**Documento de implementación del juego**

Tipo de IA: AI-Based Game Design.

Plataforma: Samsung Gear VR

Engine: Unreal Engine 4.

Lógica de IA: Npcs con diferentes personalidades seleccionadas de manera random que merodean por un determinado espacio. En un determinado momento del juego, uno de ellos se vuelve un criminal e intenta robar un objeto.

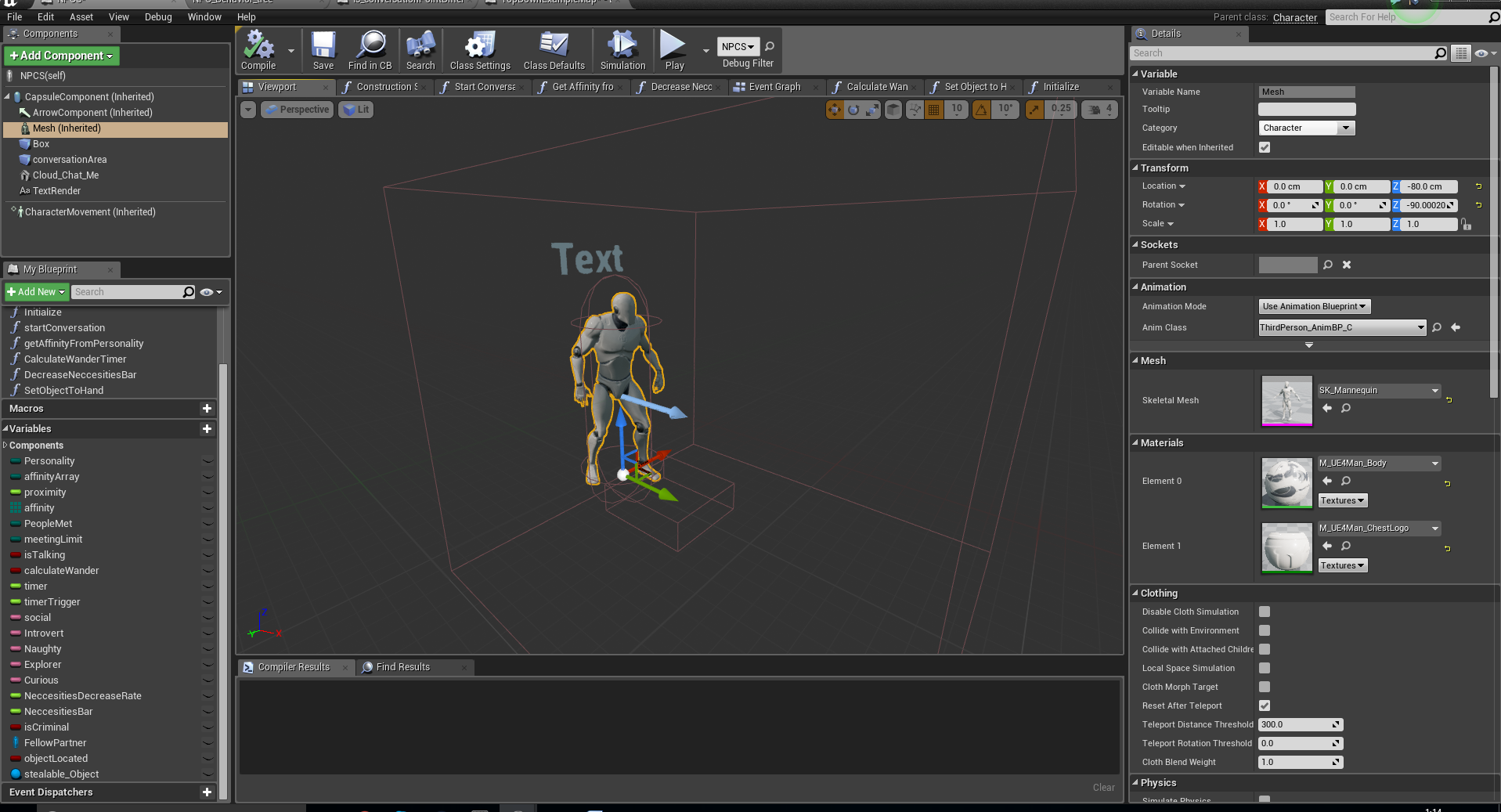
Lógica del juego: Pillar al criminal antes de que se acabe el tiempo, sin acusar a un inocente.

