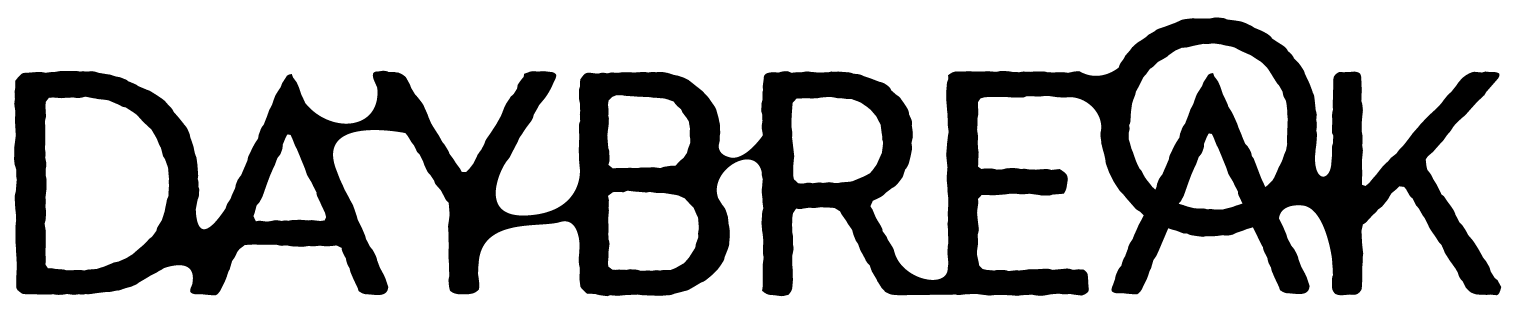
**FINAL YEAR PROJECT**

**TECHNICAL DESIGN DOCUMENT**

****

**[UE2]**

**Developer - Winston (212668S)**

**Developer - Jiaxu (210681N)**

**Developer - Yokawa**

**Artist - XinYing (212035K)**

**Artist - Kerrin (213155U)**

**Artist - John (Admin No)**

**TABLE OF CONTENTS**

[**1.0. Software Architecture 3**](#_heading=h.30j0zll)

[**2.0. Naming Convention 4**](#_heading=h.3oq19qnhuzvq)

[**3.0. Features 4**](#_heading=h.pxsd0rw1i2ho)

[3.1. Collision Detection 4](#_heading=h.4d34og8)

[3.2. AI State Machines 4](#_heading=h.2s8eyo1)

[Detection Sequence 6](#_heading=h.9eknfp5ffdre)

[Human AI 8](#_heading=h.grpi488pqkul)

[Thrull AI 8](#_heading=h.f8snye2qaqv2)

[Vampire AI 10](#_heading=h.zij1y329udae)

[3.3. Abilities and Objects 10](#_heading=h.zbbgtdb4t1au)

[3.4. Player Detection and Post Process 12](#_heading=h.b2aghn1sqsj4)

[Line of Sight 12](#_heading=h.m6npddo6xxvb)

[Xray Ability 13](#_heading=h.bsn4kxrcz8bk)

[3.5. Aiming and Teleportation 13](#_heading=h.f9z3z4wsj9xr)

[3.6. Camera Controls 14](#_heading=h.bfulhg5lauv)

[**4.0. Installation Manual 14**](#_heading=h.3rdcrjn)

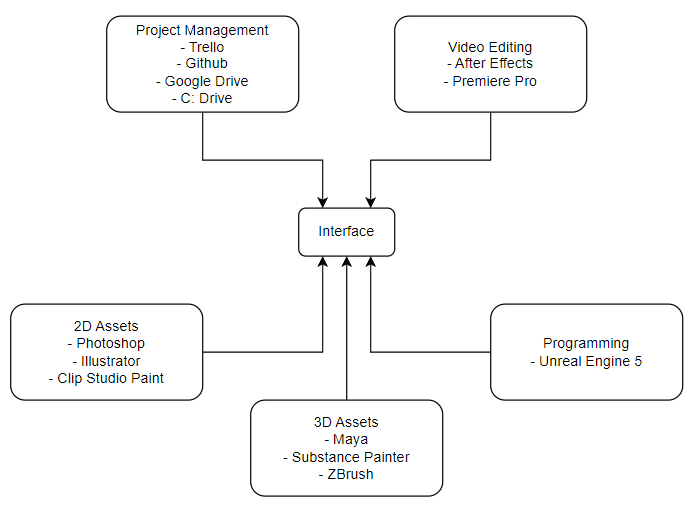
[4.1. Build steps 15](#_heading=h.26in1rg)

[4.2. Installation steps 15](#_heading=h.lnxbz9)

[4.3. Hardware and software requirements 15](#_heading=h.35nkun2)

[**5.0. 15**](#_heading=h.a7vcbnv95voh)

## Software Architecture



## Naming Convention

| Asset | Prefix |
| --- | --- |
| Blueprint |  |
| Animation Blueprint | ABP |
| Actor | AC |
| Actor Component | ACC |
| Blueprint | BP |
| Blueprint Interfaces | BPI |
| Enum | E |
| Structure | S |
| Widget | W |
| **Assets** |  |
| Material | M |
| Material Instance | MI |
| Skeletal Mesh | SKM |
| Static Mesh | SM |
| Texture | T |
| **Sound** |  |
| Sound Class | SC |
| Sound Cue | Cue |
| Misc |  |
| Niagara System | NS |
| Widget Blueprint | WBP |
| Datatable | DT |

## Features

### Collision Detection

Subject detection uses UE5’s physics as well as chaos solver. all channels have been tweaked to allow more customization between traces and collisions.

