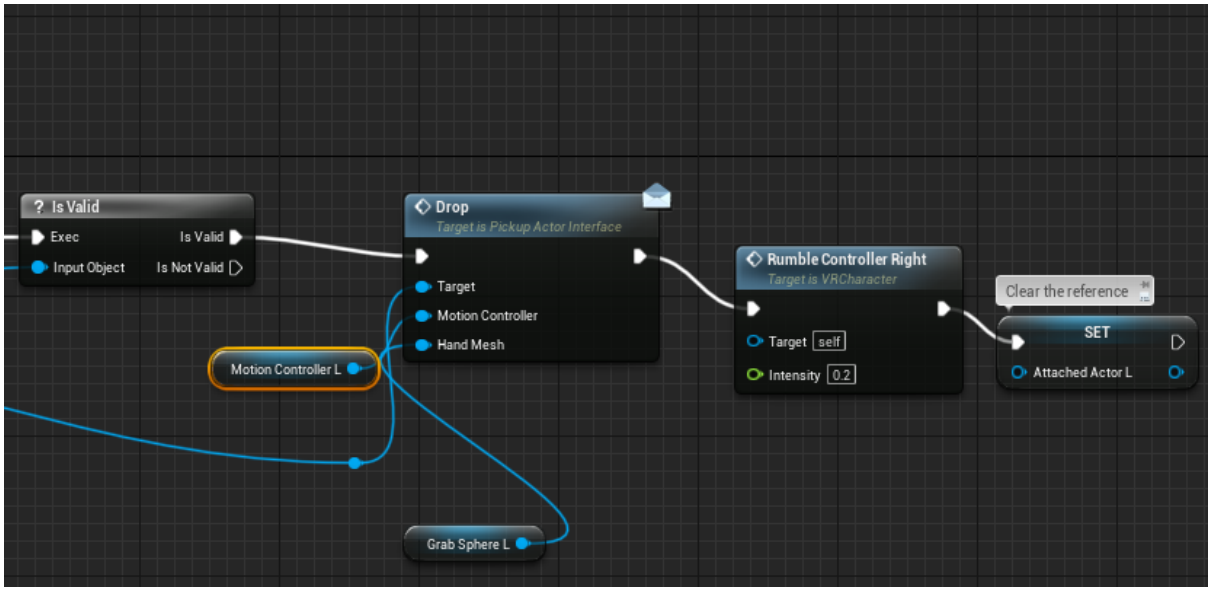
(Figure 14) Where the Release Actor Left function is called, inside VRCharacterBP

When you release the same trigger you were grabbing with then depending on the controller (left or right controller) a function called ReleaseActorLeft or ReleaseActorRight will be called.



(Figure 15) Get Release Actor Left Function, inside VRCharacterBP

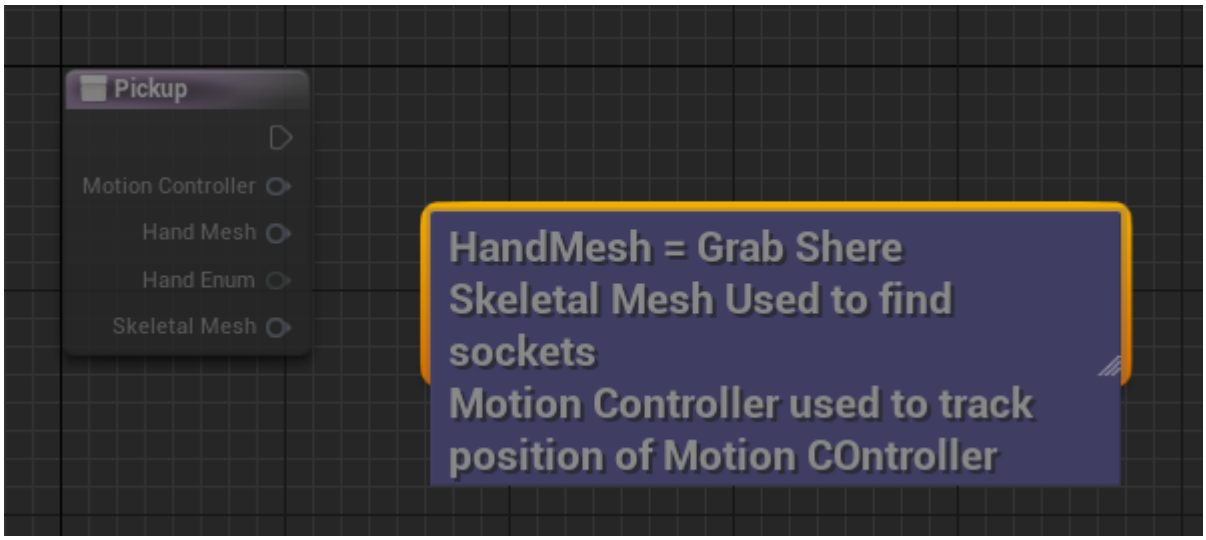
Instead of setting the Boolean that tells the game that the player is trying to grab hold of something (Called WantsToGrip) to true in this function it sets it to false telling the game that the player is no longer trying to grab holding of something and instead of checking through all the object that is touching the hand it will just check if there is anything that the player is holding onto which is stored in the “AttachedActor” variable.



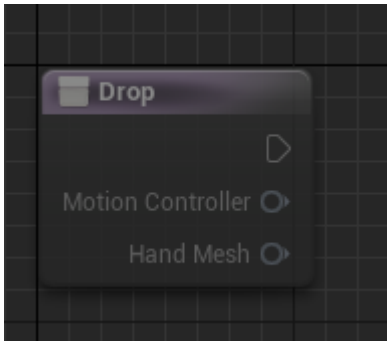
(Figure 16) Get Release Actor Left Function, inside VRCharacterBP

If there is an item that the player is holding then a function called Drop will be called from within the Blueprint of the attached object object and the motion controller variable of the hand that is grabbing onto the object will be passed into this function with the Hand mesh but this time without the skeletal mesh since it is not needed for this operation, the Drop function will be explained later on. After that the function to make the controller vibrate will be called but with an intensity lower than when an object is picked up and finally to make sure we can repeat the process with a different object we set the “AttachedActor” variable to null.

The Pickup/ Drop Functions

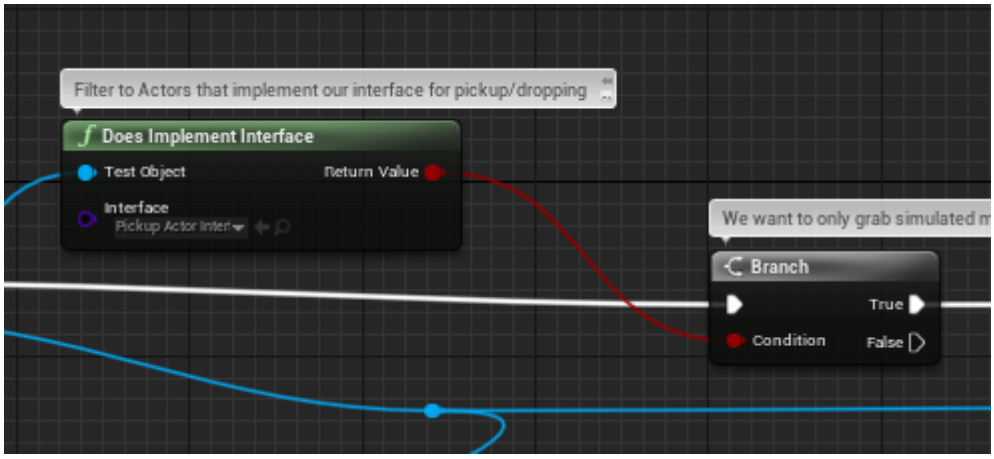


(Figure 17) Pickup Function ,inside PickupActorInterface



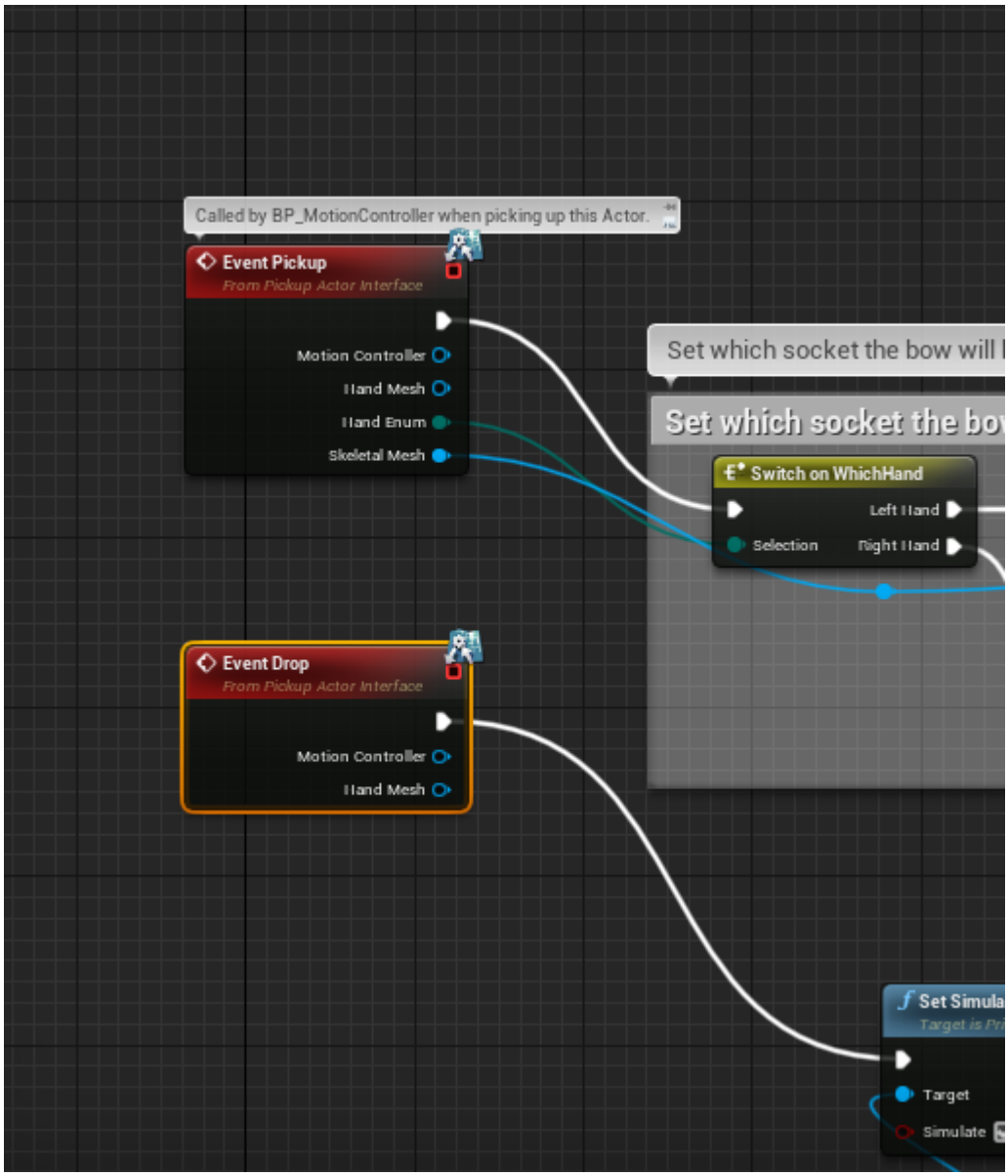
(Figure 18) Drop Function ,inside PickupActorInterface

The Pickup/ Drop functions that were previously stated are interfaces what this means according to the Unreal Engine documents is that "its useful for ensuring that a set of (potentially) unrelated classes implement a common set of functions" so even though the VR Character Blueprints and the blueprints are not related at all we can still link them together.



(Figure 19) GrabActorLeft Function,inside PickupActorInterface

How this works in The Last Arena VR is that when the player finds the object its trying to grab it will only call the Pickup function of the objects that has that interface otherwise it will just ignore that object.

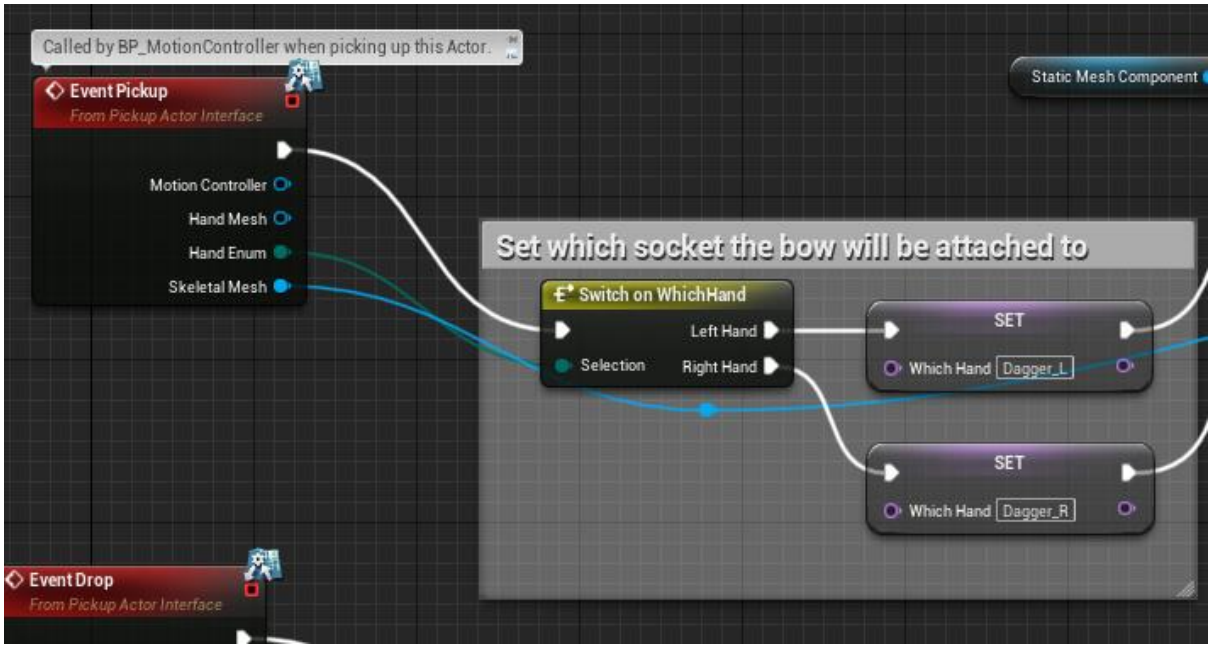


(Figure 20) Where the function from the Event interface is placed, in Bow BP

Then inside of the weapon blueprint using the interface you are free to implement how you want the weapon to be attached to the hand and even though the VR Character blueprint and Weapon blueprint are completely different you with have all the parameters from the VR Characters at your disposal.

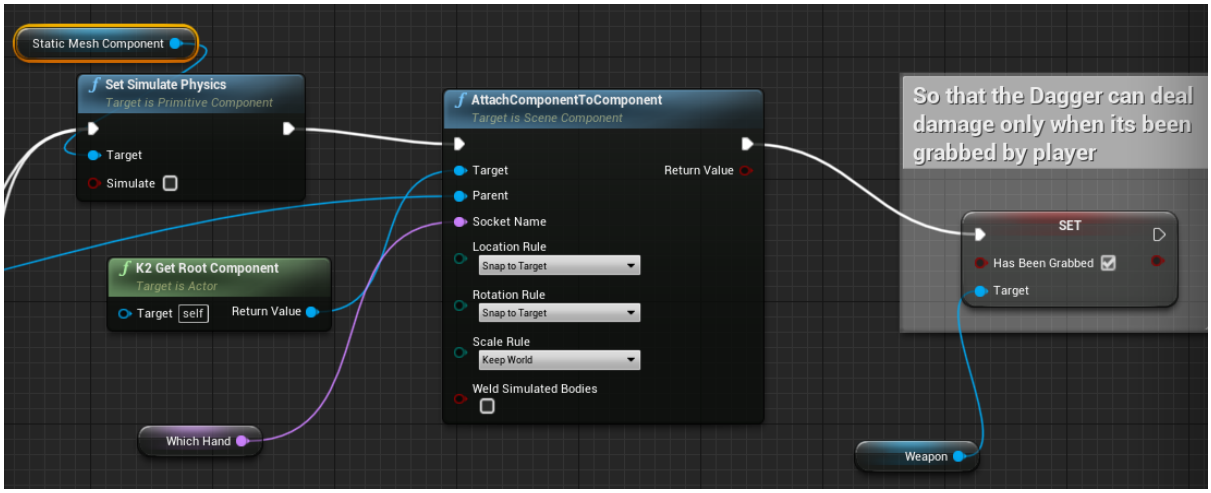
How Picking Up & Dropping Certain Weapons Work

The Dagger



(Figure 21) Grabbing The Weapon, in Dagger BP

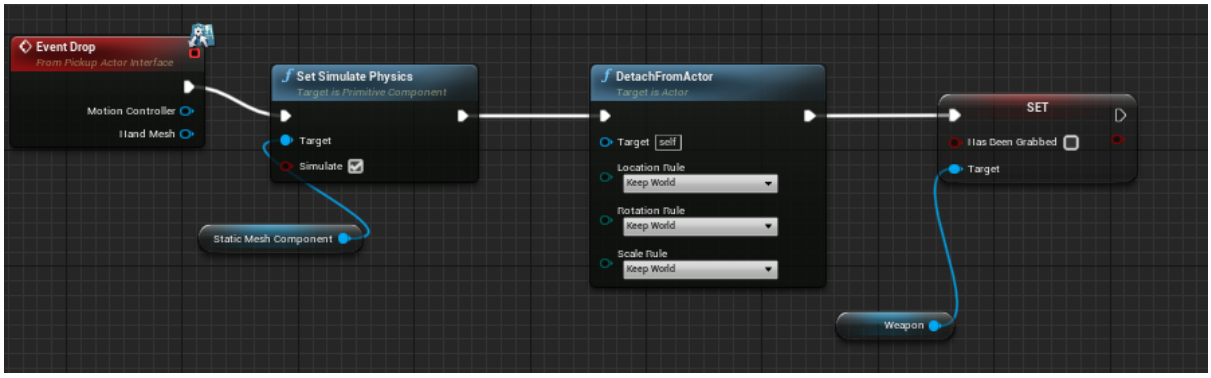
The implementation for picking up and dropping is the most simplest due to the fact that the Dagger only requires one hand, once the Dagger blueprint receives the call from the VR Character the Dagger needs to first determine which hand it will be attached. This can be done thanks to the Hand Enum and depending on the result of this enumerator a variable of type “Name” will be set which will be the socket for the Dagger in either the left or right hand.