**Arquitectura Interna del Juego.**

Al estar desarrollado en UE4 el diagrama de bloques se vuelve algo enrevesado, por eso, intentaré explicarlo de una manera parecida, ya que debido al caso creo que es lo mejor.

Existen tres clases prioritarias para el funcionamiento del juego:

**NPCS**: Son los agentes de IA principales del juego, funcionan gracias a un Behavior Tree y funciones internas.



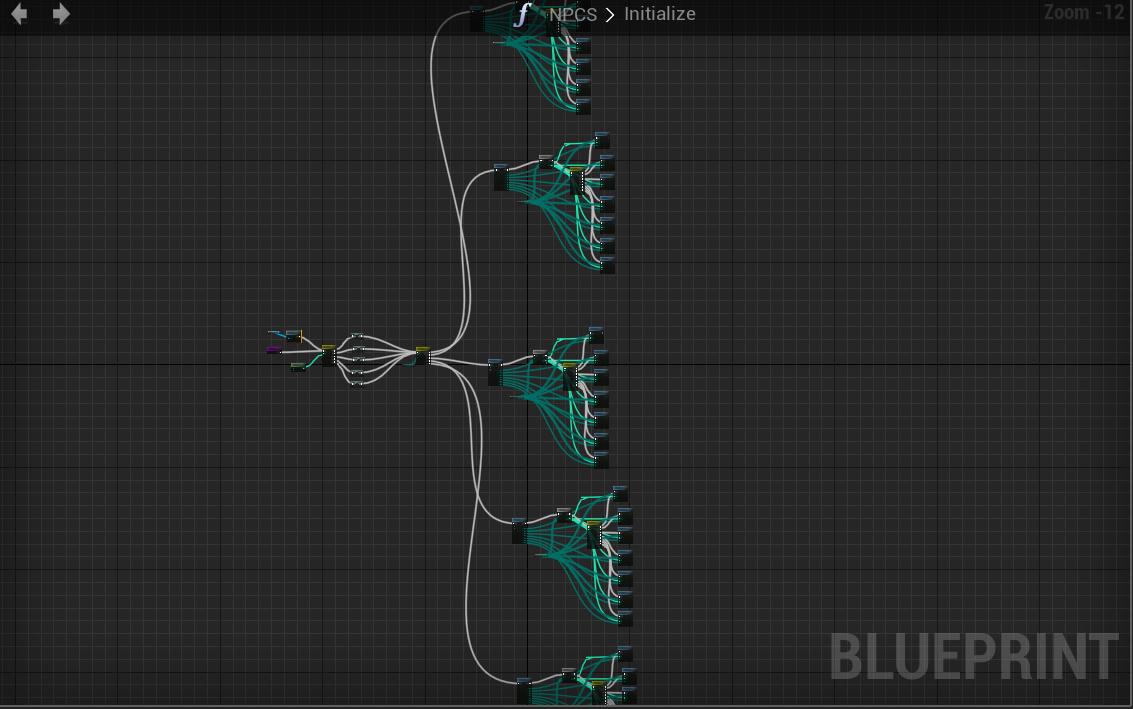
**NPC\_Manager:** Clase encargada de dirigir toda lógica del juego. Decide qué usuario se vuelve criminal dentro de los npcs en el juego y activa una alarma cuándo el criminal roba un objeto.