Trace channels: Visibility, Camera and Interaction.

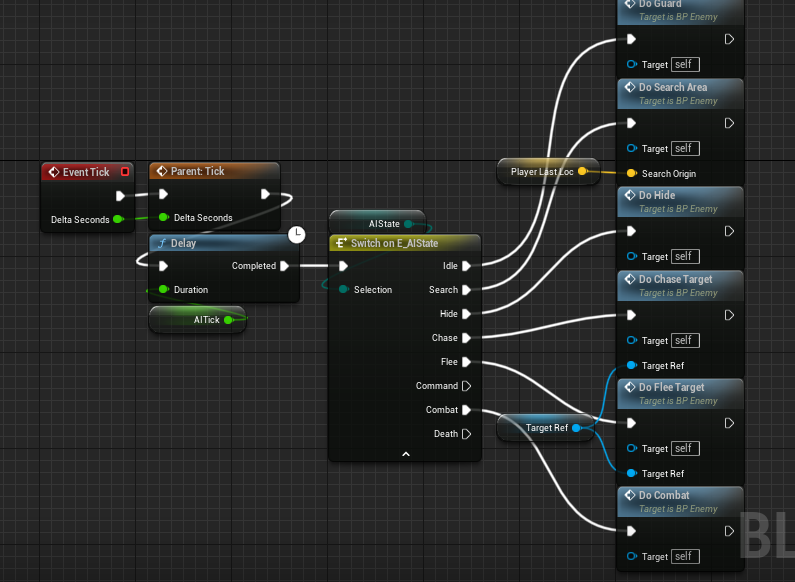
Collision Channels; WorldStatic, WorldDynamic, Pawn, PhysicsBody, Vehicle and Destructible.

Presets are also used to facilitate faster prototyping as they reuse and standardize collision channels between entities. Despite the changes and customization between units, I have relied on inheritance to allow for entities to reflect their parent’s collision state, thus by using both methods, have achieved a faster and standardized collision choice.

### AI State Machines

Hybrid of decision tree and FSM, dubbed State tree, with self learnt implementation of code alongside the blueprint systems of UE5. States are as follows, with decision branches and outlined designer friendly code bodies. Each enemy that uses the base finite states uses events and functions with clear wording and description, allowing a designer to implement them easily.

Example code implementation, with segmented behaviors and throttled performance with the introduction of AITicks.



The specific states are: Idle, Search, Hide, Chase, Flee, Command, Combat and Death. They have a few different detection mechanics, but loosely based on the environment query system, self coded as before, as well as splinter cell’s bone detection. This allows for a more robust detection system, at the cost of more intensive calculations, and by extension, lower performance. This returns back to the reason for AITicks, delayed up to 20 ticks per second as well as an activation condition, now based on the level and quadrant the player is in. Despite following hitman’s AI performance methods, I am unable to replicate the complex pathing and NPC sim behavior with interactions and delays.

Main Behaviors gathered from the above states:

Guardsmen - stands at a location, rotating camera slightly, chases on sight, will activate alarm system after a period of time, alerting a large number of units around the level.

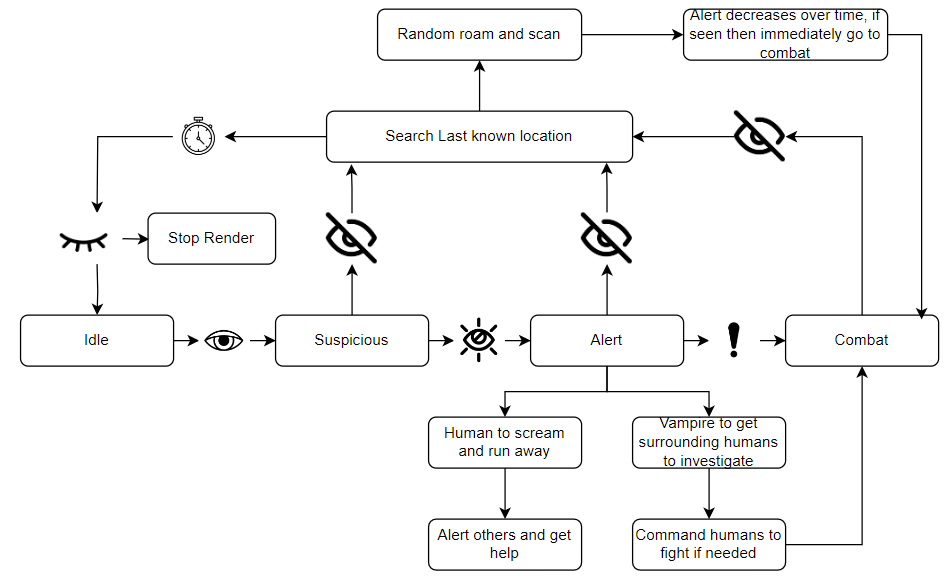
Worker - stands at a location, engrossed in work, flee on sight, gets help from other units(except from thrull).