(Figure 22) Grabbing The Weapon, in Dagger BP

After that to make sure that the Dagger doesn’t move from the hand due to something physics related like gravity the physics of the Dagger is disabled, the Dagger is then attached to the hand by attaching itself to the socket that is on the skeletal mesh aka the VR Character model. Then a

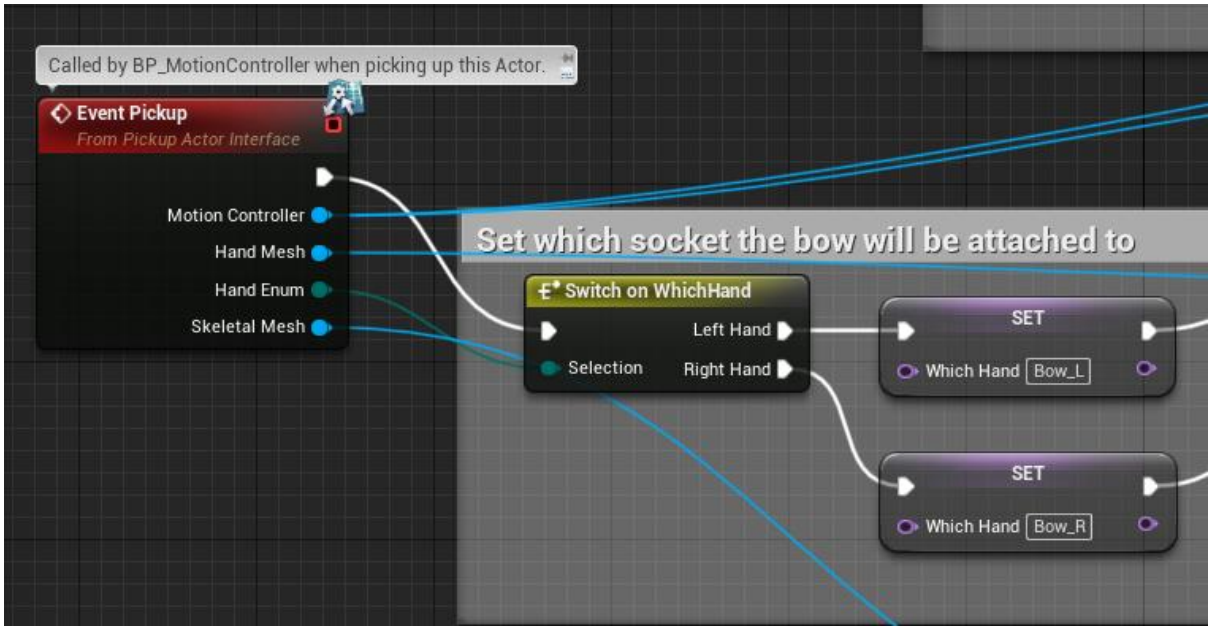
variable called “HasBeenGrabbed” will be set to true to tell the game that the Dagger has been picked up and it also means that the Dagger can deal damage to the Enemy which will be explained later on when we talk more about the Weapons individually.



(Figure 23) Dropping The Weapon, in Dagger BP

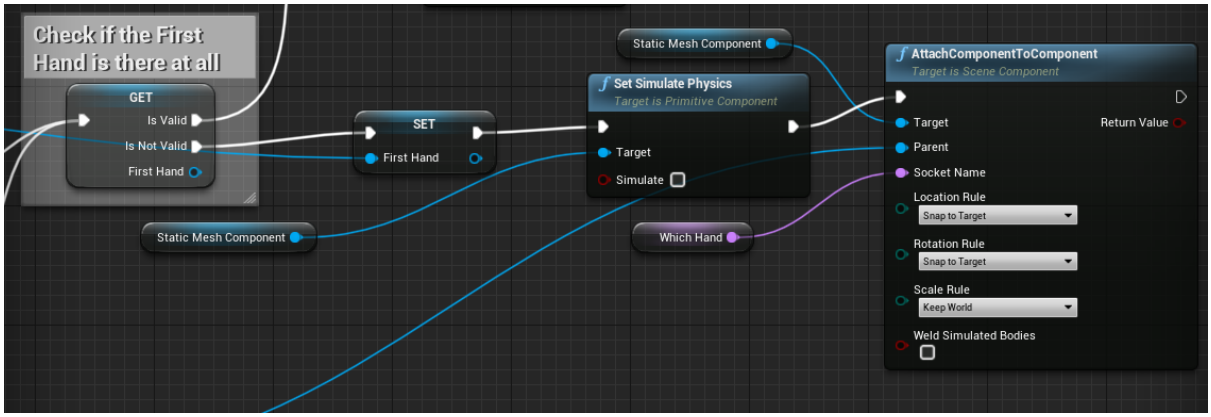
Dropping the weapon is extremely simple, when the Drop function is called by the VR Character Blueprint the physics of the Dagger is enabled once again so that it can drop the ground, the Dagger will detach itself from the Player and the “hasBeenGrabbed” variable will be set to false to tell the game that it is no longer being picked up and it can no deal any damage.

The Bow



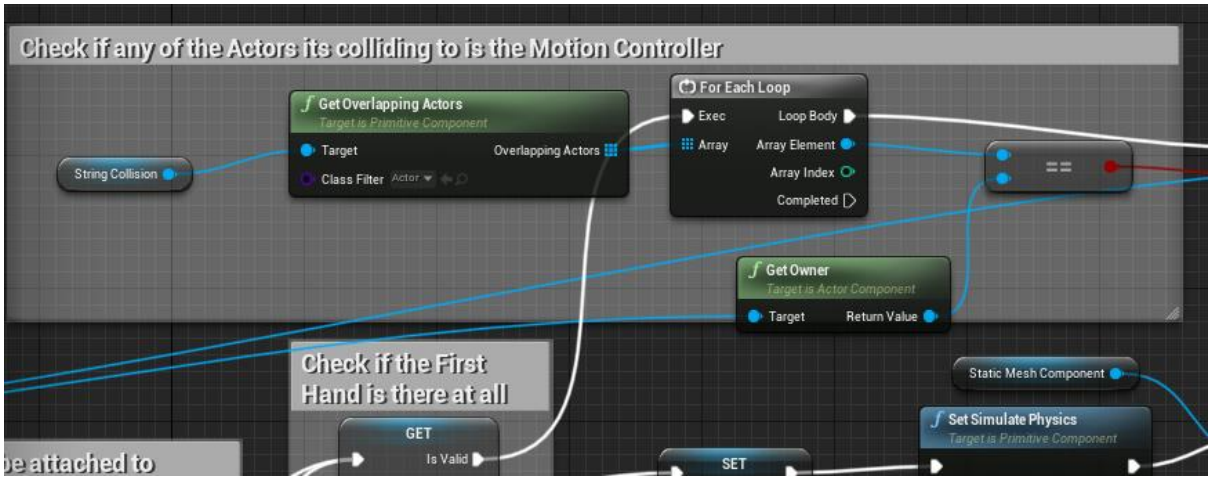
(Figure 24) Grabbing The Weapon, in Bow BP

Since the Bow is two handed the picking up and dropping implementation is more sophisticated than the Dagger, but at first just like the Dagger the Bow will check which hand it will need to be attached to and depending on the hand a variable of type name will be set so that the Bow knows which socket it needs to attach itself to.

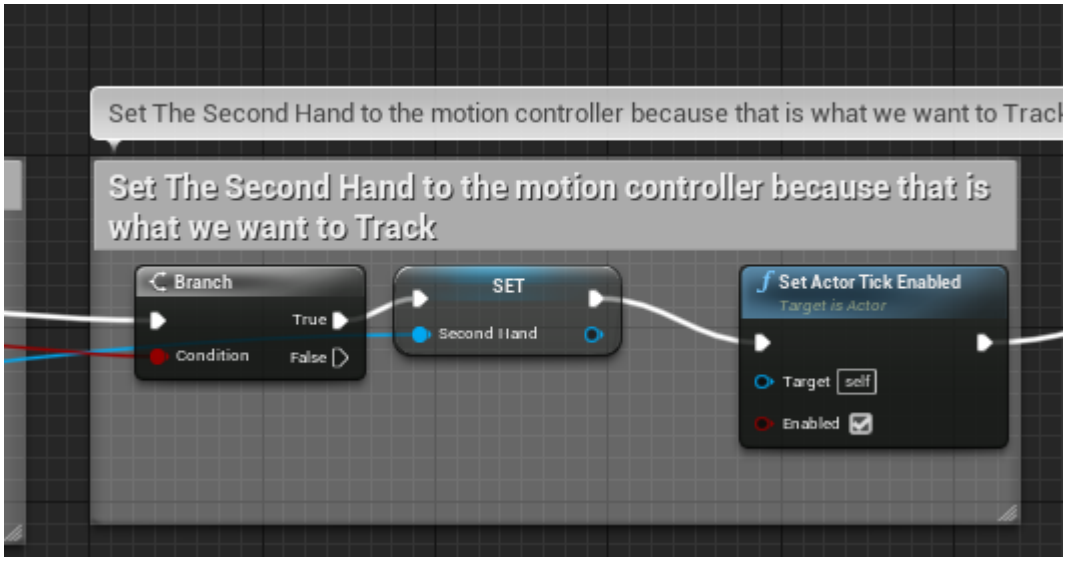


(Figure 25) Grabbing The Weapon, in Bow BP

Here is where the grabbing for the Bow gets different, there is a scene component variable named "First Hand" that will hold reference to the hand (Or the grab spheres in this case) and allows the bow to be attached to the first hand which after means we can check for a second hand. Before we get onto the second hand we must first check if there even is a First hand that is holding onto the Bow, if the "FirstHand" variable is set to null that means the bow is not being held and if that is the case set the "First hand" variable to the hand that is trying to pickup the Bow, after that Physics are disabled and the Bow is attached to the appropriate socket just like what happened with the Dagger.

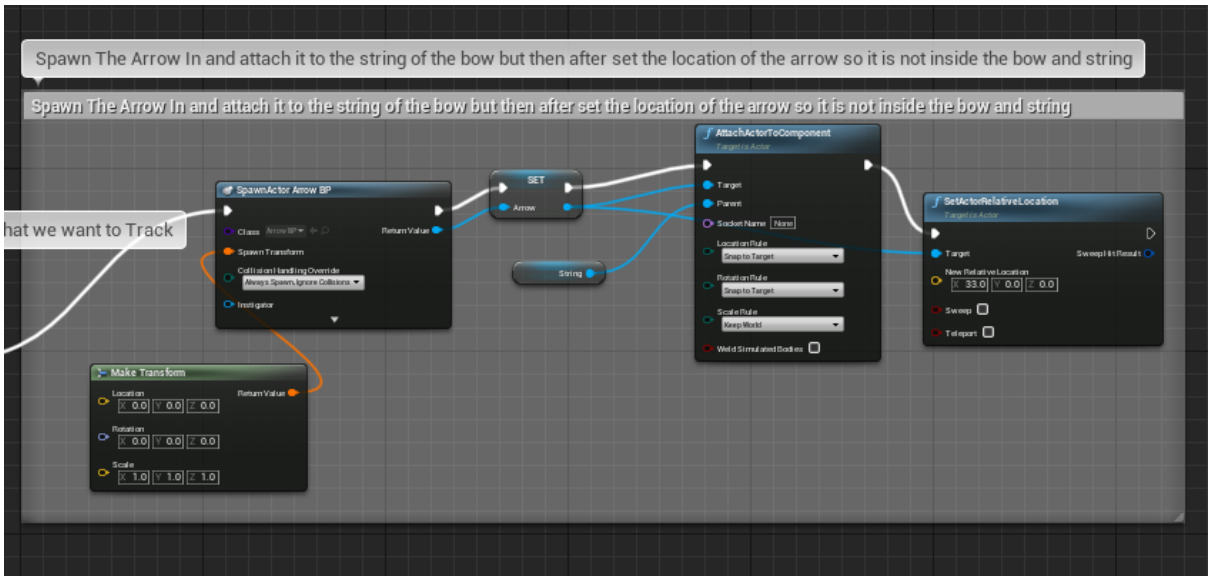


(Figure 26) Grabbing The Weapon, in Bow BP



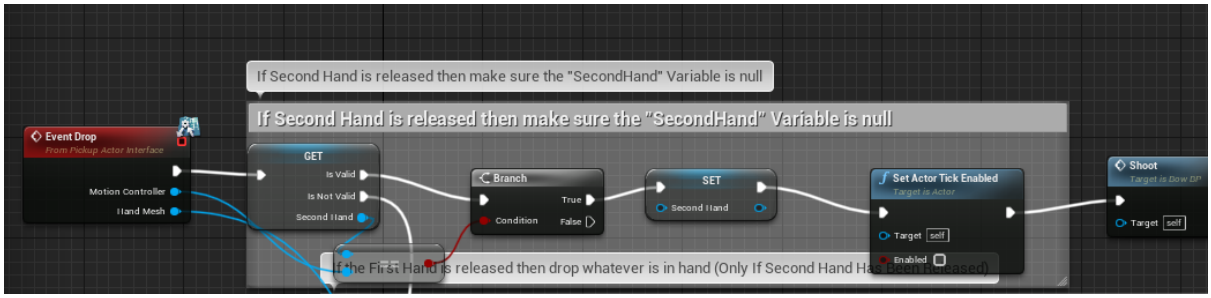
(Figure 27) Grabbing The Weapon, in Bow BP

If "First Hand" variable has already been set then the string collision (that will be explain later on when we talk about the Bow individually) will check for everything that is overlapping it and if it is a motioncontroller set it to the "Second Hand" variable which does the same thing as the "FirstHand" variable. After that enable the event tick of the Bow blueprint which will basically allow the string of the bow to move to the same place as the second hand (This also will be explained later on).



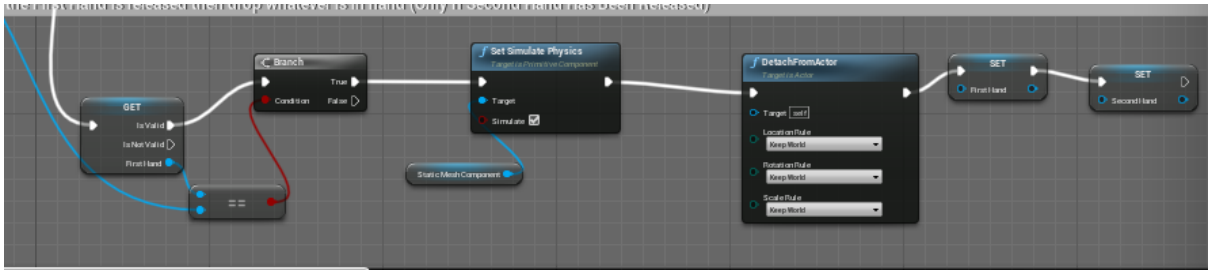
(Figure 28) Spawning In The Arrow, in Bow BP

After the arrow for the Bow will then be spawned in and attached to the string so that it can move along with the string when the player uses their second hand to pull it back.



(Figure 29) Dropping The Weapon, in Bow BP

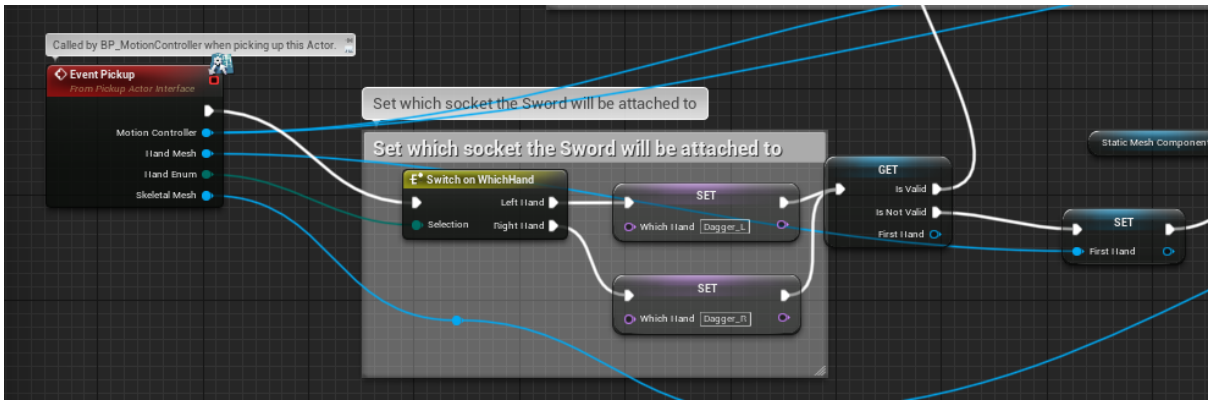
When dropping the bow with your second hand that is pulling on the string the “Second Hand” variable will be set to null which means that the string of the Bow is no longer being pulled, along with that the event tick will be disabled so the string wont attempt to follow the “SecondHand” component and lastly a function called “Shoot” will be called which will be explained later on.