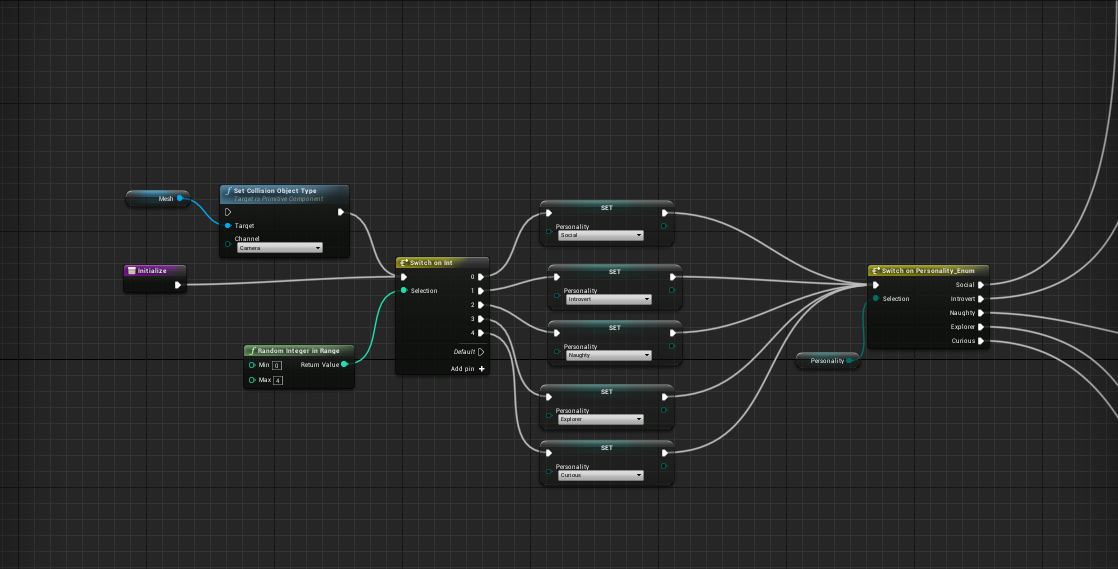
**NPCS**

**Movimiento**: Del movimiento no hay mucho que decir ya que se encarga de todo unreal recreando una navegable mesh, lo único que he cambiado por motivos de gameplay es que en los BT he implementado una función propia de Move To y Wander que explicaré más adelante.

**Representación:** Los Npcs al comenzar el juego poseen la información guardada en una base de datos de la afinidad que poseen con otros npcs, según la combinación de personalidades el valor será mayor o inferior.

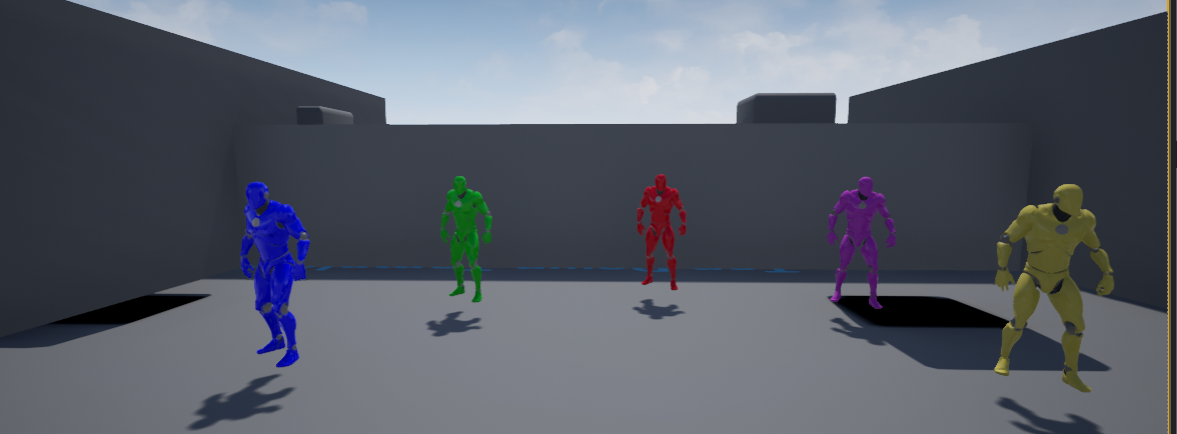


Al comenzar el juego el propio NPC hará un random para generarse su propia personalidad, esto está dentro de la función Initialize de la clase Npcs. Una vez elegida su personalidad, se revisa la base de datos para generar guardar en un array estático las afinidades que posee con otras personalidades, de esta manera, cada vez que se quiera revisar la afinidad no se tenga que ir a mirar en la base de datos. 