Patrolling Guard - strict patrol, random patrol and random roam. all 3 behaviors select locations to move to before stopping and performing a task there or to scan the location. the behavior reflects the guardsmen where they’ll be rather aggressive in exposing and alerting others.

Idler - a sleeping guard or worker that ignores the player. line of sight is disabled and sound threshold is increased to allow some actions around them. will awake shocked before shouting, alerting others.

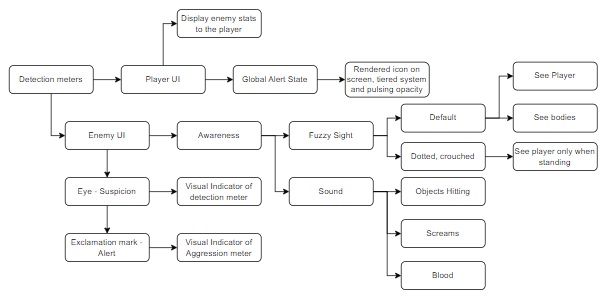
different entities behave slightly differently after getting alerted, such as the thrull attacking on sight and screaming, or the more defensive guard to alert others and get help afterwards.

Unique NPCs such as the Vampire have not been added yet. Following the provided diagram below, it shows the generalization of all units’ actions in a controlled loop, where the actual behavior sequence is more varied and branches out further before looping back.



#### Detection Sequence

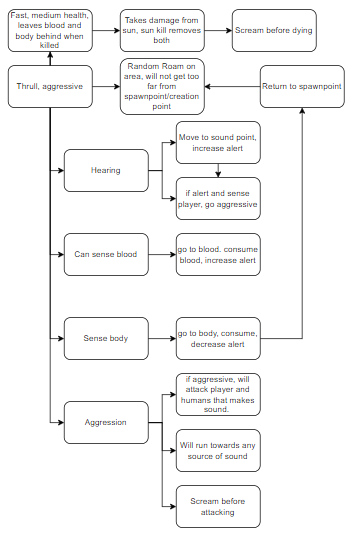
As stated above, the characters detect via bones as well as contextual and environmental cues. This leads to the balancing of the detection to be more forgiving to the player. There is a meter above their head denoting the amount of detection, visualizing the awareness of the Unit. There is also a fuzzy detection to ensure accuracy of the detection, granting around 50% of the body invisibility while standing and 100% of the body while crouching. Sound detection is stricter due to a range detection and path distance from the source, this ensures that units cannot hear you through walls, although minor movements while hiding can alert them. Crouching also reduces the sound you produce, granting a smaller detectable radius.