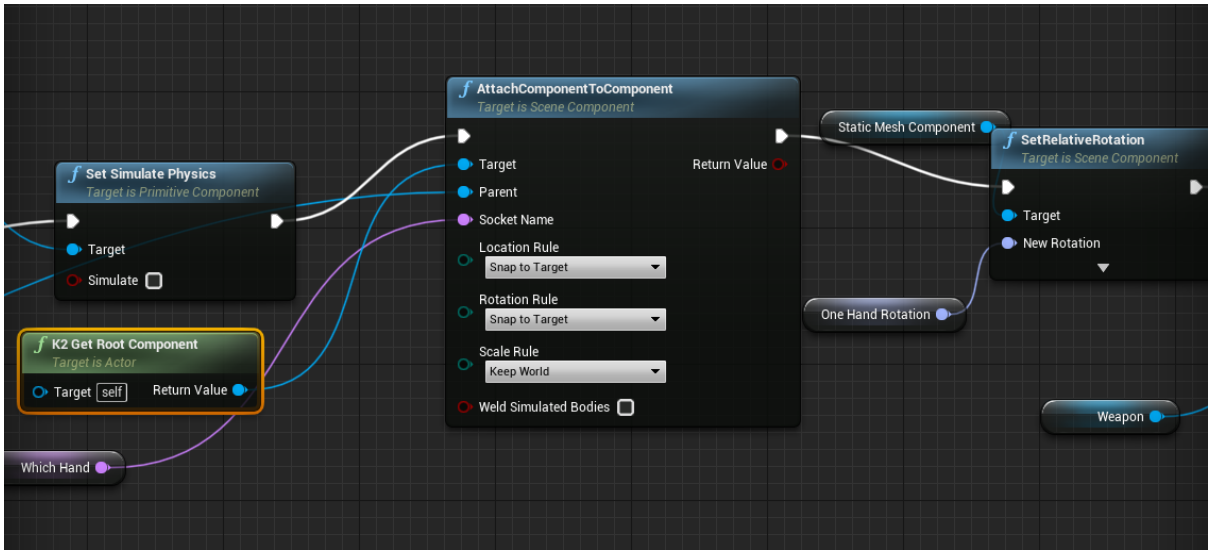
(Figure 30) Dropping The Weapon, in Bow BP

If the “Second Hand” variable is now null but the first hand is still holding the bow then do the same thing that was done to drop the Dagger and then make sure the “FirstHand” variable and “SecondHand” variable are set to null.

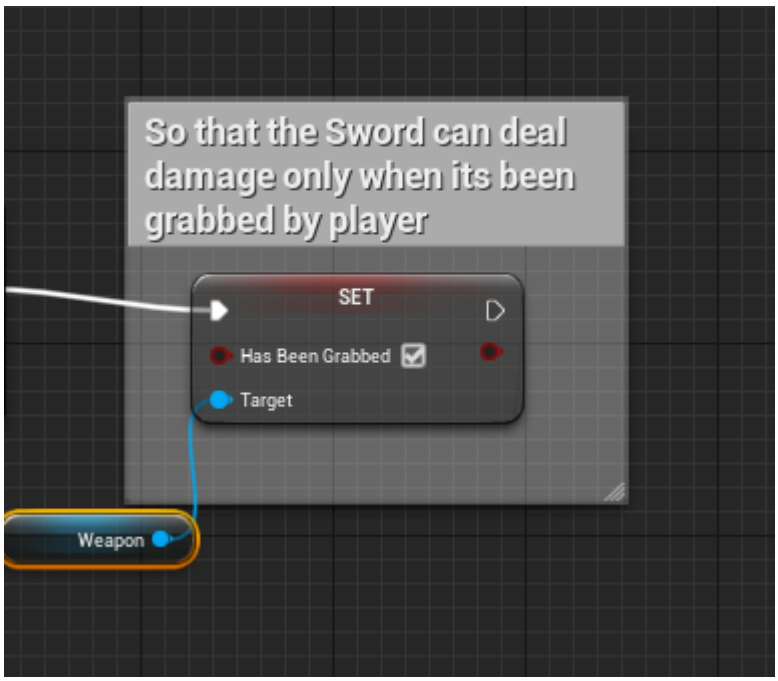
The Sword



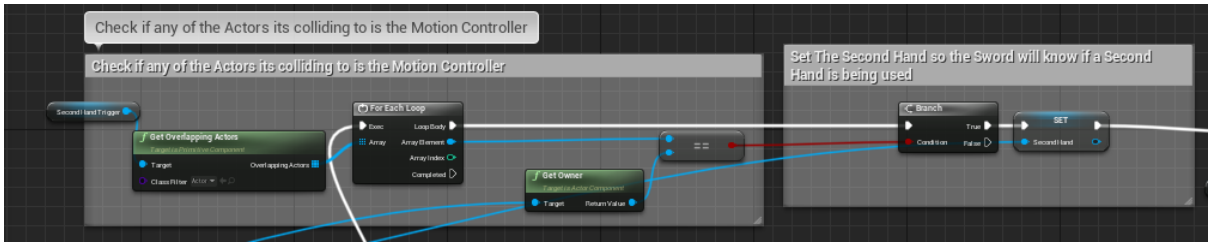
(Figure 31) Grabbing The Weapon, in Sword BP



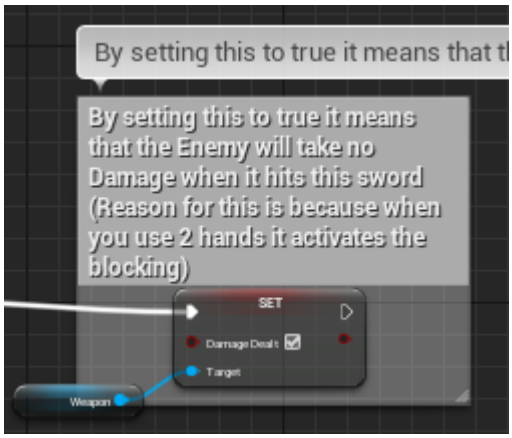
(Figure 32) Grabbing The Weapon, in Sword BP



(Figure 33) Grabbing The Weapon, in Sword BP



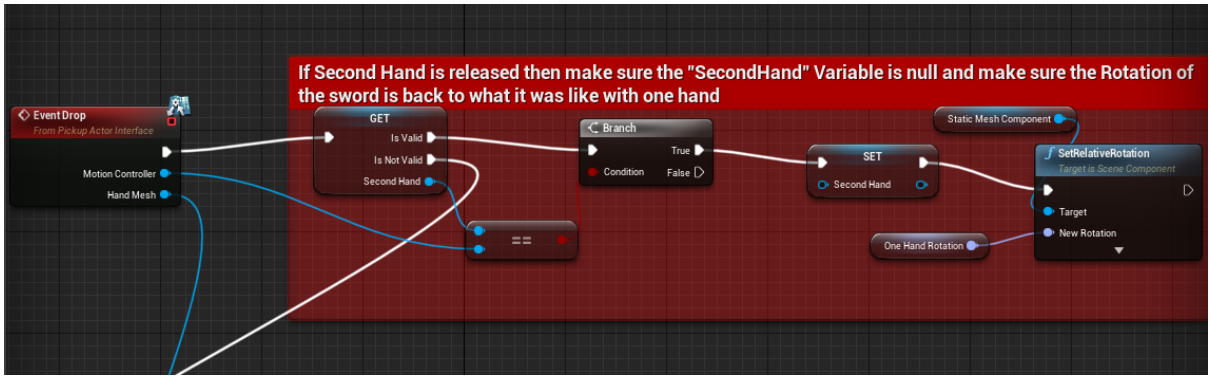
(Figure 34) Grabbing The Weapon, in Sword BP



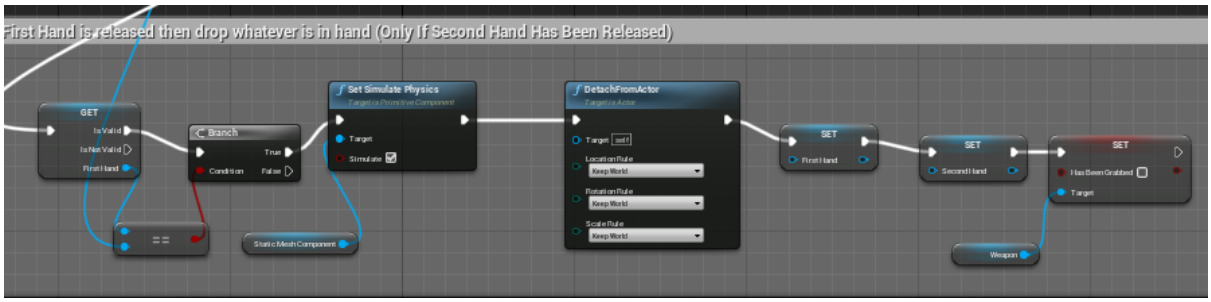
(Figure 35) Grabbing The Weapon, in Sword BP

Just like for the Bow a check for the First hand will be done and if its not valid then it will be set, the only difference really is that the "HasBeenGrabbed" variable is back just like for the Dagger which will determine whether or not the weapon can do damage. The other difference is that the sword uses the same socket as the Dagger but this time it is rotated after its attached to make it easier for the player to hold.

Something that is not shown in the screenshots above is that the Sword will change rotation when the second hand grabs onto it but this will be explained in more detail later.



(Figure 36) Dropping The Weapon, in Sword BP



(Figure 37) Dropping The Weapon, in Sword BP

Just like the Pickup function there isn't really any difference apart from that when you let go of the Sword with the second hand the rotation of the sword goes back to normal and the "HasBeenGrabbed" variable changing back to false when you let go of the weapon completely.

VR Combat & Weapons

This section explains the crucial parts of how the weapons work and will show thow each of the weapons work (excluding the Pickup features since they were discussed in the previous section) and will explain the C++ scripts that helped make the VR Combat.

Note: You will see the OnComponentBeginOverlap function be used twice since C++ can't really communicate with Blueprints that well so to combat that issue the OnComponentBeginOverlap function was also used in the Blueprints of the weapon.

Health Component C++ Script

```cpp
#include "HealthComponent.h"
#include "Engine.h"
// Sets default values for this component's properties
UHealthComponent::UHealthComponent()
{
    // Set this component to be initialized when the game starts, and to be ticked every frame.  You can turn these features
    // off to improve performance if you don't need them.
    PrimaryComponentTick.bCanEverTick = false;

    //Set The Max Health to 100
    maxHealth = 100;

    //Current Health will start off with MaxHealth
    currentHealth = maxHealth;
    // ...
}


// Called when the game starts
void UHealthComponent::BeginPlay()
{
    Super::BeginPlay();

}

void UHealthComponent::TakeDamage(float Damage)
{
    if (Damage <= 0)
    {
        return;
    }

    //Makes it so that the max value for Current Health is whatever is in maxHealth and makes sure the minimum value is 0  and it reduces the currentHealth variable
    currentHealth = FMath::Clamp(currentHealth - Damage, 0.0f, maxHealth);
    GEngine->AddOnScreenDebugMessage(-1, 1, FColor::Red, FString::Printf(TEXT("Health is %f"), currentHealth));

}
```

(Figure 38) Health Component Script

The Player and the Enemies in the Last Arena VR will have a component called "HealthComponent" which is originally a C++ script, the purpose of this script is so that the Player and Ais all have health and can all take damage. The max health is set in the constructor so as soon as the Component is created the max health will be set to 100 and the current health will be the same as the max Health.

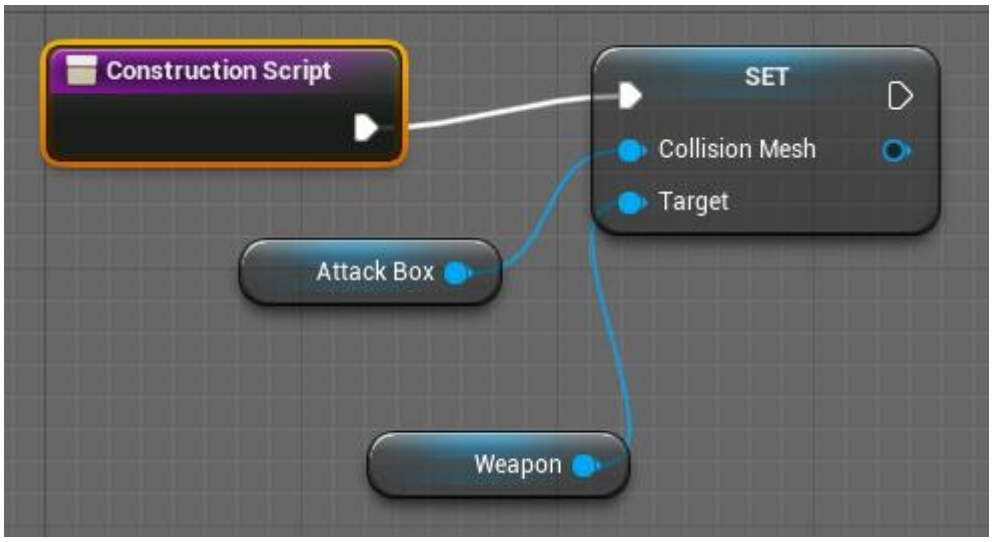
As for the Take Damage() function it will run as long as the Damage float value being passed into it is more than 0, if the damage value is more than 0 then the game will get the currentHealth value of the object this component is on and it will subtract the health by the Damage float variable.

Weapon C++ Script

```cpp
// Called when the game starts
void UWeapon::BeginPlay()
{
    Super::BeginPlay();

    damageDealt = false;
    hasBeenGrabbed = false;
    //Override OnComponentBeginOverlap
    CollisionMesh->OnComponentBeginOverlap.AddDynamic(this, &UWeapon::onOverlapBegin);



}

//TESTING PURPOSES (Overrides On Component Begin Overlap)
void UWeapon::onOverlapBegin(UPrimitiveComponent* OverlappedComponent, AActor* OtherActor, UPrimitiveComponent* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult)
{

    //FOutputDeviceNull ar;
    //We put the Function we want to call inside the TEXT bracket
    //const FString command = FString::Printf(TEXT("Test"));
    //If the player it has been collided with has been found

    //If the Weapon collides with the HitBox and the weapon hasnt dealt damage yet also make sure the weapon has been Grabbed
    if (OtherComp->ComponentHasTag("Enemy") && !damageDealt && hasBeenGrabbed)
    {
        //Find the Health Component and then call the Take Damage function (Inputs the Damage variable)
        OtherActor->FindComponentByClass<UHealthComponent>()->TakeDamage(Damage);
        GEngine->AddOnScreenDebugMessage(-1, 1, FColor::Blue, FString::Printf(TEXT("ENEMY FOUND")));
        damageDealt = true;
    }
    else
    {
        GEngine->AddOnScreenDebugMessage(-1, 1, FColor::Red, FString::Printf(TEXT("ENEMY NOT FOUND")));
    }

}
```

(Figure 39) Weapon Script

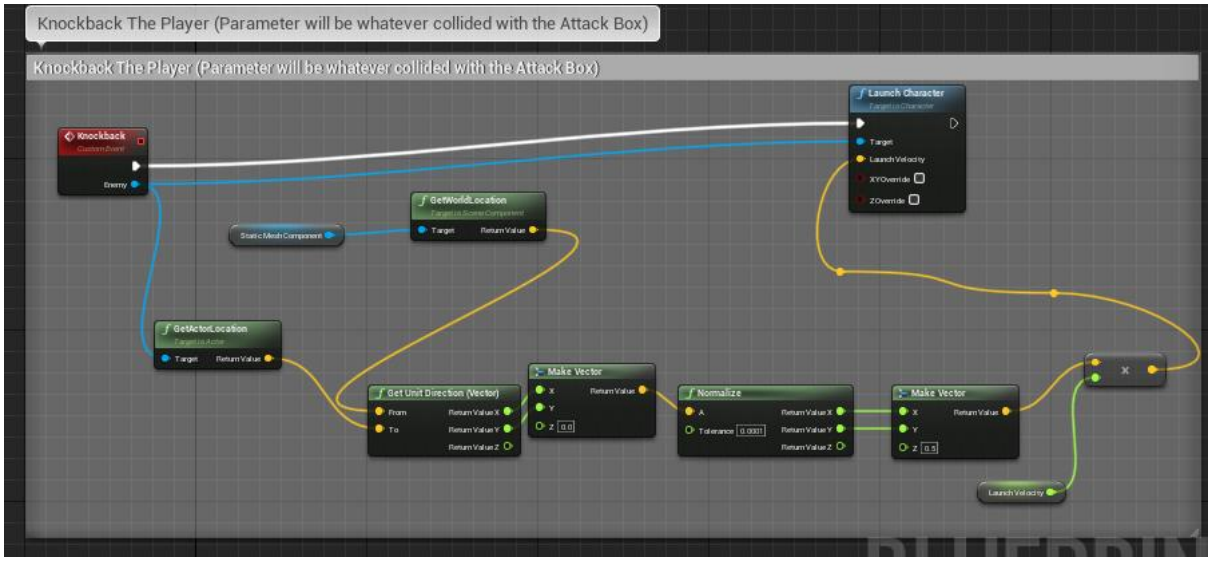
Each weapon in the Last Arena VR will have a component called “Weapon” which is originally a C++ Script, the purpose of this script is so that when the collision mesh of the weapon overlaps with the hitbox of the enemy (which is a component that will have a tag called “Enemy”) it will get the actor.