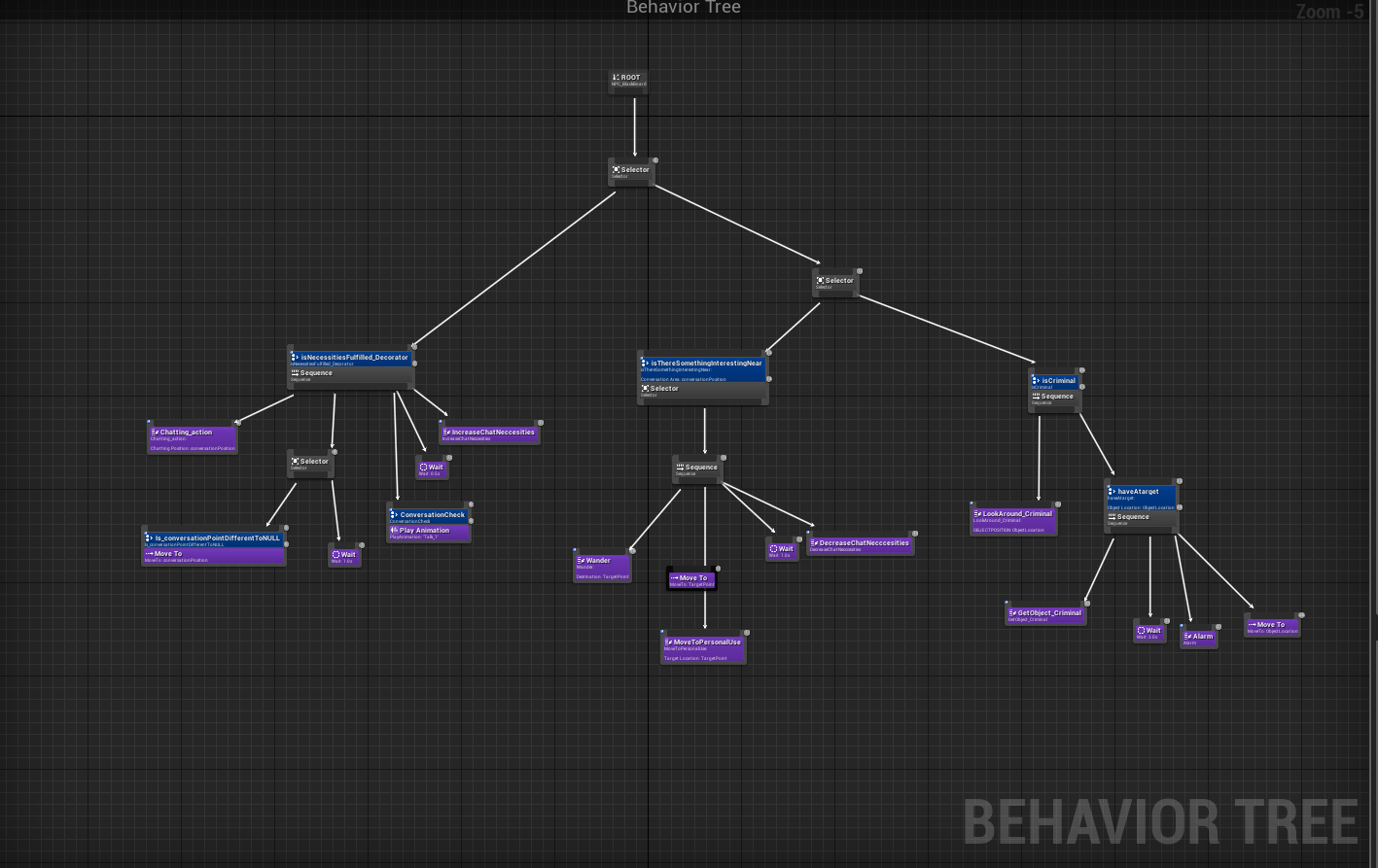
Una vez ejecutada la función Initialize(), el npc cambiará su material según su personalidad (por cuestiones de feedback), también se le dará un decrease Rate de la necesidad de socializar según su personalidad.