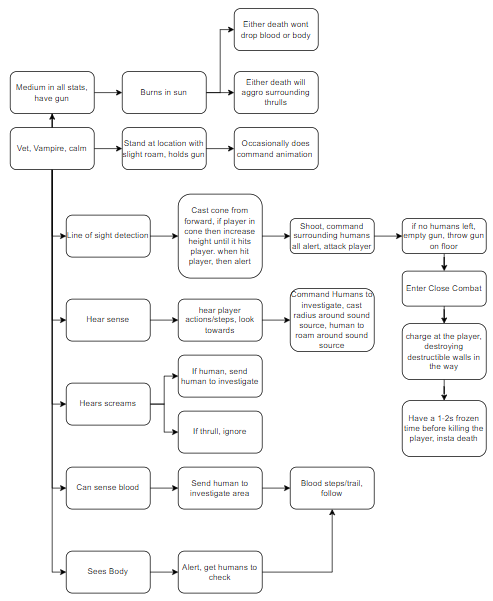
#### Human AI

#### 

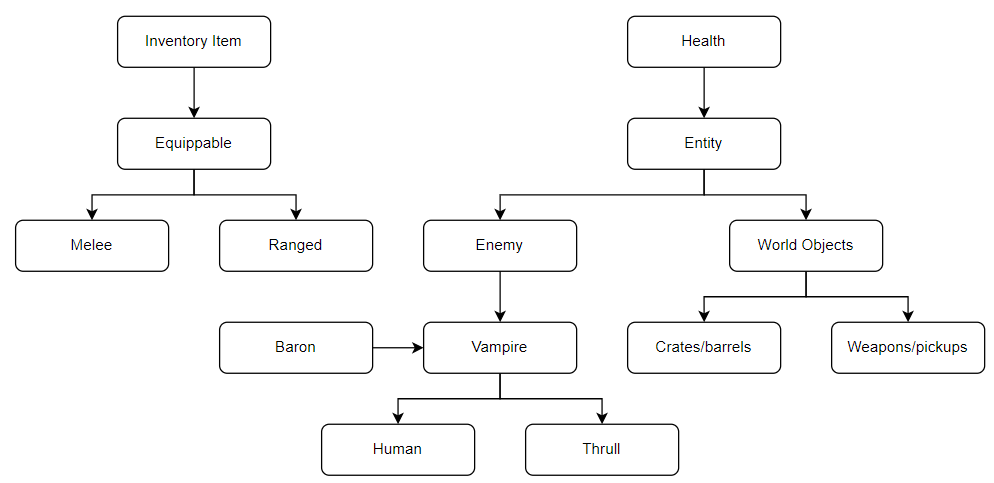
#### Thrull AI

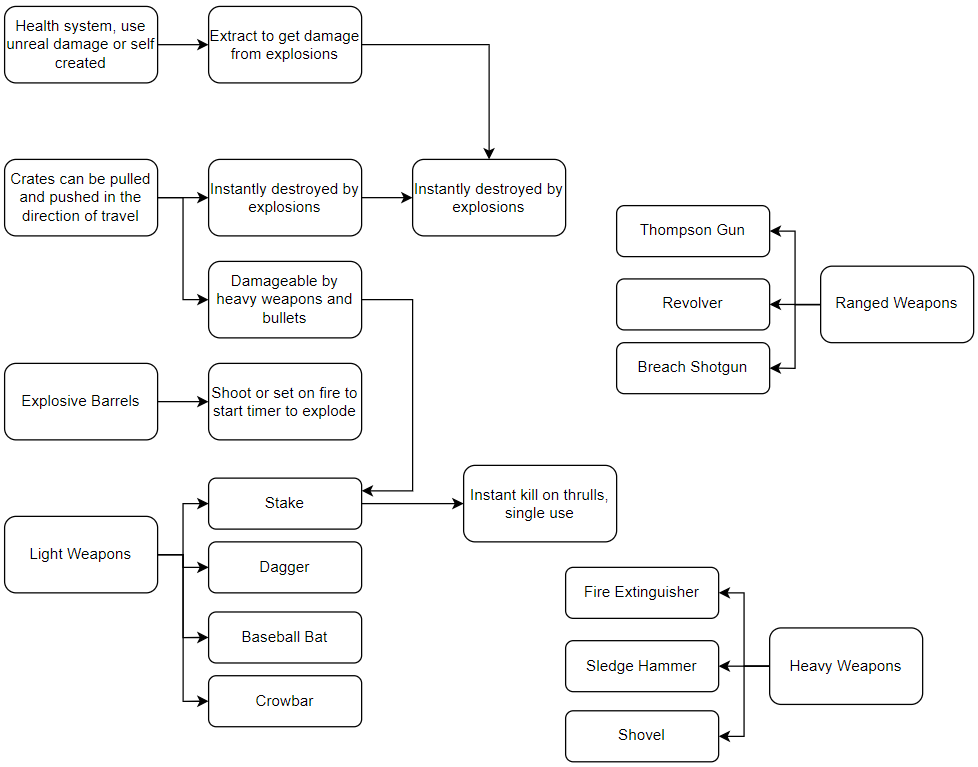


#### Vampire AI

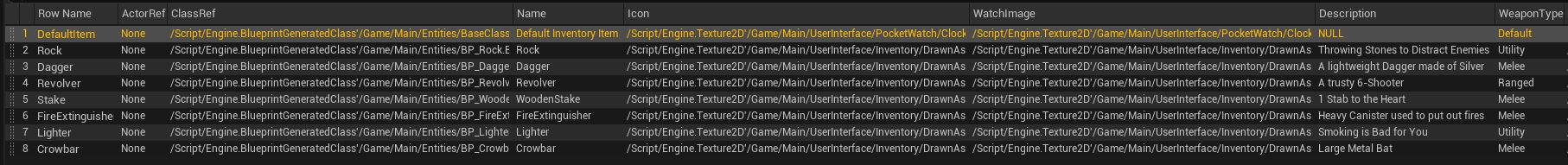


### Abilities and Objects



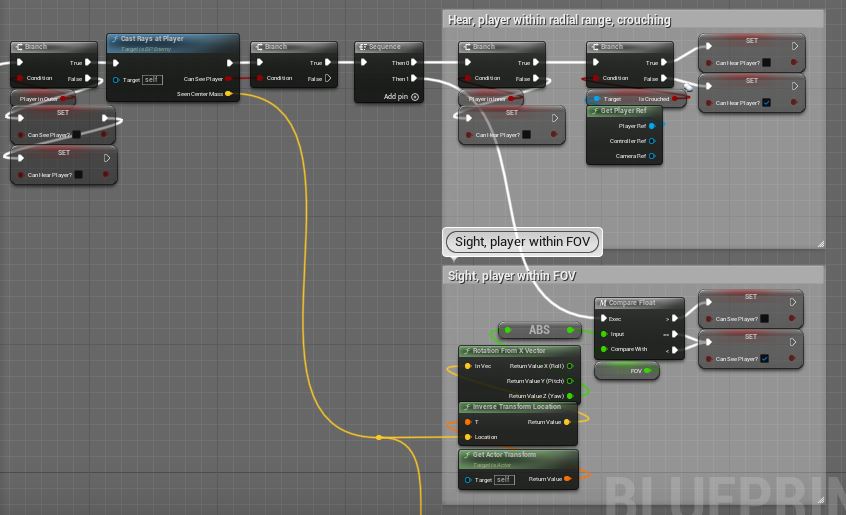


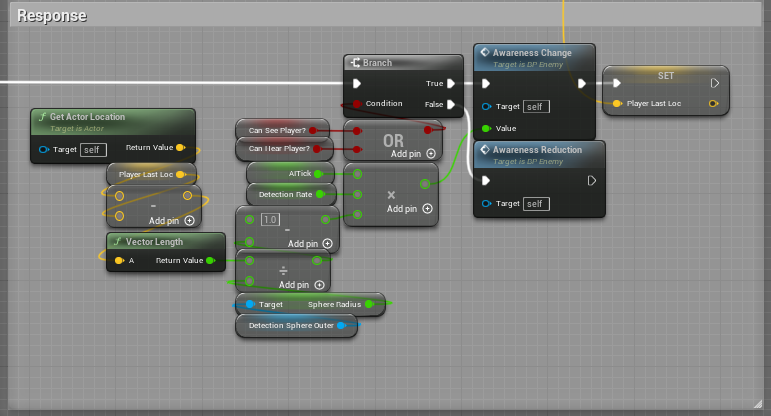
Interfaces used to generalize interactions and use scenarios between weapons, utilities, interactables and inventory items. This is more readable and highly scalable to include behaviors from inherited interfaces and inherited code, or to overload it with unique code from the children’s own use case.