(Figure 40) Conscruction Script function, in Dagger BP

Since the script doesn’t know of the existence of the attack box in the ConstructionScript function the collision mesh for the Weapon script is set there for all the weapons.

Knockback Function (That All Weapons Have)



(Figure 41) Knockback Function, in Dagger BP

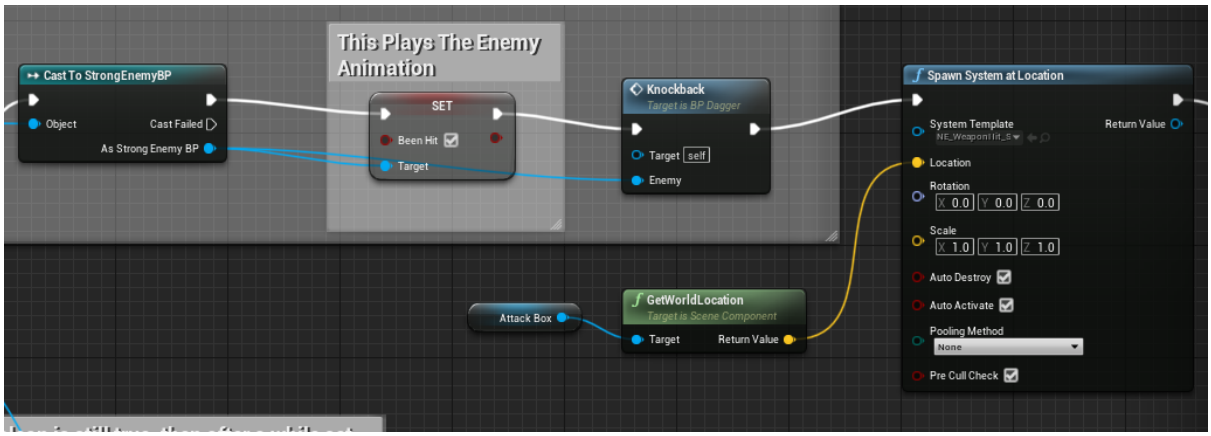
To get the direction at which the Enemy should be launched at the game will first get the direction vector from the location of the weapon that is attacking the enemy to the location of the enemy that is getting attacked but then the game will only get the X and Y axis of that vector. Once it gets the Vector values it will normalize it and set the Z axis to 0.5 and then multiplies it with whatever value is in the “LaunchVelocity” variable.

This value is then sent to the LaunchCharacter function that is provided by Unreal Engine, the function will know what Enemy to launch because of the parameter of the Knockback event where the “Enemy” variable is given in the OnComponentOverlap Event.

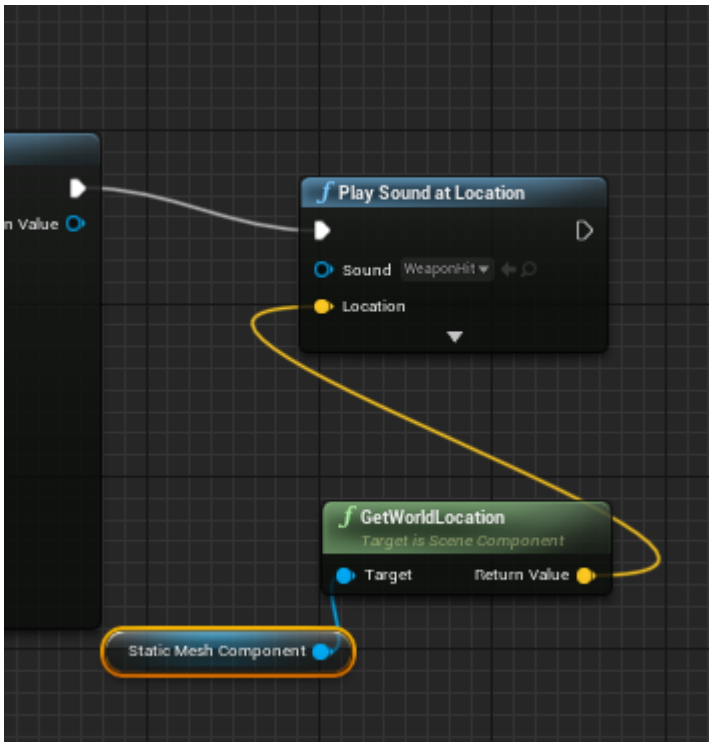
OnComponentBeginOverlap for Weapons

This section will explain the OnComponentBeginOverlap for the weapons since they are all pretty much the same but an example for it on the Dagger Blueprint will be used.

(Figure 42) The OnComponentBeginOverlap function, in Dagger BP



(Figure 43) The OnComponentBeginOverlap function, in Dagger BP



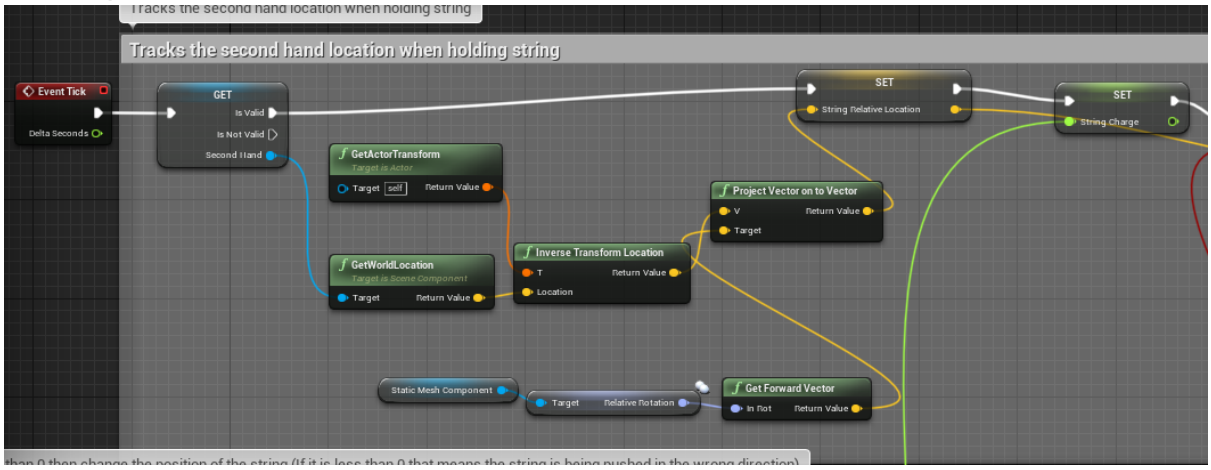
(Figure 44) The OnComponentBeginOverlap function, in Dagger BP

Just like in the C++ script a check is done to see whether the Component that the Attackbox is colliding with has a tag called “Enemy” and whether or not it has dealt damage and been picked up. If everything goes through correctly then the DaggerBP will cast to the Enemy that it is colliding with and would then set the variable named “BeenHit” to true which just plays the animation of the Enemy being hit. Once that is done the knockback function that was previously explained will be called to knock the Enemy back and then a particle system will spawn to indicate the Enemy got hit and a sound will also play.

(Figure 45) The OnComponentEndOverlap function, in Dagger BP

The only way to deal damage again is to make sure that the Dagger is no longer overlapping with the Enemy and a delay for 0.4 seconds will happen which will then set the "DamageDealt" variable to false which means damage can be dealt again.

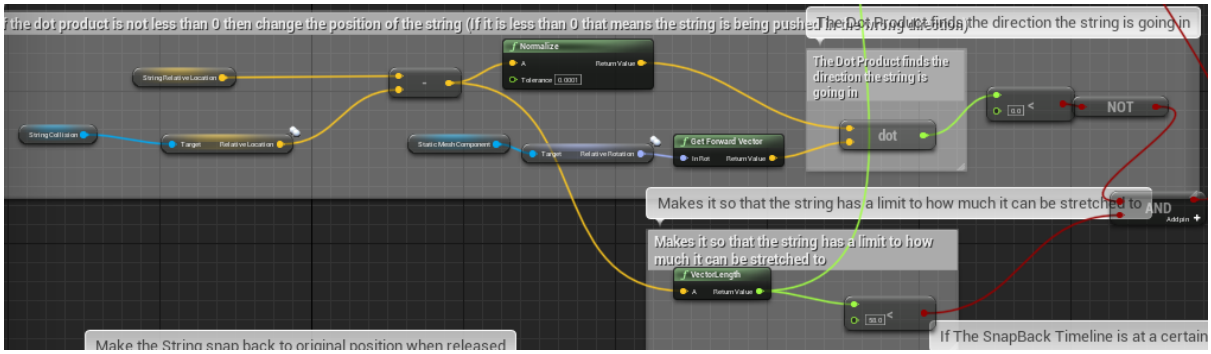
Shooting With Bow & Arrow



(Figure 46) Event Tick, in Bow BP

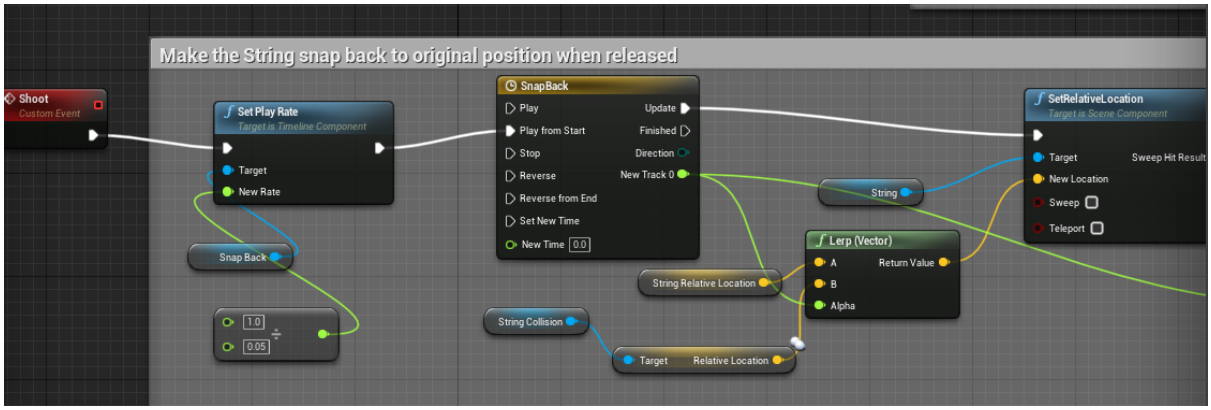
As explained in the picking up & dropping the bow section the arrow spawns in once the player releases the second hand, in the Arrow BP there isn't really anything different from the other weapons its main purpose is to just deal damage and delete itself. The interesting part comes with pulling the string of the bow back so just before the Arrow is spawned in the when the event tick is enabled.

If the Second hand is valid then calculations will be done to allow us to move the string of the Bow backwards and forwards with our second hand.



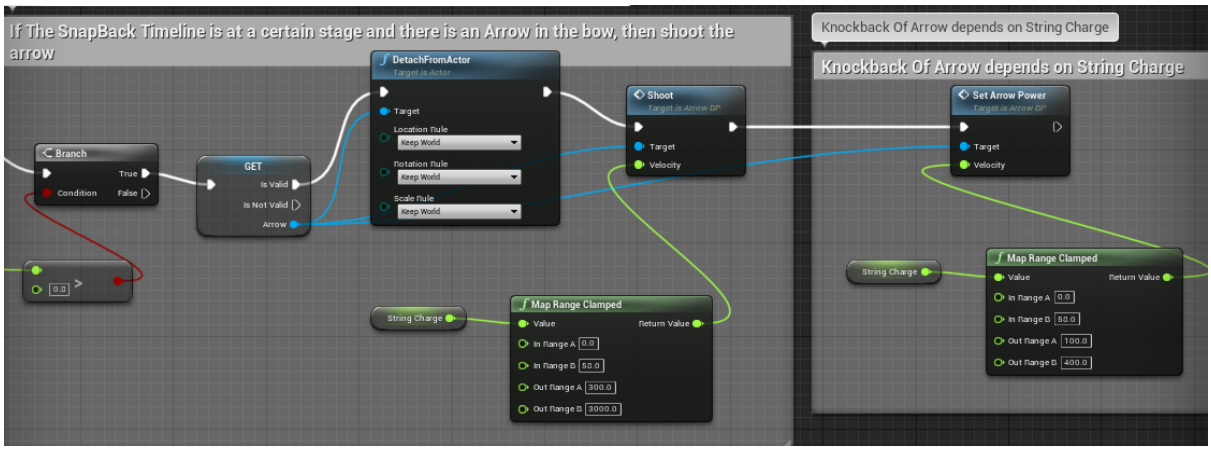
(Figure 47) Calculations for pulling back the string, in Bow BP

To make sure that the string can only go backwards there are some calculations that will check the dot product of the string relative location and the string collision relative collision, if this dot product value is more than 0 and the vector length is less than 50 then move the string, if it isn't don't move the string.



(Figure 48) Shoot Function, in Bow BP

When the second hand is released and the shoot function of the Bow blueprint is called then a timeline called "SnapBack" will be called which will quickly and smoothly put the string back to its original position.



(Figure 49) Shoot Function, in Bow BP

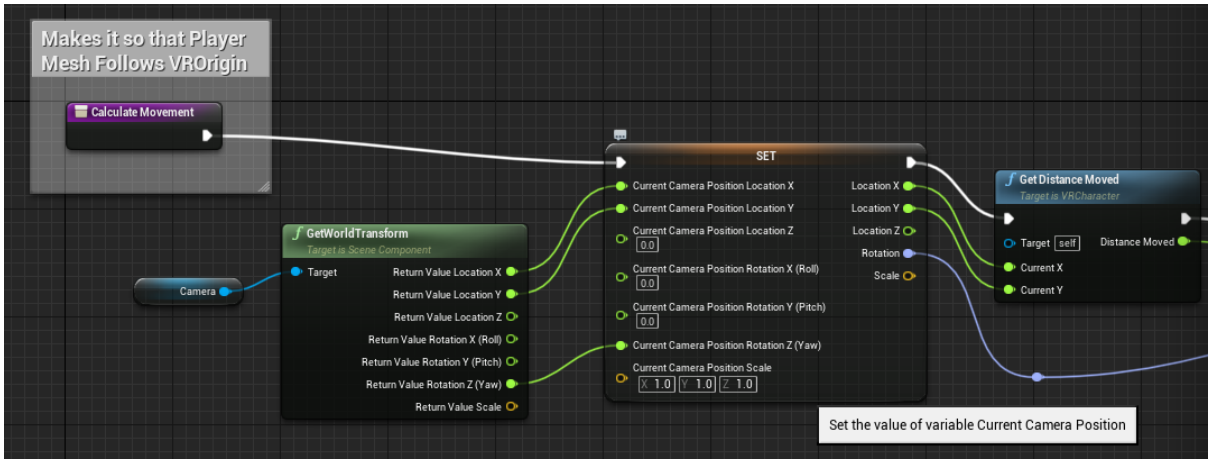
It will also at a certain point of the timeline call the shoot function of the Arrow blueprint which means the arrow will be launched forward and finally how far the arrow will be shot will be set in the ArrowPower function of the Arrow BP where the “stringCharge” variable will be clamped and passed into the function so that the power that the arrow is launched at is not too much.

(Figure 50) Holding The Bow and Arrow and pulling string, in The Last Arena VR

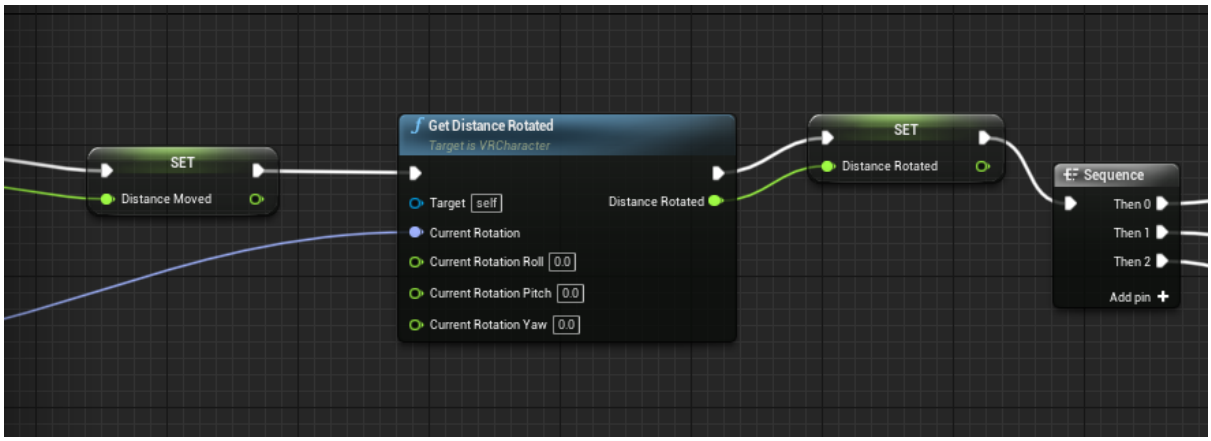
Full body Immersion

This section will show what was done in order to get the character to follow the player in real life, so for example if the player crouches in real life thanks to the Foot Ik and the character model being moving down by the Z axis because its following the Camera (The VR Headset in this case) then the character model will also crouch down.