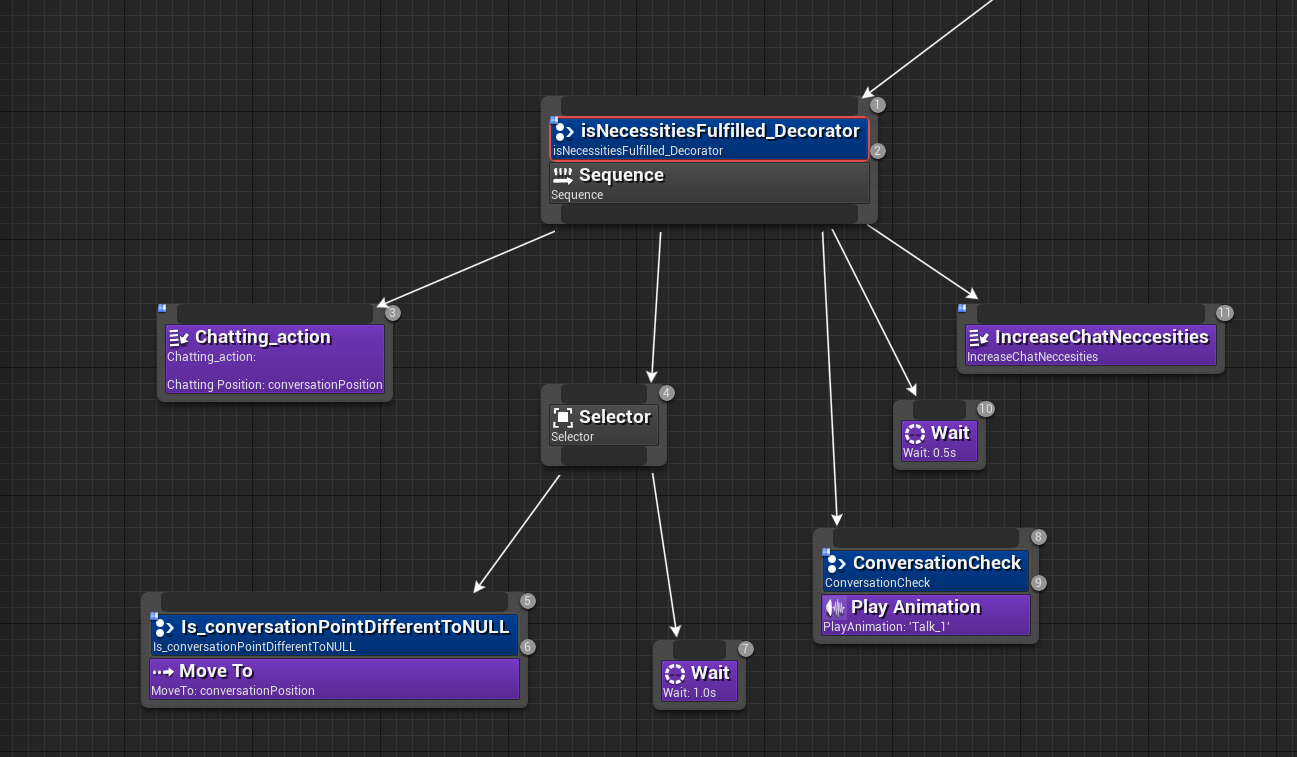
Existen 5 personalidades de NPC ( Social [Rojo], Introvertido[Azul], Naughty[Amarillo], Explorer[Verde] y Curious[Rosa]). Estas personalidades significan principalmente en la actualidad del juego la frecuencia con la que interactuarán con otros NPCs, siendo por ejemplo el social que más afinidad tiene con todos y el que mayor decrease Rate tiene, por lo que será el que mayor necesidad tendrá de interactuar, en su contraparte se encontrará el introvertido.