### Player Detection and Post Process

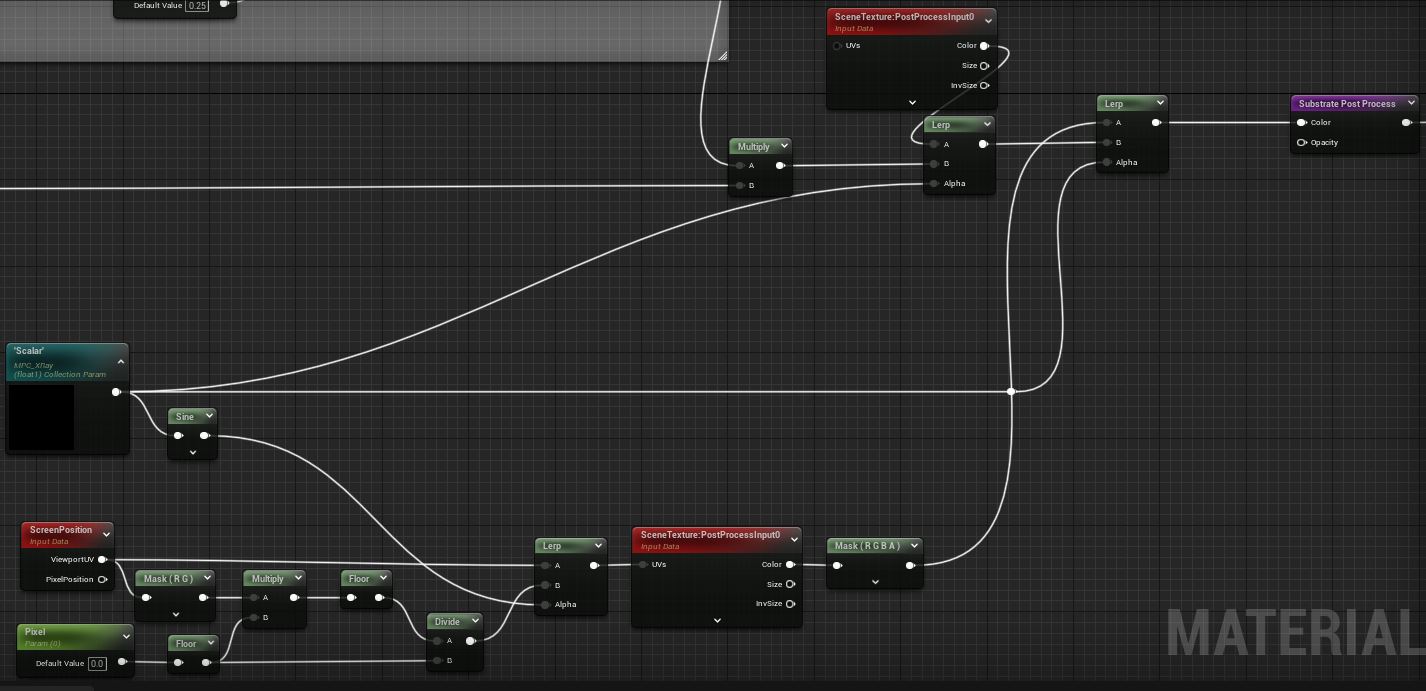
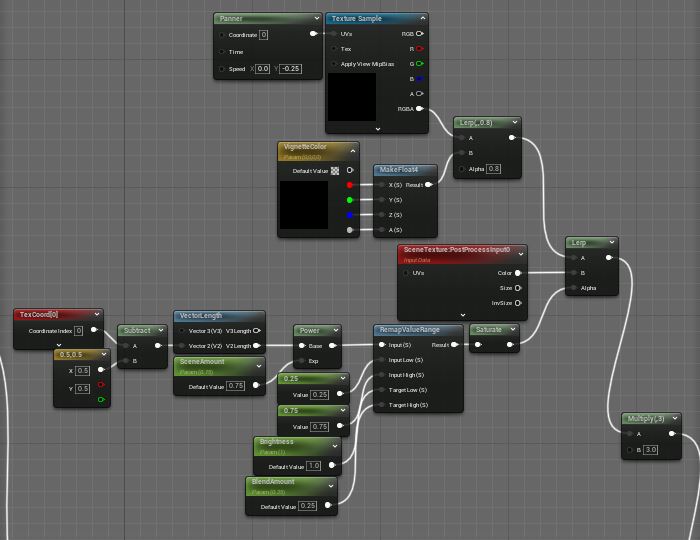
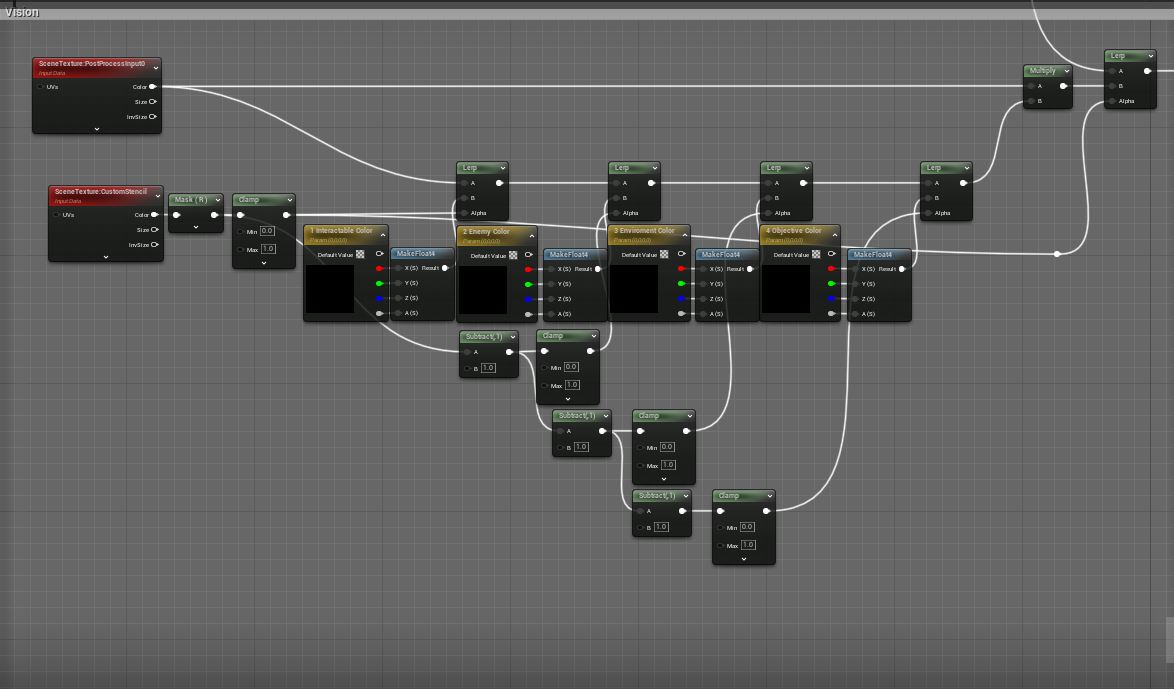
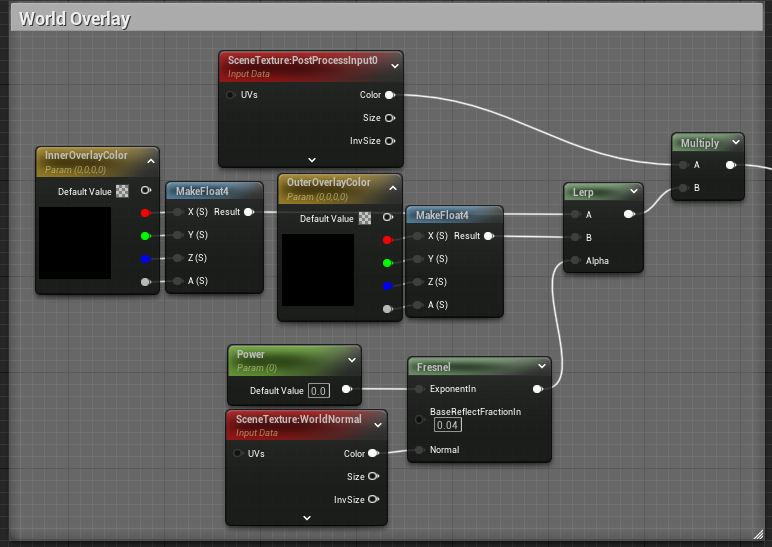
#### Line of Sight



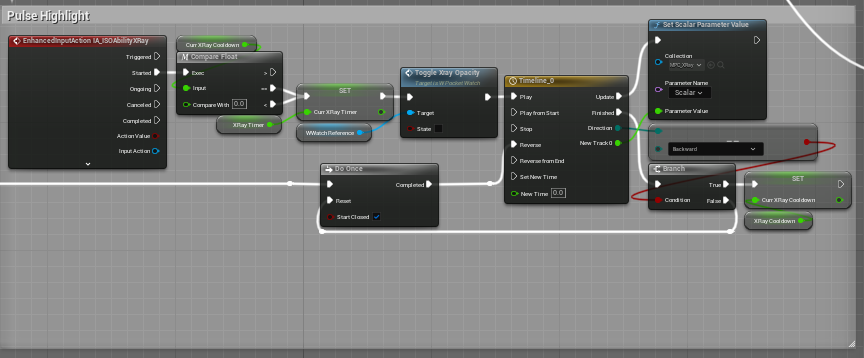


Applied OpenGL technique to generate multiple triangles using player ray casts from the eye level, then assigning it a forward direction using the head bone’s rotation. This allowed me to generate an image which I passed into a post process function where i used the inverse alpha of the image to blur and darken the environment, then assigned the player’s position to the post process to anchor it onto the player. To remove tiling, I increased the image texture size to the maximum limit of 2048 pixels.