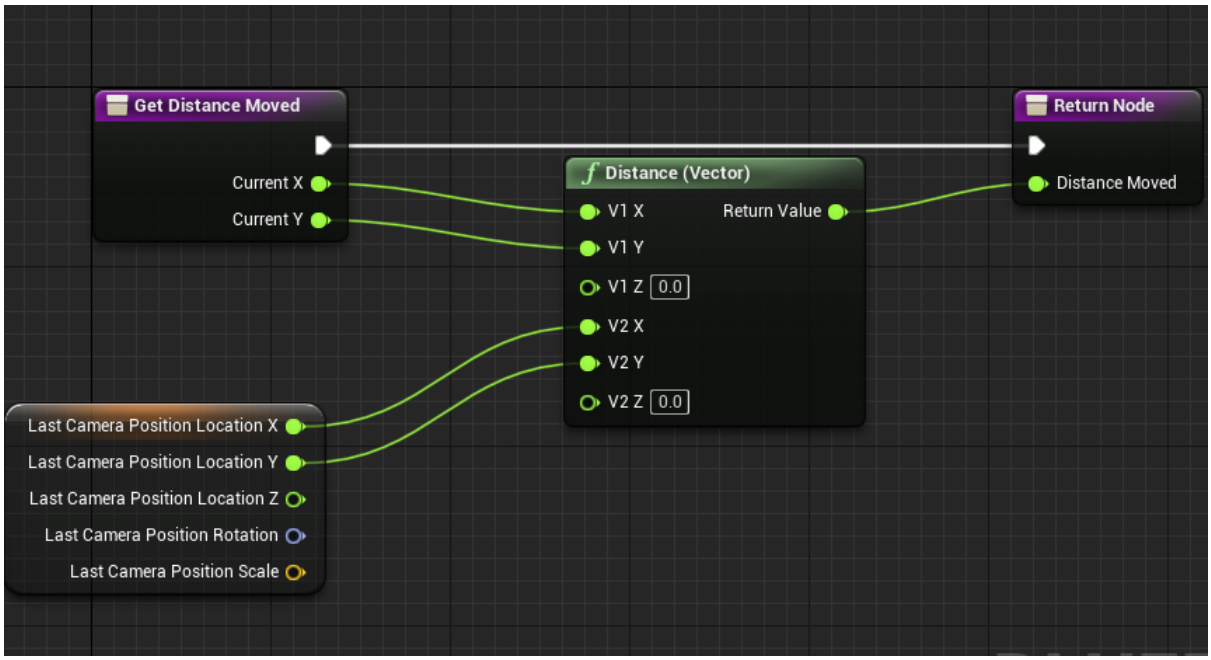
Making Character Body Follow Camera



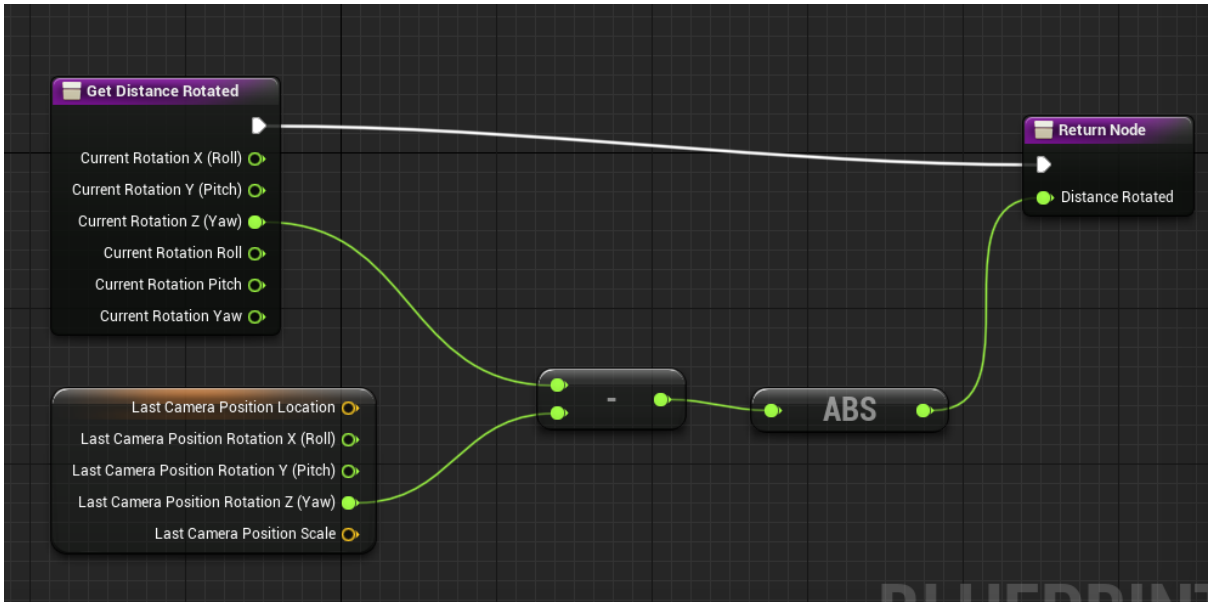
(Figure 51) Calculate Movement Function, in VRCharacterBP



(Figure 52) Calculate Movement Function, in VRCharacterBP

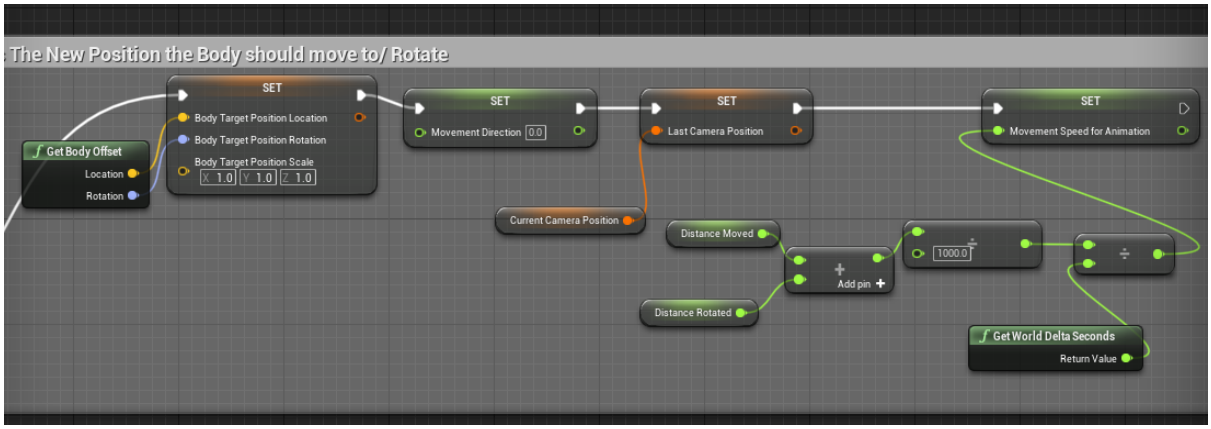


(Figure 53) GetDistancedMove Function, in VRCharacterBP

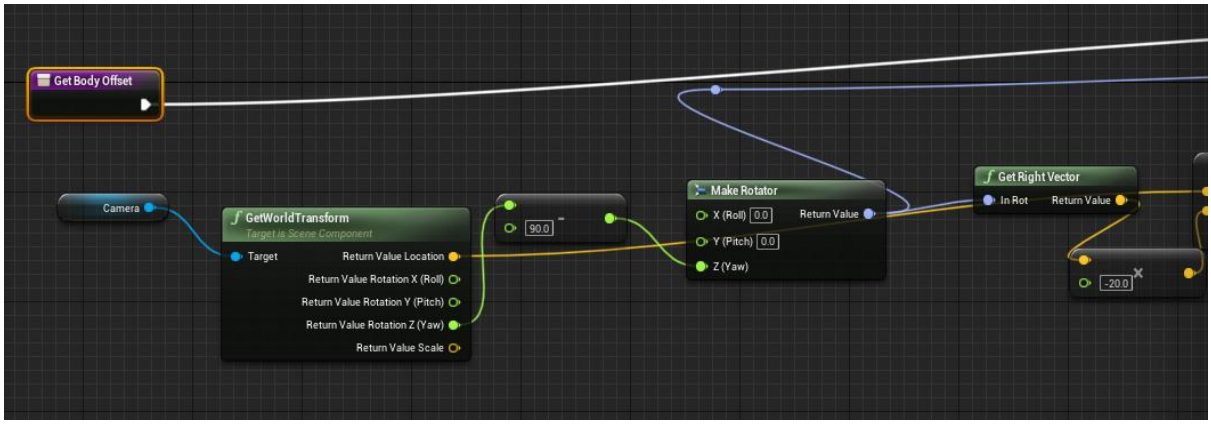


(Figure 54) GetDistancedRotated Function, in VRCharacterBP

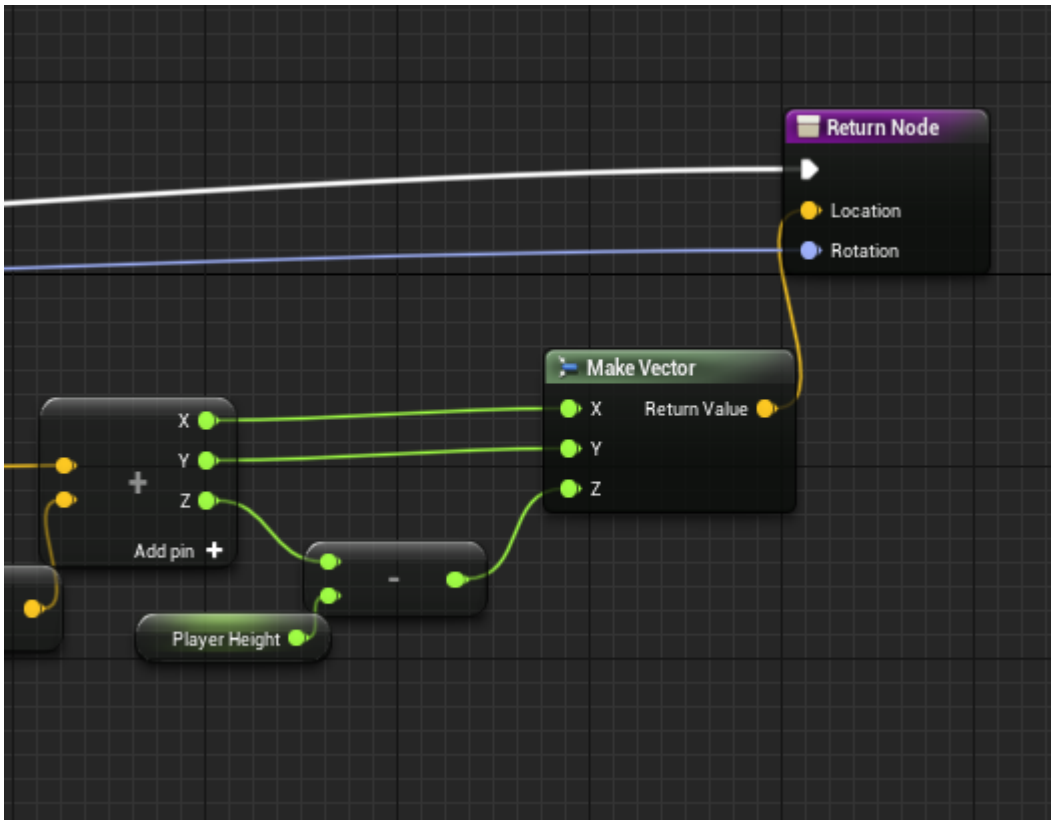
The way the full body immersion works is that the VR Character mesh Z position will always be linked to the location of the VROrigin which for the player is the location of the VR headset. To make sure that the body follows the VROrigin and its camera a function within the VR Character Blueprint was created, in this function other functions such as the GetDistanceMoved() and GetDistanceRotated() are used to get the distance between the last camera position/ rotation and its current position/ rotation(A local variable within the CalculateMovement() function).



(Figure 55) CalculateMovement Function, in VRCharacterBP

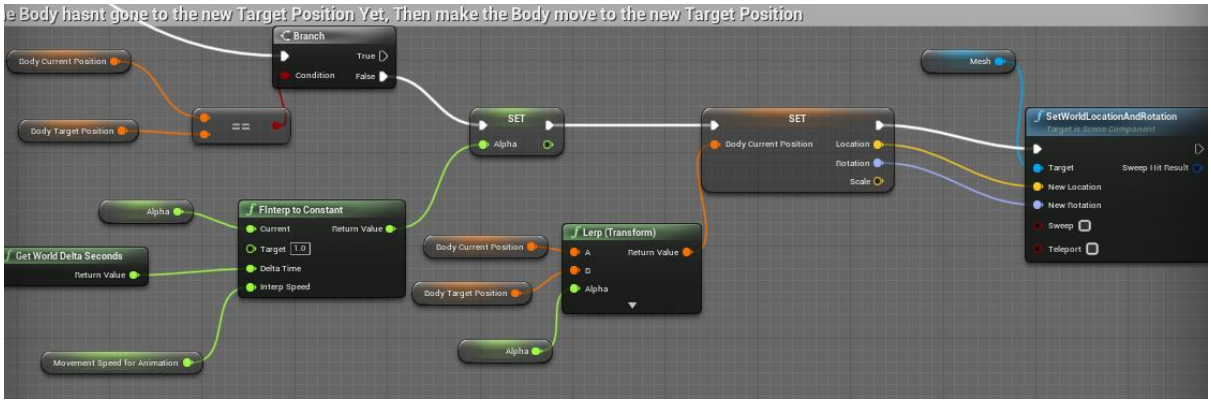


(Figure 56) GetBodyOffset Function, in VRCharacterBP



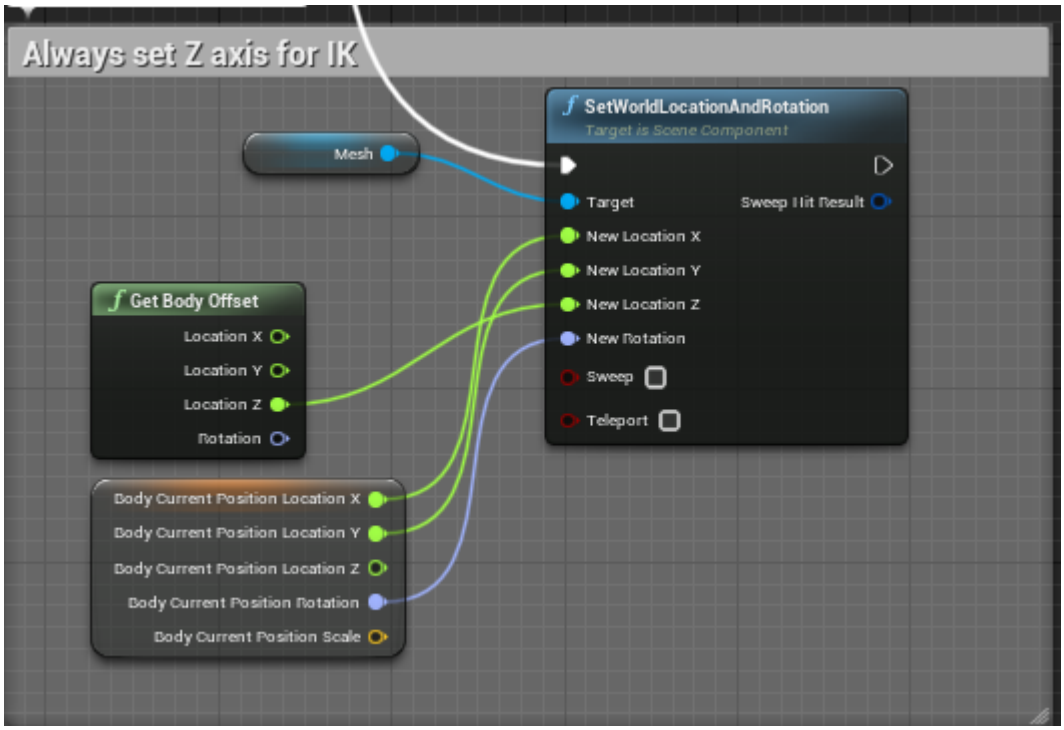
(Figure 57) GetBodyOffset Function, in VRCharacterBP

After those calculations the target position will be set thanks to the BodyOffset() function which will make sure the character mesh will always be set at the right place and the value of the last camera position will be the current position. Finally the movement speed will be set depending on the Distance moved and the distance rotated and this will be used for the animation blueprint.



(Figure 58) CalculateMovement Function, in VRCharacterBP

If the character body is not in its target position then this will set it so that it will lerp towards the position and not teleport around chasing the Camera.