**Behavior Tree & BlackBoard**

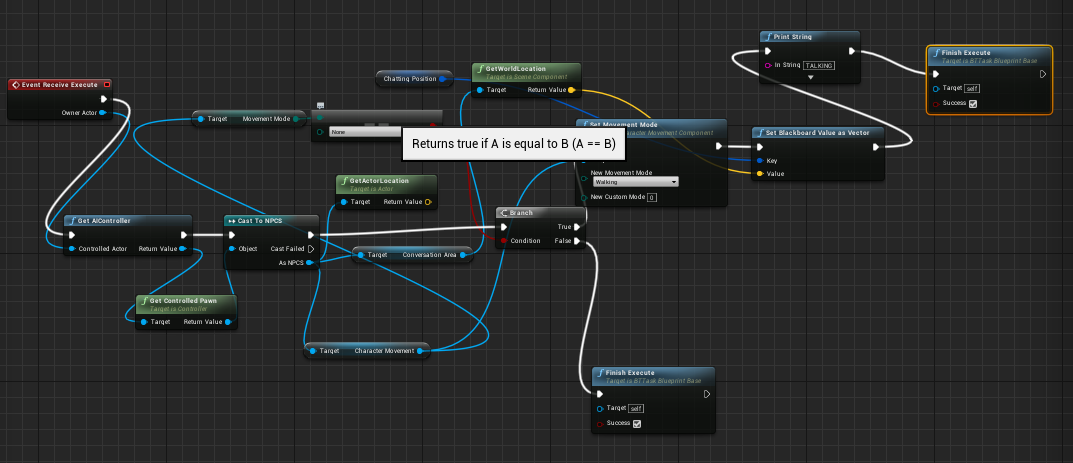
Cada Npc tiene su propia Blackboard y ejecución de BT, que guía completamente su comportamiento. Dejar Claro que todos los nodos excepto el moveTo, selector, sequence han sido creados por mí.





El primer nodo que revisa el selector es un nodo de secuencia en el que se encuentra un decorator que comprueba si el personaje tiene conversación o no. En caso de tener alguien con quién conversará retornara SUCCESS otherwise FAILURE.

Al retornar SUCCESS seguimos en el nodo secuencia, en el que entraremos en el nodo de Chatting\_action.

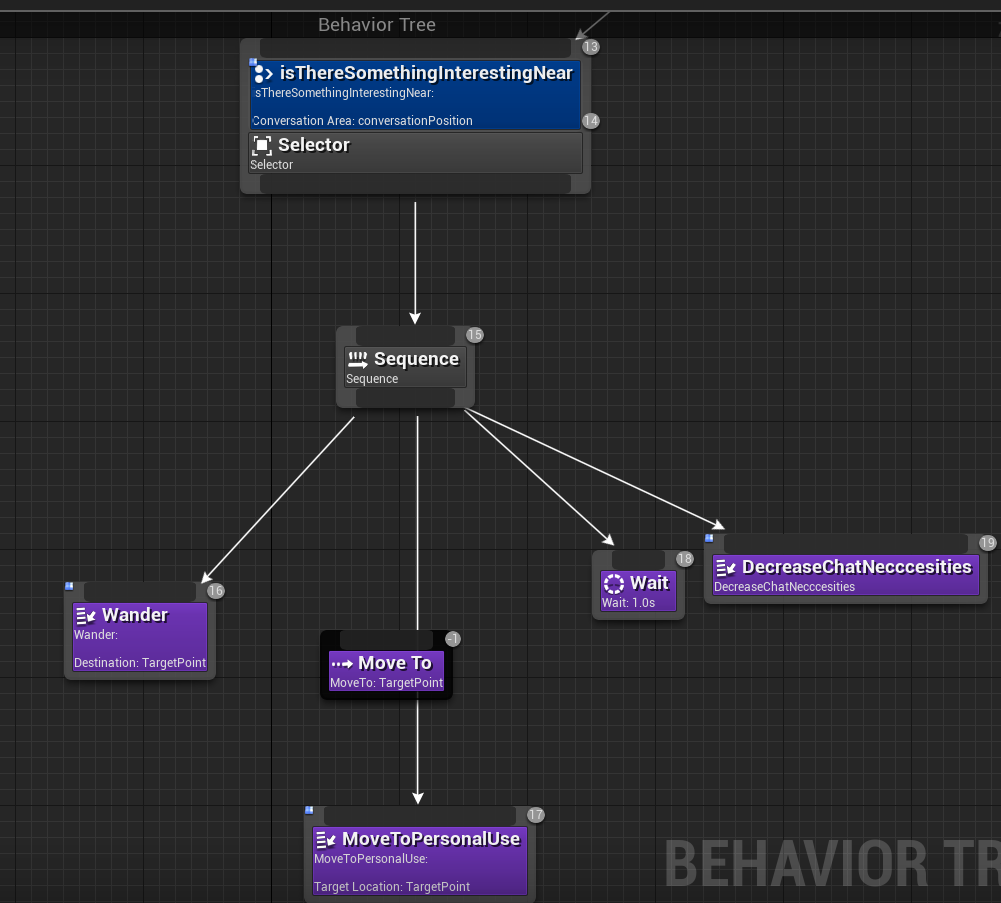


Que comprueba si los jugadores están correctamente ubicados para charlar dentro del área, en caso de no estarlo, se bloquea el movimiento a uno de ellos y al se cambia el valor del chattingPosition del otro.