#### Xray Ability

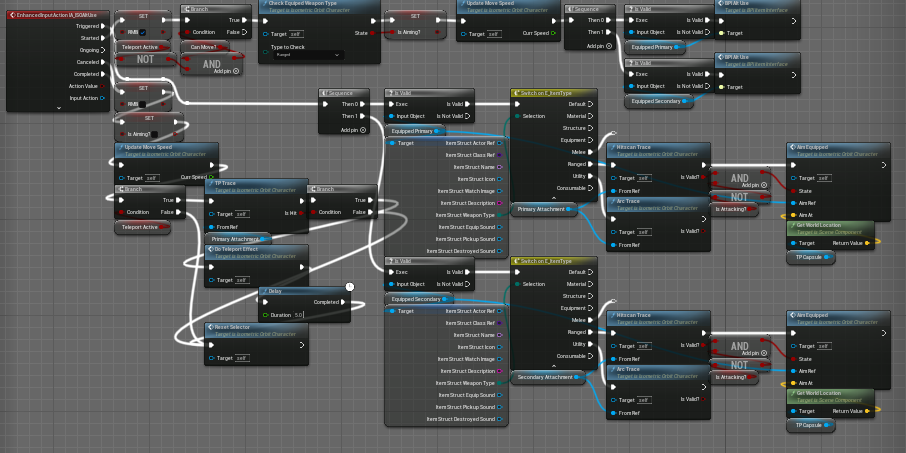


Use a post process to identify the custom stencil depth of entities, then render them after everything else with a pre-allocated color channel. Important to note that the effect can also be a material, allowing for custom glows and patterns appearing on theX-Ray, like holographic or warped perception

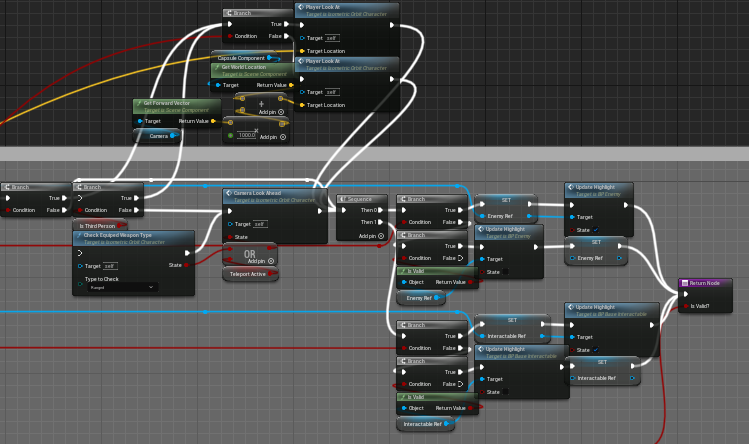
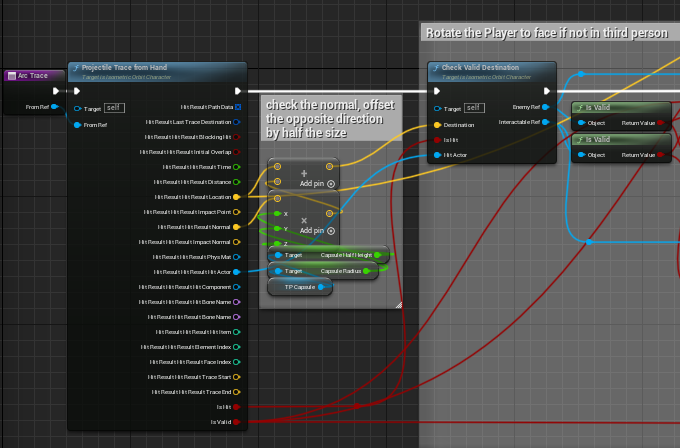
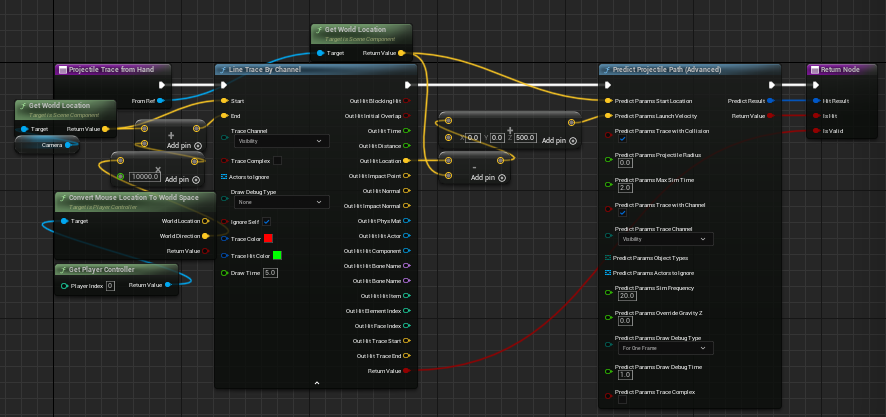
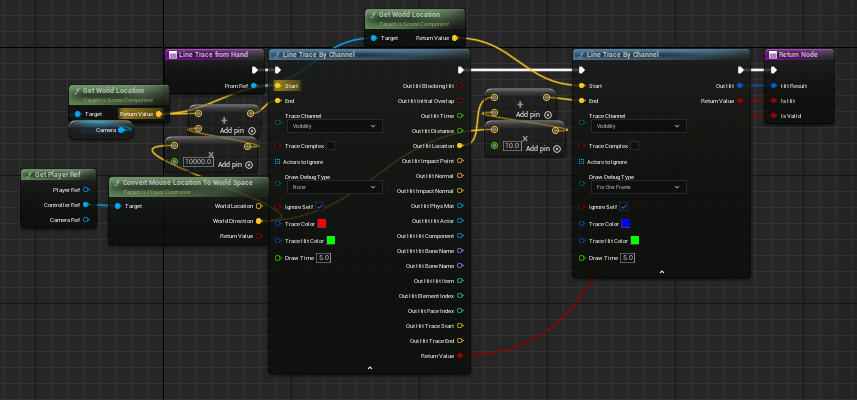