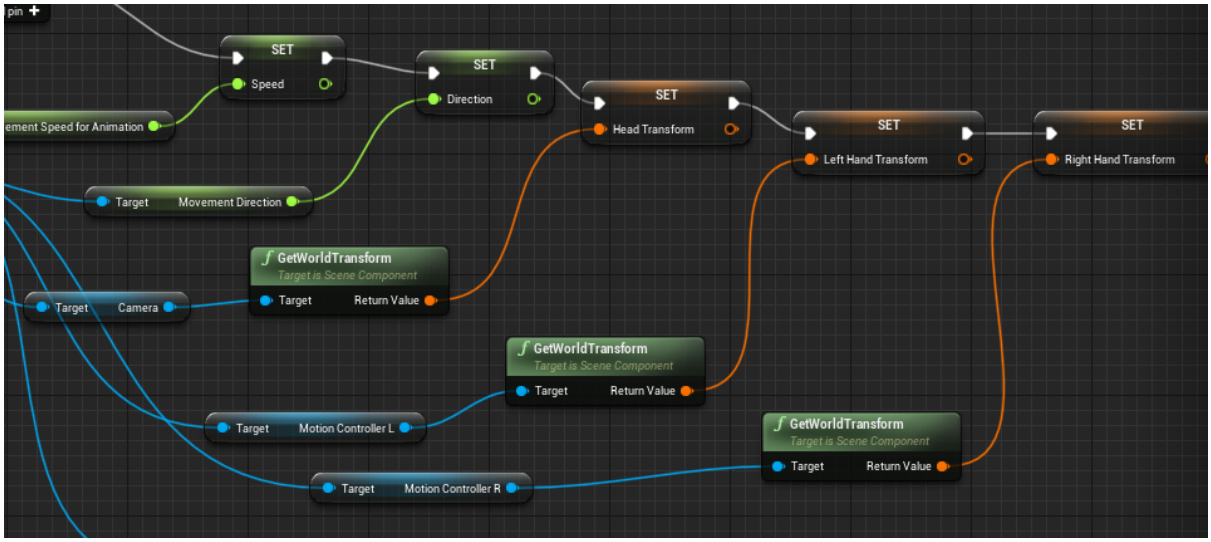


(Figure 59) CalculateMovement Function, in VRCharacterBP

Finally this will change the height of the Character model and is the main thing that allows the player to crouch in game and stand up.

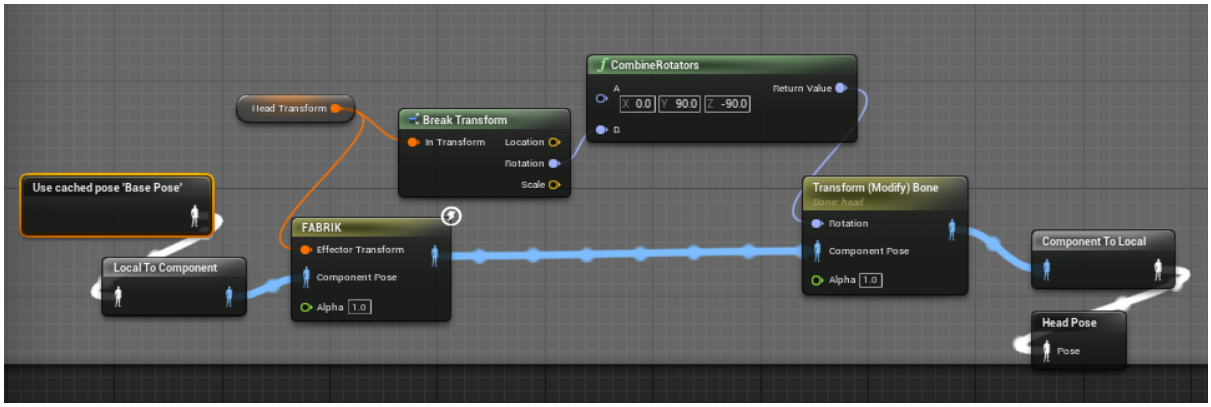
Full Body IK

Head & Hands

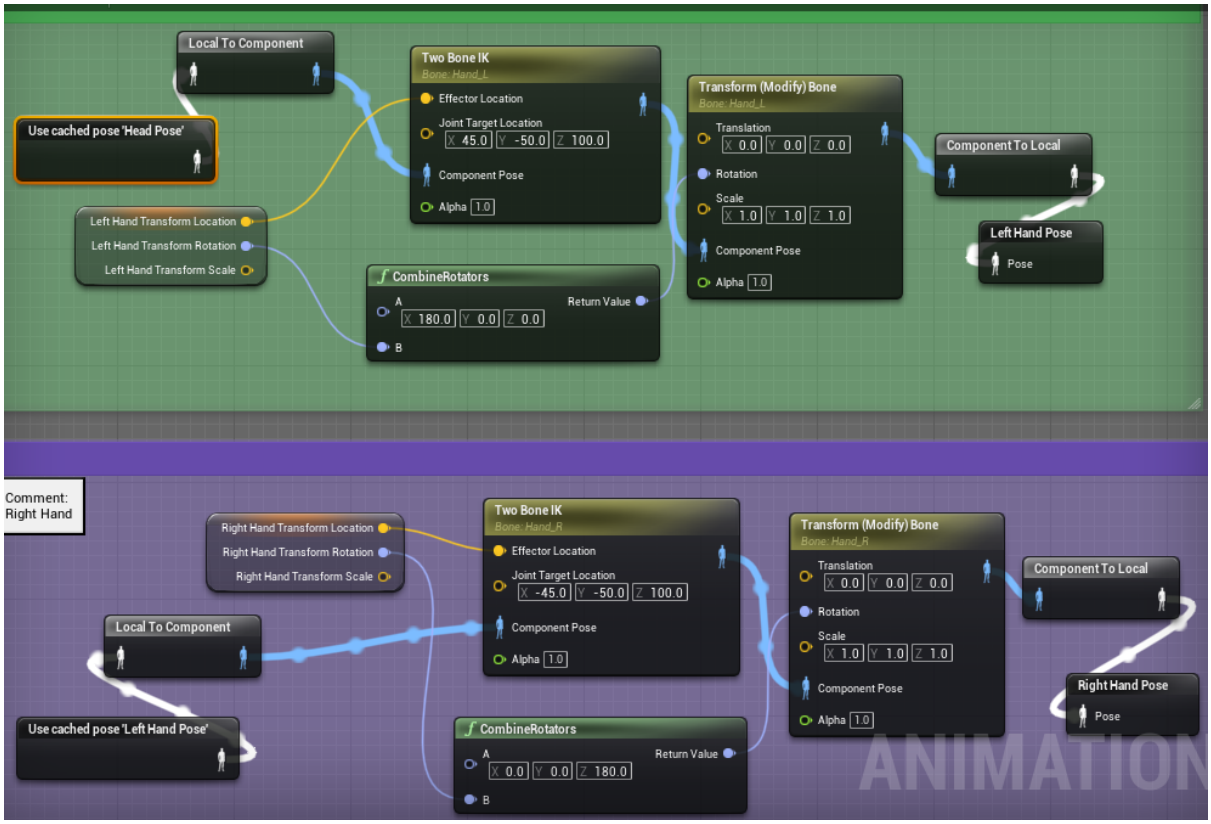


(Figure 60) EventBlueprintUpdateAnimation Function, in ABP_VRCharacter

For the IK of the body we are going to need get reference to the body parts that we want the bones of the mesh to follow, for The Last Arena VR the head bone will move according to the camera (The VR Headset) and the left and right hand bone will move according to the motion controllers (The VR controllers).



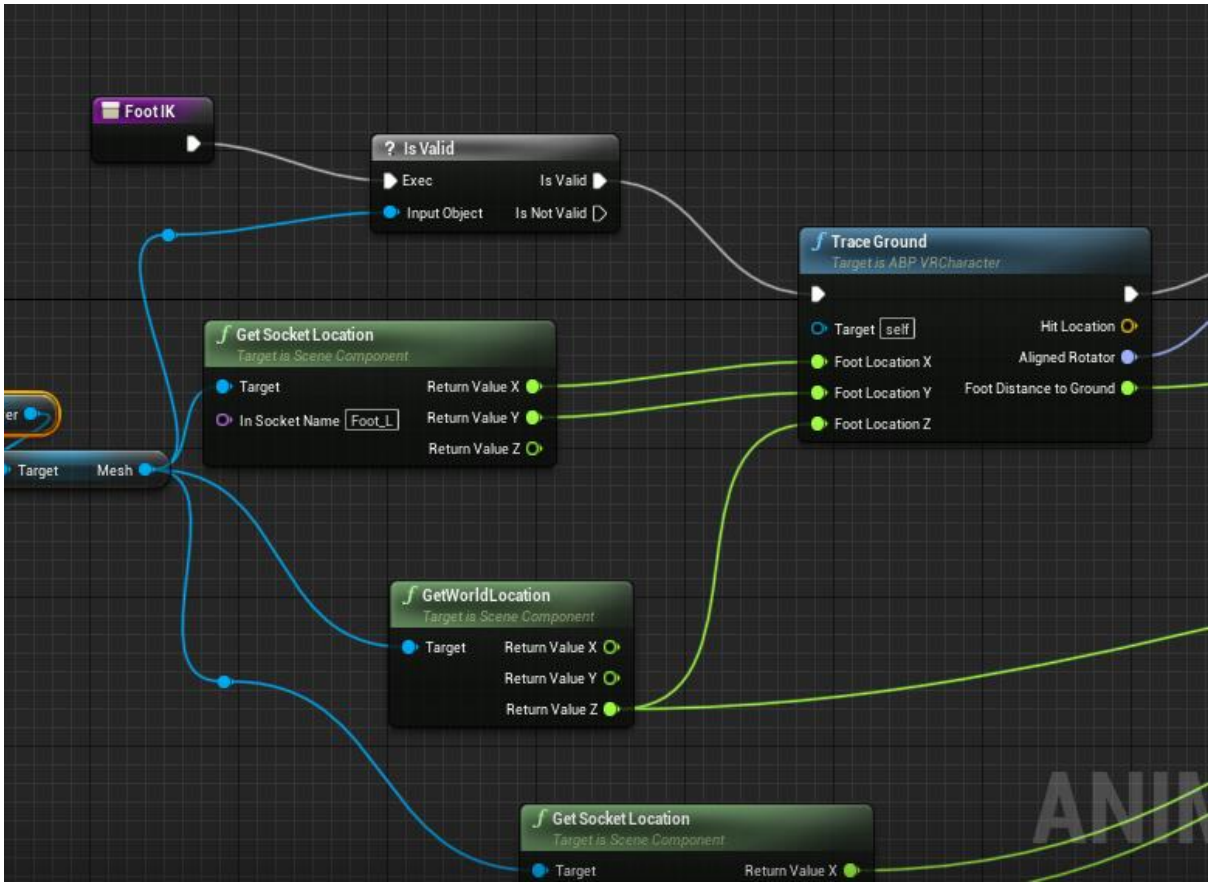
(Figure 61) IK Handling , in ABP_VRCharacter



(Figure 62) IK Handling , in ABP_VRCharacter

Inside the AnimGraph of the VR Character animation blueprint is where all the magic happens with the bones of the head and the hand being modified to follow the players movement with hands and head in real life.

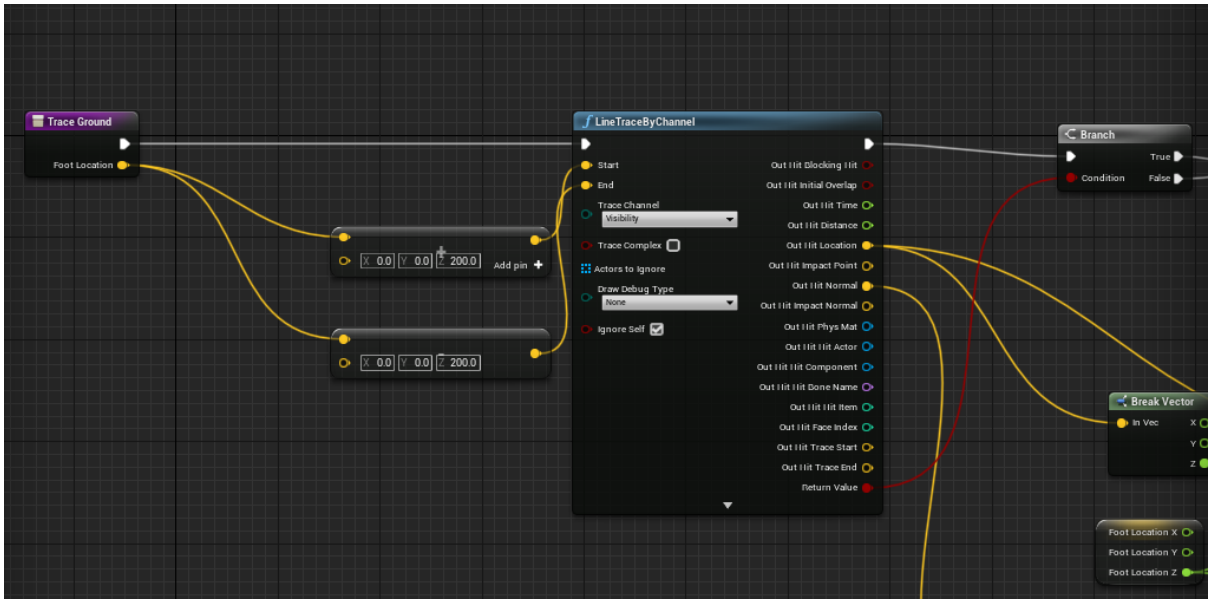
Foot/ Leg



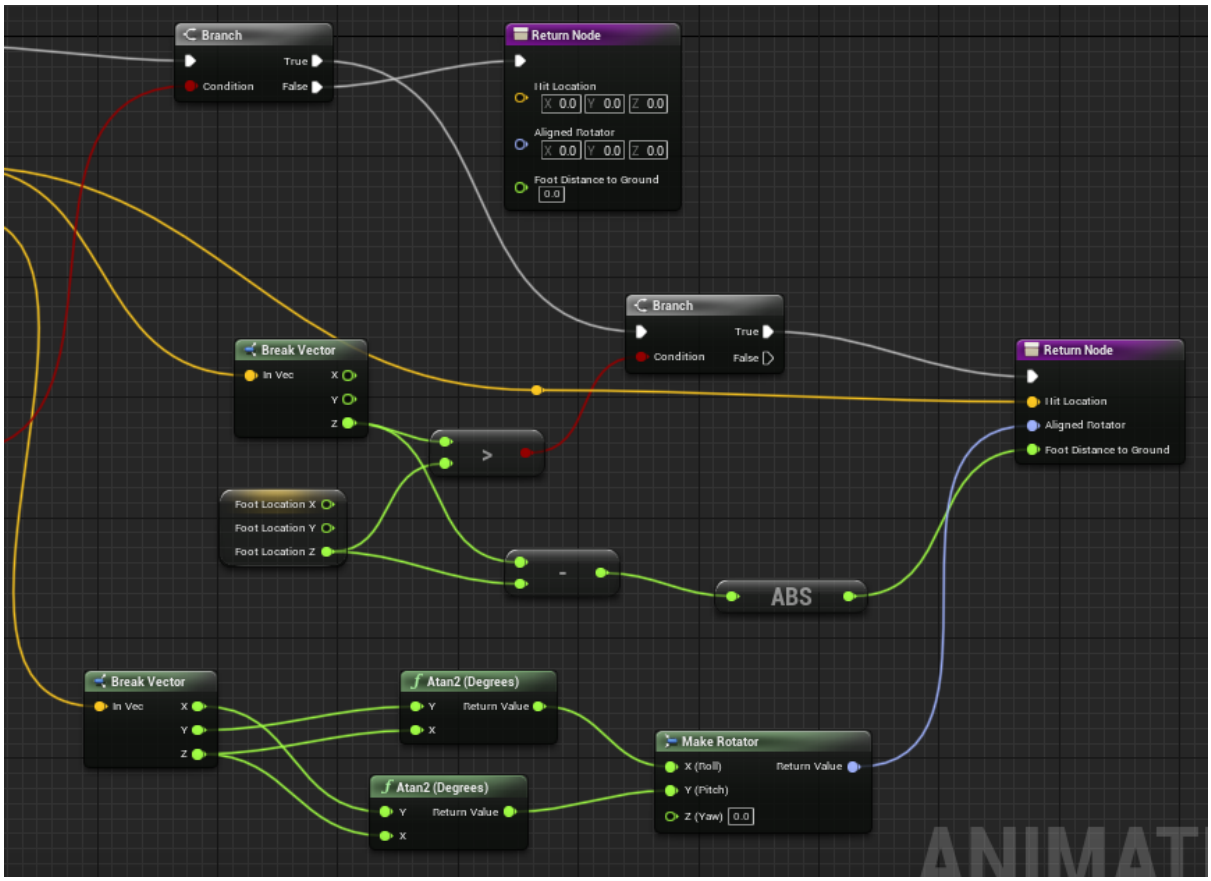
(Figure 63) FootIK Function, in ABP_VRCharacter

In the VR Character Animation Blueprint is where the Leg IK is also stored, thanks the foot ik the character mesh will not sink through the ground even if its Z position is going down because of the Vr camera instead the foot of the character mesh will stay where it is and the legs will bend just like in real life.

This is done in the FootIK() function where it will first check if there is a mesh from the VR Character blueprint, if there is then a function called TraceGround() will be called which will first get the X and Y location of the left foot bone.