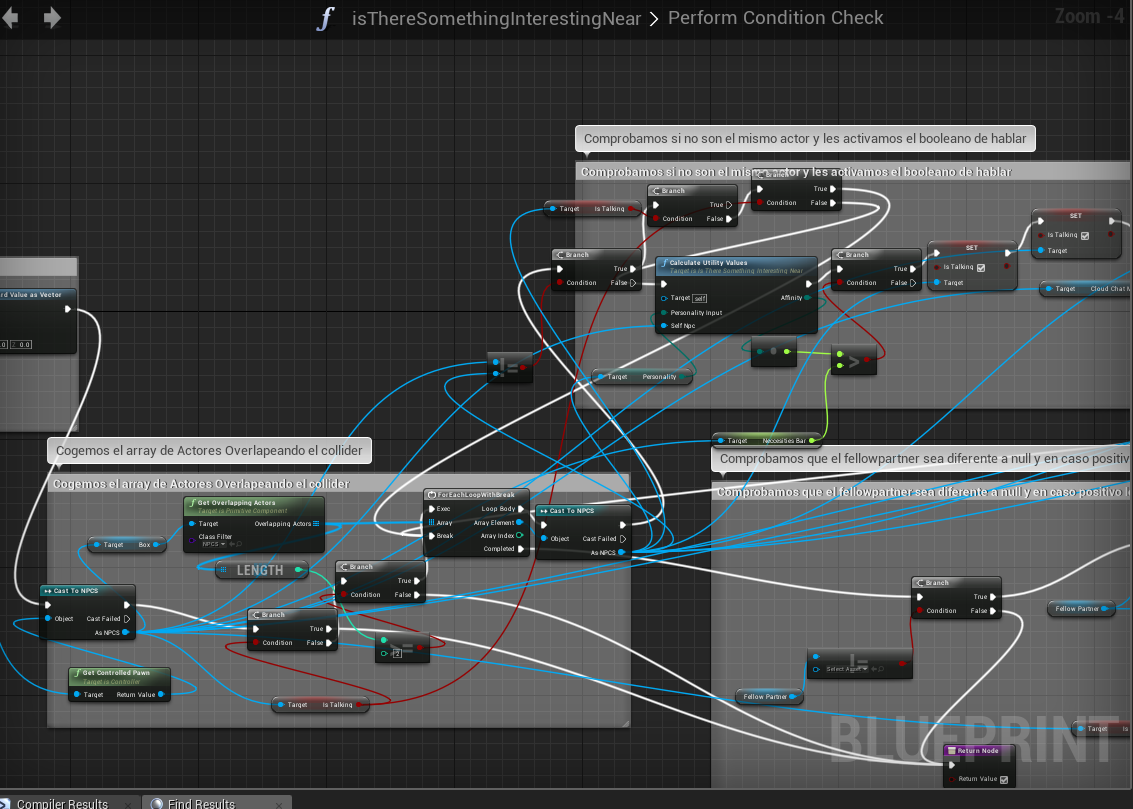
Al terminar en este nodo seguiremos con un nodo selector que nos lleva a un nodo MoveTo con un decorator Is\_conversationPointDifferentToNull que comprueba si el valor chattingPosition es diferente a NULL, de esta manera sólo moveremos a uno de los personajes.

Al acabar, entraremos en un decorator en el que pararemos el movimiento de ambos personajes para activar su animación de hablar. Y finalmente iremos al nodo de aumentar la barra de necesidades, una vez está este llena (se llena por segundos, depende de la personalidad), se devolverá el movimiento a los personajes y se les desbloqueará de la conversación.