Use of timeline to trigger transitions between on and off state, with little logic code to enable a do-once to prevent spam activation of ability. Trigger state is easily controlled as well as the resulting effect and duration of the ability and the transitions

### Aiming and Teleportation



Use of a 2 staged detection for isometric orbit camera allows the player to select a point on screen before the hand casts a separate ray to check if the hit location is valid as well as any intersections between the hand and the target location, compounding this with the line of sight code allows for a robust detection of abilities and interactions that is more accurate as well as intuitive for the player, as there are more detectable line renderers to highlight and direct attention to. future development could replace the line renderer with a stylized beam to make it more coherent in art style.



Use of a math based approach to handle aiming via a separate player capsule to offset it to ensure that the player can teleport safely without glitching through objects. such as getting the capsule dimensions before offsetting it based on the hit to find a non-intersecting hit location for the player to teleport to.

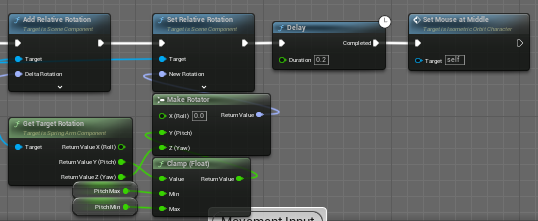
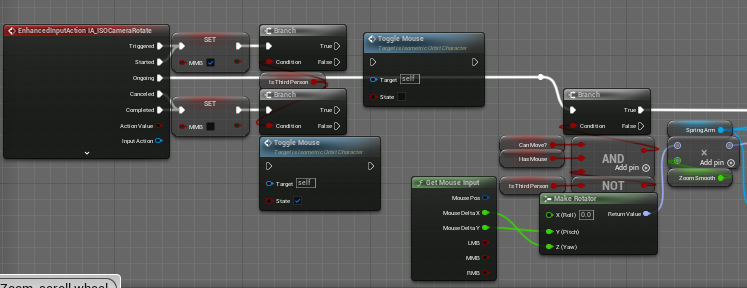
Teleportation of the player was inspired by vampire mist walking, where the player turns to smoke or mist before moving around. Balancing this ability was hard due to the nature of stealth games, where we sought to change the ability to teleportation with flair. the player now channels a teleportation while enveloping both the body and the target in smoke, then teleporting, allowing the player to select and display the target location as a selecting phase before the actual ability.

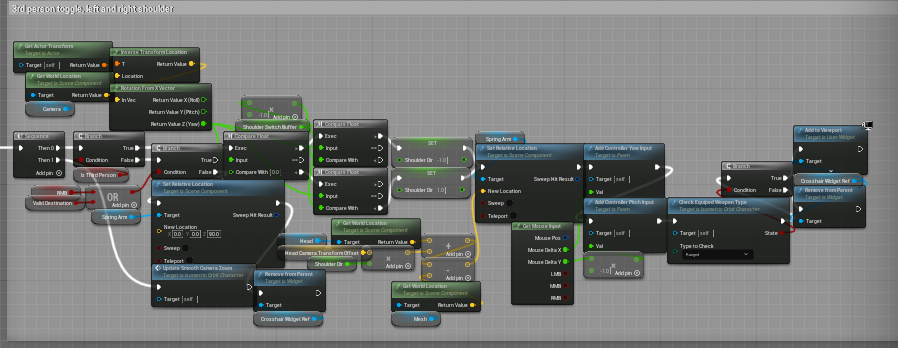


Above are the particle effects used to hide the player’s body, attached to the player’s skeleton as well as the second capsule reused to target objects. reusing the capsule meant that the ability has negligible performance impact while also improving the accuracy of the effect to the hit location.

### Camera Controls

The player camera is custom created to facilitate 3rd person view as well as isometric orbit, blending in styles from Hitman and Desperado. This allows more information to be gathered while not being too foreign to the player.





## 

## 