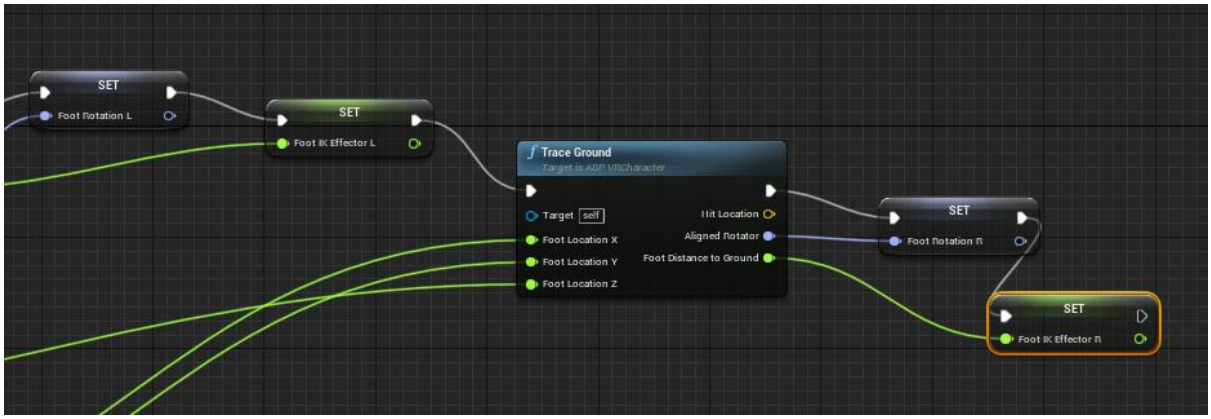


(Figure 64) TraceGround function, in ABP_VRCharacter



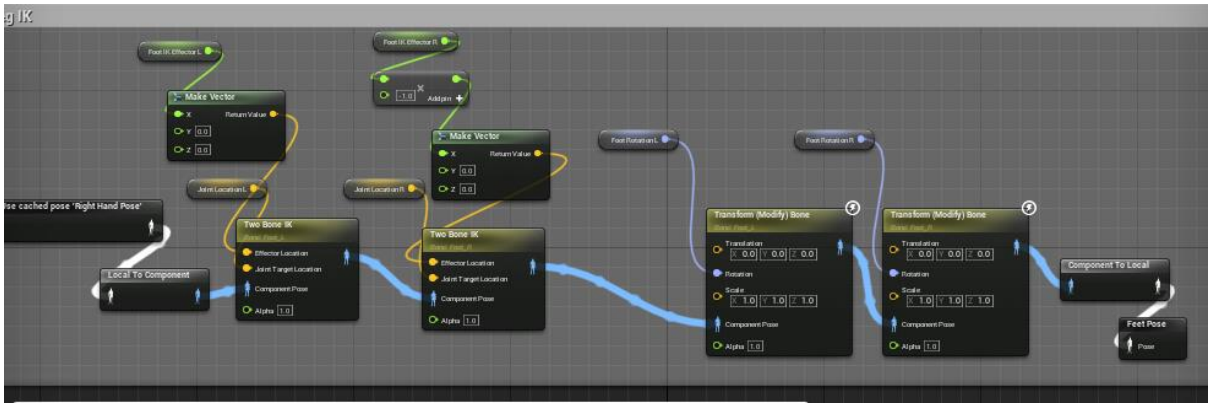
(Figure 65) TraceGround function, in ABP_VRCharacter

The TraceGround function() will use a line trace to get the location of the foot when it is touching the ground, once it gets that value calculations are done to get the distance of the foot from the ground.



(Figure 66) FootIK Function, in ABP_VRCharacter

After that is done the value that was returned from the TraceGround() function called "AlignedRotator" will be set into a variable called "FootRotationL" and the "FootDistancetoGround" will be set into a variable called "FootIkEffector" which will be used in the AnimGraph to get the Foot IK working, the process is then repeated for the right foot as well.



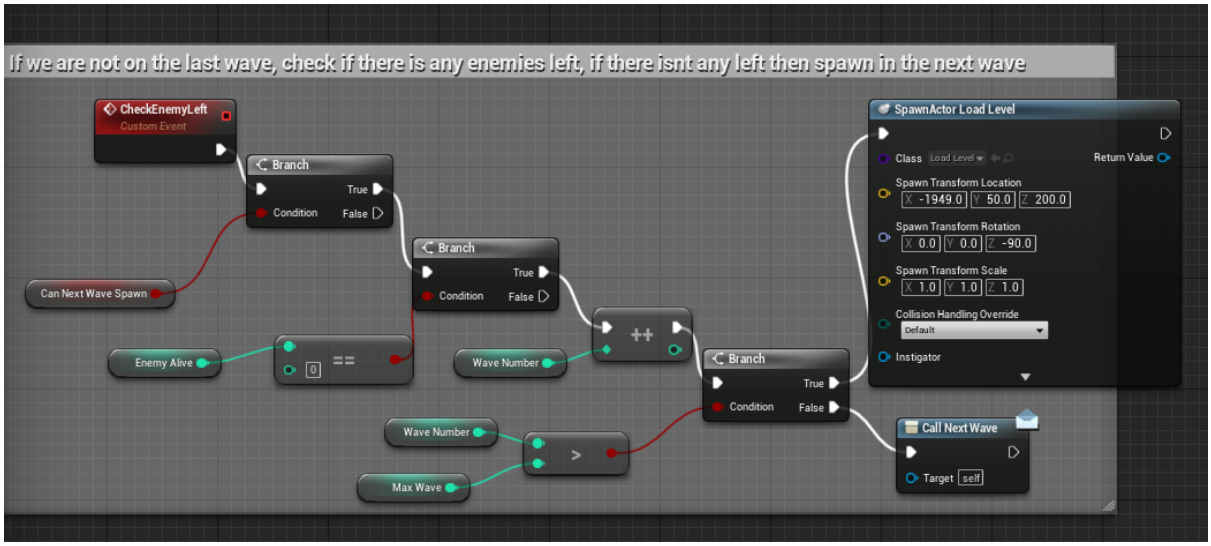
(Figure 67) Leg/Foot IK Function, in ABP_VRCharacter

Similar to the Head and hands the modifier for the bones is also done in the anim graph but instead of following something like a motion controller or camera it is set by the values we got from the FootIk() function.

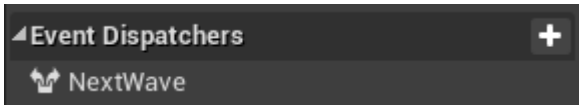
(Figure 68) Full body Immersion finished, in The Last Arena VR

Waves Of Enemies

This section will explain how the Wave system for The Last Arena VR works and how and when the Enemies will be spawned in.



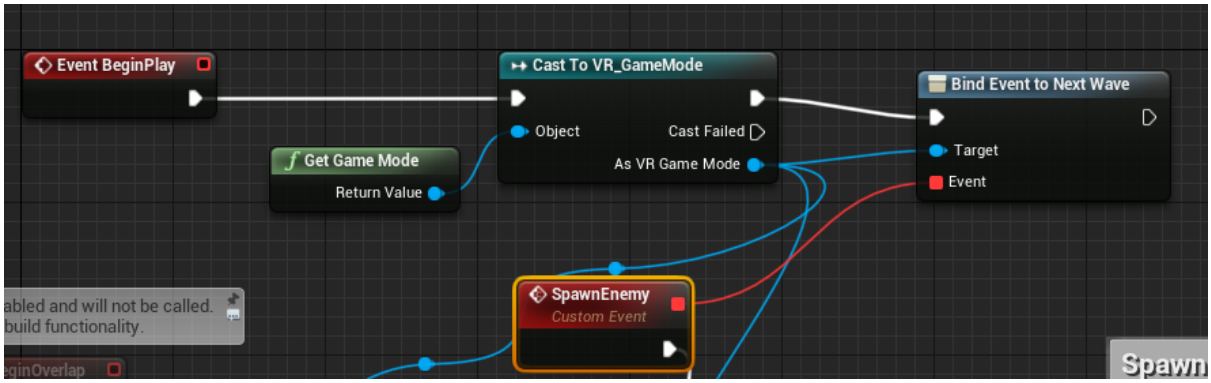
(Figure 69) CheckEnemyLeft Function, in VR_Gamemode



(Figure 70) Event Dispatcher, in VR_Gamemode

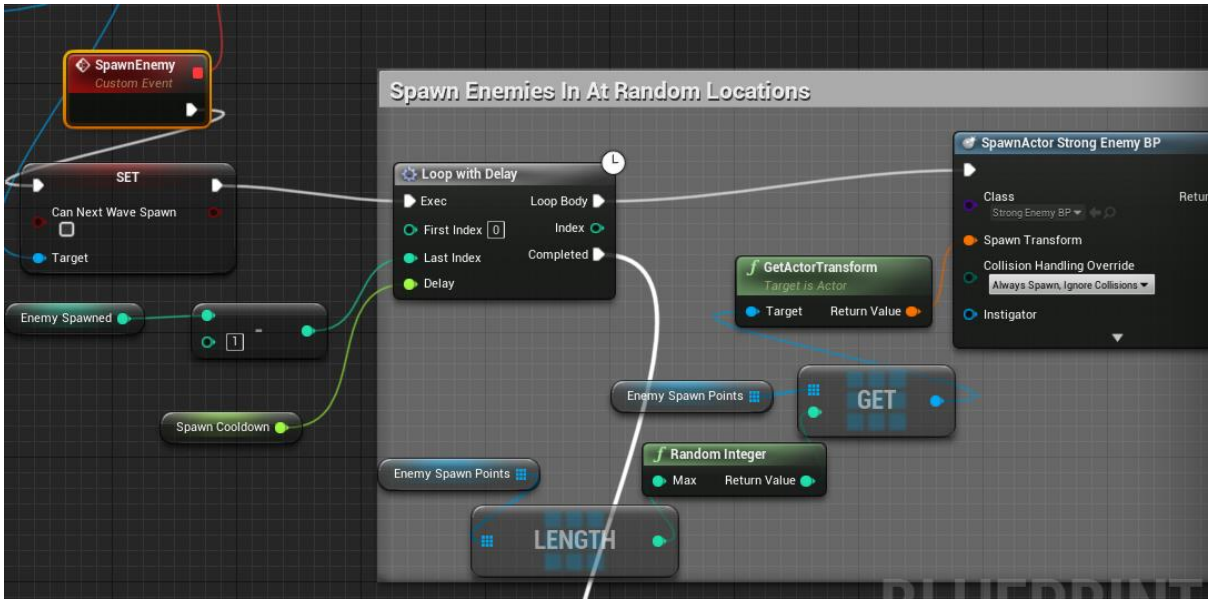
The CheckEnemyLeft() function in the VRGamemode blueprint is what will determine whether or not the game has finished or not. A bunch of branches were used to make sure all the Enemies are spawned in correctly and to make sure the game finishes correctly. The first thing that is checked is if the "CanNextWaveSpawn" variable is true this is to ensure that the wave number cannot change whilst enemies are spawning in, if it is then it will go on next to check if the amount of enemies in the game currently Is 0 this is to make sure that the next wave of enemies don't spawn when there are still enemies on the field.

If there are no enemies on the field then increase the wave number and then the game will do another check to see if the player has completed all the waves so, if the "wavenumber" is more than the "maxwave" variable then that means the game has finished so that means spawn in the LoadLevel blueprint which is just a portal that takes you back to the lobby. If the game is not finished then It will call the NextWave() function which is an Event Dispatcher.

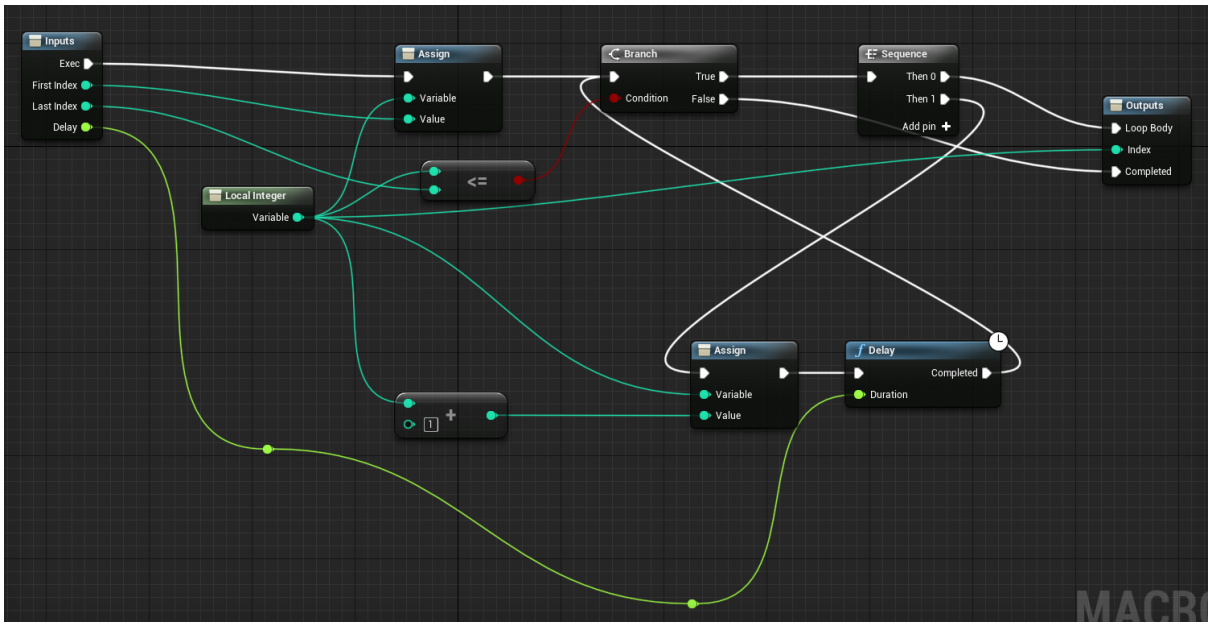


(Figure 71) EventBeginPlay Function, in WeaponSpawner

The event dispatcher will call any function outside of the VRGamemode blueprint that is binded to the Nextwave() function when the Nextwave() function is called. In The Last Arena VR the SpawnEnemy() function inside of the WeaponSpawner Blueprint is binded by the NextWave() function from the VRGamemode Blueprint.



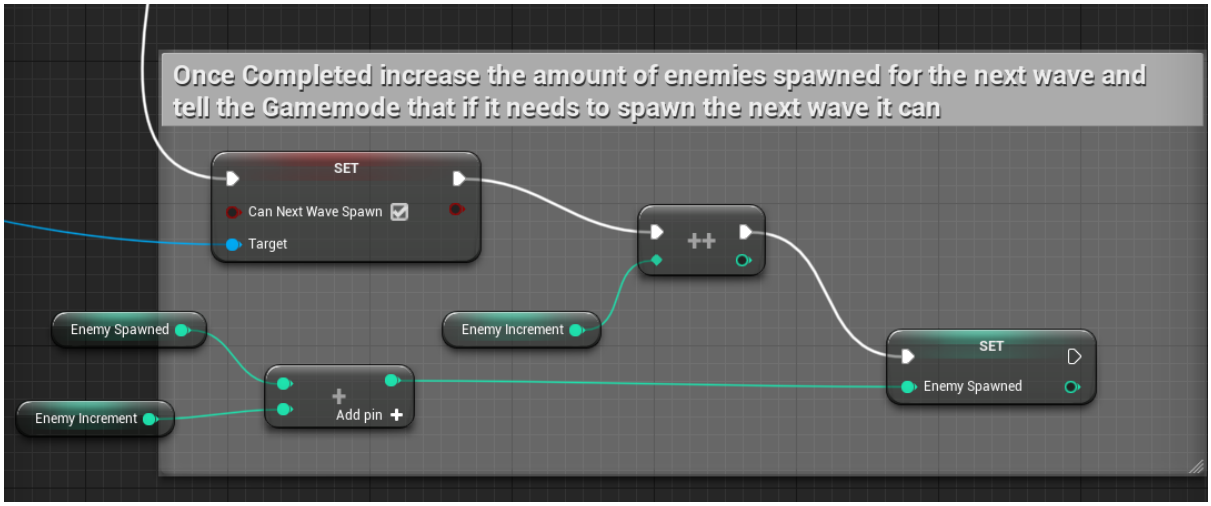
(Figure 71) SpawnEnemy Function, in WeaponSpawner



(Figure 72) SpawnEnemy Function, in WeaponSpawner

The SpawnEnemy() function will first of all make sure the “CanNextWaveSpawn” variable from the VRGamemode is false to make sure another wave cannot be spawned, after that a custom macro named “Loop with delay” which is just the same as the normal loop but the delay node is added.

The For Loop with delay loops around for whatever the value in the “EnemySpawned” variable and the delay lasts for the value in the “SpawnCooldown” variable. For every loop an Enemy is spawned in at one of the random spawnpoints in the map then after the delay will start and the loop will repeat until it is finished.



(Figure 73) SpawnEnemy Function, in WeaponSpawner

Once the loop is completed the "CanNextWaveSpawn" variable will be set back to true and then increase the amount of Enemies that will be spawned.

Testing And Evaluation

Since an Agile method was used to create the Last Arena VR testing happened throughout the project. Whenever a new feature was added it would then be tested and then if it worked properly then it was onto the next feature. Testing was very useful especially when it came to the VR Combat because some weapons would be too strong and make the games very easy and some weapons would be too weak also from the feedback that was given from people who played the game with their VR headset or if they were just watching the game was not only able to improve in a game programming sense but also in a design sense.

Blackbox Testing

Test Name	Expected Output	Actual Output	Comments
Picking Up the weapons with one hand with either hand	Weapon Gets attached to the hand that is overlapping it	Weapon attaches to the hand that is overlapping it	
Drawing the string back of the bow with second hand	Arrow will spawn in and the string will follow the hand of the player but can only go backwards and to a limit	Arrow spawns in and the limit to how far back the string can go is applied	
Releasing the string after it has been drawn back	Arrow should shoot forward and the string will go back to its original location	Arrow is shot forward and string goes back to original location	

Play Testing

Due to the Last Arena VR being a Vr game it was quite difficult to find people to test the game out but I was able to find someone who has access to a VR headset and good feedback given that helped the game progress a lot. The tester had a different VR headeset than what The Last Arena VR was developed with the Vive Pro 2 and the play tester had the Oculus Quest so it was also a good test to see if the game would work on a different VR headset.

Problems that came up during the Playtesting that the player reported

- The player was able to walk through the walls
- If an Enemy is killed before all the enemies spawn making the Enemies alive 0 then the next wave will spawn
- Enemies will continue to spawn in even though the text in the battle arena says that the player has won
- After the Blueprint was changed from a pawn class to a character class the Player couldn't pick up the Bow properly due to the collision
- Enemies would go on top of the melee weapons when colliding
- The arrows were doing too little damage and was taking ages to kill
- Player was unaware of the ability of the Dagger where they can teleport to it
- Player was unaware that you can block with the sword

Positives the player said about the game

- The game is fun to play and challenging
- The player was able to easily navigate through the game and understood how to play
- Game worked fine on the Oculus

Moscow Review

When looking back at the Moscow analysis a lot of the requirements that were planned were added to the game especially the ones in the Must Haves and Should Haves.

Everything from the Must haves were added to a really good standard the only one that could have been improved on was that more variety of weapons could have been added but for what is there now with the Dagger being able to teleport, the Sword being able to block and the Bow being able to attack from range it is pretty good.

Almost everything in the should haves was done the only thing that was left out of the game was the different types of enemies this was due to the fact that adding fully body immersion was very complex and took a lot of time to figure out.

Nothing from the Could haves was added since they were not really important and it was more important to polish the features that were higher in the priority list then adding the features from the could haves.

Functional & Non Functional Requirements Review

Everything inside of the Functional requirements was completed since it was mainly linked with the Must Have list from the Moscow analysis also everything in the functional requirements were the key things that should be added to the Last Arena VR for example, if the game didn't allow the player to pick up items then the enemy wouldn't be able to fight or try out the unique weapons.

All of the Non functional requirements was able to happen throughout the testing and development phases no crashes happened to the game whilst playing it, the only crashes that happened was in

the Unreal editor. The game is challenging which is what the tester of the game gave as feedback, they also said they had a good time playing the game in VR.

Legal, Social, Ethical Implications

PEGI Rating: 12

Target Audience: Young Adults, Casual Gamers, People who want to do exercise

The Last Arena VR has an age rating of PEGI 12, the reason for this is because pegi.info states that a PEGI 12 rating is applied when the video game shows non – realistic violence towards human like character which is something the proposed game will definitely have, all the other stuff that PEGI lists for a PEGI 12 game is pretty much irrelevant to the proposed game.

The reason for the selected target audience is because the proposed game is an intense fighting game which almost all young people enjoy playing and since it's a VR game it will basically be like doing a workout hence the reason why it will be an appealing game to people who like or want to do exercise.

Ethical

The Last Arena VR will be set in a fantasy world using 3D models, the enemy models will vary between humanoid character and non humanoid characters. The game is suited to be played by everyone apart from people who are too old since it requires a lot of movement. The kind of players that will be playing my game will be teens/ young adults of all genders who would like to have a VR fighting experience where they are just defeating enemies, this could also be for people who want to release their frustration or anger.

There will be violence towards humanoid characters but it will not be too realistic like in the previously stated game Gorn, instead of blood coming out when you hit an enemy particle effects will be used instead. There will be no environmental issues in the game since the game will be based in a coliseum type arena and there is no specific social message to get from the game since its just a fun beat em up game.

In the Last Arena VR the player will have to do movements in real life such as, swinging their arms towards the enemy to deal damage to them, make a blocking stance with your arms with the sword, a bow and arrow motion where they have to make movement with their arms where they must put the arrow on the bow then pull back and let go and other real life movements. Players can also throw things such as daggers and magical spells.

Security Implications And Data Protection

The Last Arena VR will not be networked and it will not store any of the players personal details.

Copyright

Most of the assets in the Last Arena VR will be copyright free and obtained from the internet although there is a chance that some of the assets could be self created using software such as

Photoshop for the UI and possibly Blender or Maya for simple 3D objects that are not free on the internet. Even if credit is not needed to be given for the assets from the internet it will still be given.

Some of the coding for the game will most likely come from a tutorial but will be heavily edited to suit the game but the original tutorial will still be given credit.

Game Engine Licenses

The Game engine that will be used to create the proposed game idea will be Unreal Engine 4 and according to the Frequently Asked Questions page on the Unreal Engine website it is free to publish the game but if the game were to be monetised and the gross revenue of the game exceeds $1,000,000 USD then 5% royalty will be given to the inventors of the Unreal Engine.

Critical Review

Summary Of Achievements

I am extremely proud of this project since it was my first ever VR game on the Unreal engine I have little experience with especially when comparing it to Unity. I was able to add all the features for VR to a good standard for example, the VR combat for the Last Arena VR is very fun for me personally and I was told that it was fun by the tester as well. I was able to add in some C++ scripts but not anything advanced but I am still happy with that since its my first time using C++ with Unreal Engine.

What I am most proud about for this project apart from the VR combat was implementing the Full body immersion since there was very little information about it, infact there was not a lot of information on the internet about making a VR game.

Test Results

Testing for The Last Arena VR was super beneficial for the final product, the self-testing in between adding mechanics was helpful since it allowed me to go back and fix any issues before going on to the next mechanics.

The most helpful test results came from the tester which helped fully balance the game out, having the tester play the game instead of self-testing helped realised a lot problems that could not have been found out alone for example, the tester was able to figure out that the player could phase through the walls also a reminder of adding instructions to the lobby was given by the tester.

Time Management

The time management for the start of the game was really good but then other projects were also due and the implementation of the full body immersion took a lot longer than expected. But there was also times that could have been used to do the small parts of the project.

What Did I Learn From This Project

The Last Arena VR was a project that I decided to create to not only improve my understanding of the Unreal Engine but to also get some experience to creating a VR game. I was able to learn what is required to create a VR game for example, one of the features that I was extremely happy implement the full body immersion for that you will need to be able to modify the Ik of the head, foot and hands, you will also need to make sure that the character model will rotate with the VR Origin (VR Headset for the player) and need to make sure it moves with it.

I was also able to find out about Blueprint interfaces and how they are important for communication between different blueprints without understanding this then it would have been extremely difficult to make the weapons be able to be picked up by the player.

I believe one of the most important things that this project made me improve on is the skill to be able research things that isn't really on the internet and then still being able to complete that task for example, there was very little on full body immersion and even though the implementation may not be the best there was I was still able to add it into my very first VR game.

Future Work

There are still a lot of things that can be added to the Last Arena VR such as different types of enemies, different types of weapons along with different maps and the features in the Could Have section of the Moscow analysis can still be added. The Last Arena VR will not be marketed but it could be further improved to not only improve my portfolio but to improve my knowledge of the Unreal engine and VR games, if the decision to not market The Last Arena VR does changes free DLC/ Updates would be a thing I would consider.

In the future I would like to work on the Unreal Engine since after this project I feel a lot more comfortable with it and as for making VR games I would also like to do that too if the opportunity arises.

References

- Answers.unrealengine.com. 2021. Get Actor Component Reference in C++ Class - UE4 AnswerHub. [online] Available at: <https://answers.unrealengine.com/questions/960832/get-actor-component-reference-in-c-class.html> [Accessed 7 November 2021].

- Answers.unrealengine.com. 2021. How can I make one collision just ignore specific actor or collision
- UE4 AnswerHub. [online] Available at: https://answers.unrealengine.com/questions/394231/how-can-i-make-one-collision-just-ignore-specific.html [Accessed 1 December 2021].

- Answers.unrealengine.com. 2021. My funtion is inaccessible (while I think it's not) - UE4 AnswerHub. [online] Available at: <https://answers.unrealengine.com/questions/136632/index.html> [Accessed 7 November 2021].

- Discussion, D. and Programming, C., 2021. logging simple float variables in Tick(float DeltaTime) function causes crash. [online] Unreal Engine Forums. Available at: <https://forums.unrealengine.com/t/logging-simple-float-variables-in-tick-float-deltatime-function-causes-crash/69624/3> [Accessed 8 November 2021].

- Docs.unrealengine.com. 2022. Get Current Level Name. [online] Available at: <https://docs.unrealengine.com/4.27/en-US/BlueprintAPI/Game/GetCurrentLevelName/> [Accessed 7 April 2022].

- Docs.unrealengine.com. 2021. UBoxComponent. [online] Available at: <https://docs.unrealengine.com/4.27/en-US/API/Runtime/Engine/Components/UBoxComponent/> [Accessed 30 November 2021].
- Docs.unrealengine.com. 2022. Set and Get an Actor Reference. [online] Available at: <https://docs.unrealengine.com/4.26/en-US/ProgrammingAndScripting/Blueprints/BP_HowTo/ActorReference/> [Accessed 30 March 2022].

- Engine, U. and Animation, C., 2022. How do i stop idle anim montage. [online] Unreal Engine Forums. Available at: <https://forums.unrealengine.com/t/how-do-i-stop-idle-anim-montage/474968> [Accessed 4 March 2022].

- Engine, U. and Animation, C., 2022. New Save Cached Pose. What does it do?. [online] Unreal Engine Forums. Available at: <https://forums.unrealengine.com/t/new-save-cached-pose-what-does-it-do/407200/2> [Accessed 27 March 2022].

- Engine, U. and Scripting, P., 2022. Making an object dont be affected by Nav Mesh. [online] Unreal Engine Forums. Available at: <https://forums.unrealengine.com/t/making-an-object-dont-be-affected-by-nav-mesh/393805> [Accessed 7 April 2022].

- Patreon. 2021. VR bow tutorial assets. [online] Available at: <https://www.patreon.com/posts/52289259> [Accessed 20 November 2021].

- Reddit.com. 2022. [Help] AI needs to learn about personal space, help please!. [online] Available at: <https://www.reddit.com/r/unrealengine/comments/6zmpxg/help_ai_needs_to_learn_about_personal_space_help/> [Accessed 3 April 2022].

- Reddit.com. 2021. [online] Available at: <https://www.reddit.com/r/unrealengine/comments/9osfxo/help_how_to_get_the_actor_tag_in_c/> [Accessed 25 November 2021].

- Mixamo.com. 2021. Mixamo. [online] Available at: <https://www.mixamo.com/#/> [Accessed 8 December 2021].

- Unreal Answers. 2021. How do I remove a C++ class from my project code?. [online] Available at: <https://answers.unrealengine.com/questions/166179/how-do-i-remove-a-c-class-from-my-project-code.html> [Accessed 30 November 2021].

- Unreal Engine. 2021. Free Fantasy Weapon Sample Pack in Weapons - UE Marketplace. [online] Available at: <https://www.unrealengine.com/marketplace/en-US/product/e4494c76c3b348aba7ef9b263a6dd496> [Accessed 7 December 2021].

- Versluis, J., 2021. How to tag an actor in Unreal Engine. [online] JAY VERSLUIS. Available at: <https://www.versluis.com/2020/10/how-to-tag-an-actor-in-unreal-engine/> [Accessed 25 November 2021].

- Youtube.com. 2021. C++ Health & Damage System UE4 / Unreal Engine 4 C++. [online] Available at: <https://www.youtube.com/watch?v=oAswbd1Ngls&t=216s&ab_channel=DevEnabled> [Accessed 5 November 2021].

- Youtube.com. 2021. Calling Blueprint Functions inside of C++ Scripts | Unreal Tutorial. [online] Available at: <https://www.youtube.com/watch?v=WK1HsgF2eUM&t=131s&ab_channel=KeySmashStudios> [Accessed 7 November 2021].

- Youtube.com. 2021. Create A Platformer Pt 4: Chasing Knockback Enemy! UE4 Tutorial. [online] Available at: https://www.youtube.com/watch?v=4UsaiOEr6Bw&ab_channel=Jackson%E2%80%99sGames [Accessed 1 December 2021].

- Youtube.com. 2022. Discord Help, How To Load A New Level And Navigate In VR With UE4. [online] Available at: <https://www.youtube.com/watch?v=P0cs1Qm1HQk&ab_channel=GDXR> [Accessed 7 April 2022].

- Youtube.com. 2022. Enemy Horde Wave System - Unreal Engine Tutorial. [online] Available at: <https://www.youtube.com/watch?v=gP2ozW0dLZ0&t=84s&ab_channel=MattAspland> [Accessed 29 March

- Youtube.com. 2021. How to make a Teleporter in C++ | Unreal Tutorial. [online] Available at: <https://www.youtube.com/watch?v=K-pIBEhrXPg&t=759s&ab_channel=KeySmashStudios> [Accessed 7 November 2021].

- Youtube.com. 2022. How To Make an IK Setup for VR In Unreal Engine 4 (Full Body Room Scale). [online] Available at: <https://www.youtube.com/watch?v=qBooEZnlAA4&t=1811s&ab_channel=KazVoeten> [Accessed 26 January 2022].

- Youtube.com. 2022. How to make and use an ENUMERATION || UE4 TUTORIALS. [online] Available at: <https://www.youtube.com/watch?v=boZBN_WgSUA&ab_channel=TechnoNerd> [Accessed 28 February 2022].

- Youtube.com. 2022. How To Use Upper Body Montages - Unreal Engine 4 Tutorial. [online] Available at: <https://www.youtube.com/watch?v=bU95pf5hTkI&ab_channel=MattAspland> [Accessed 2 March 2022].

- Youtube.com. 2021. Must have VR expansion plugin for UE4 VR developers. [online] Available at: <https://www.youtube.com/watch?v=WDAZU2EeOM0&ab_channel=Sir_FansiGamedev> [Accessed 10 November 2021].

- Youtube.com. 2021. Perfect VR weapons grabbing tutorial - Unreal Engine 4. [online] Available at: <https://www.youtube.com/watch?v=K0L5bnwq4lo&ab_channel=MarcoGhislanzoni> [Accessed 3 November 2021].

- Youtube.com. 2022. Super-easy VR body with Arm IK and thumbstick locomotion - Unreal Engine 4 Tutorial. [online] Available at: <https://www.youtube.com/watch?v=EKR8ogonD68&t=13s&ab_channel=MarcoGhislanzoni> [Accessed 24 January 2022].

- Youtube.com. 2022. Two hands weapon handling in VR tutorial - Unreal Engine 4. [online] Available at: <https://www.youtube.com/watch?v=wxKYgJmc5a8&ab_channel=MarcoGhislanzoni> [Accessed 6 April 2022].

- Youtube.com. 2021. UE4.14.3-Vive/VRCollision-Part2 Basic Collision. [online] Available at: <https://www.youtube.com/watch?v=XeRv8sb1sdM&ab_channel=JonasM%C3%B8lgaard> [Accessed 5 November 2021].

- Youtube.com. 2021. UE4 Tutorial: Advanced Movement (VR). [online] Available at: <https://www.youtube.com/watch?v=zlzBifkjXDk&ab_channel=underscore> [Accessed 3 November 2021].

- Youtube.com. 2022. UE4 quick tutorial: Red hurt screen widget. [online] Available at: <https://www.youtube.com/watch?v=DmvY3nceH88&ab_channel=AlenLoebUE4> [Accessed 2 April 2022].

- Youtube.com. 2022. Unreal Engine 4 - Use Anim Montages for Non-Character Animations. So Powerful!. [online] Available at: <https://www.youtube.com/watch?v=_RU_-yNpEZc&ab_channel=TorQueMoD> [Accessed 4 March 2022].

- Youtube.com. 2022. Unreal Engine 4 Tutorial: For Each Loop With Delay. [online] Available at: <https://www.youtube.com/watch?v=zh8xx9abzZY> [Accessed 30 March 2022].

- Youtube.com. 2021. VIVE Pro 2 Setting up. [online] Available at: <https://www.youtube.com/watch?v=VAcPyq6UTws&ab_channel=HTCVIVE> [Accessed 3 November 2021].

- Youtube.com. 2021. VR Bow and Arrow in Unreal Engine 1/2. [online] Available at: <https://www.youtube.com/watch?v=6S4IfgPbPcY&ab_channel=Just2Devs> [Accessed 20 November 2021].

- Youtube.com. 2021. VR Bow and Arrow in Unreal Engine 2/2. [online] Available at: <https://www.youtube.com/watch?v=FrCONdyr9vg&ab_channel=Just2Devs> [Accessed 24 November 2021].

- Youtube.com. 2022. What Does "Accessed None Trying To Read Property" Error Mean And How To Fix It - Unreal Engine. [online] Available at: <https://www.youtube.com/watch?v=x5yx7XqZrrA&ab_channel=MattAspland> [Accessed 6 April 2022].