Si el nodo de chatting\_action retorna FAILURE, iremos al decorator IsThereSomethingInterestingNear

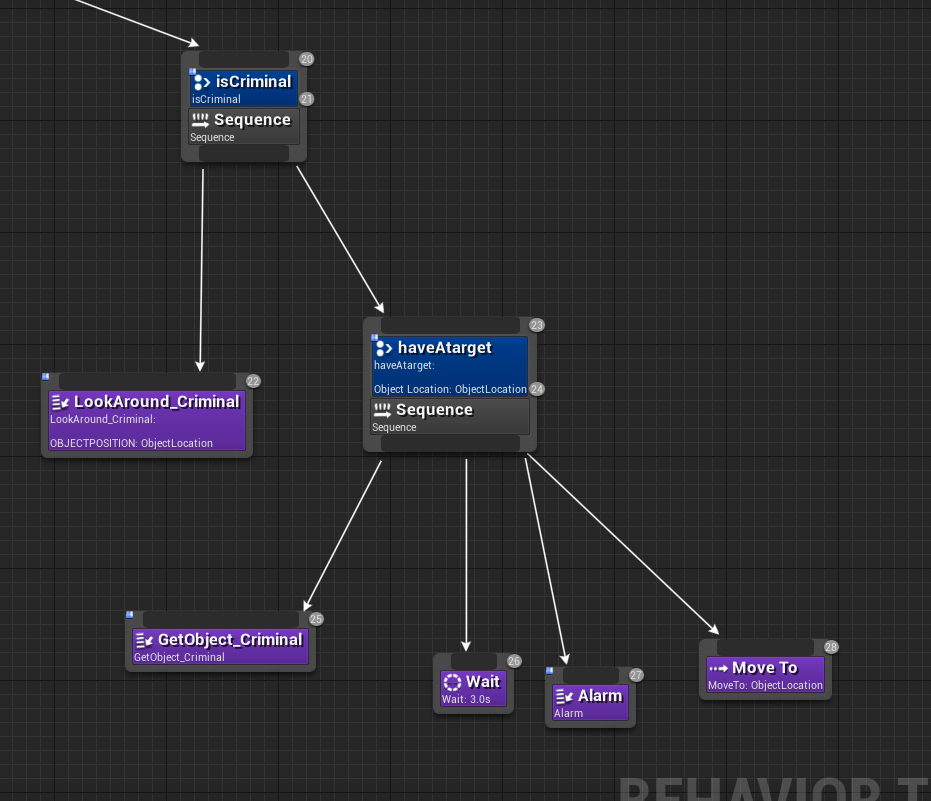


Este nodo se encargará de revisar si dentro del collider de visión del NPC se encuentra algo interesante(actualmente otros NPCs), siempre y cuando su afinidad sea superior a la necesidad en la que se encuentra ese NPC, revisa bilateralmente si ambos están dispuestos a entrar en conversación y sólo lo hacen si ambos cumplen está facultad.



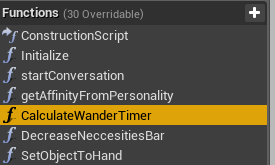
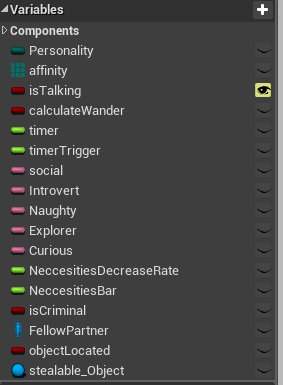
Después de asegurarse de que no se inicia ninguna conversación continúa con el nodo Wander, que se encarga de buscar un punto random(dentro de un radio) dentro del navegable mesh e ir hacía él. Seguido de esto se continúa con el nodo MoveToPersonalUse, es un moveTo hacía un vector3 creado por mí exclusivamente para poder bloquearlo si era necesario, de otra forma, había un problema a la hora de iniciar una conversación, debido a que si uno de los NPCs caminaba hacía un sitio y el otro intentaba iniciar la conversación, conversaban en lugares diferentes lo que quedaba poco realista. Por ello este nodo posee una cualidad de poder retornar Abort en caso de que se encuentre llamado a conversar.

Al terminar este nodo se reducirá la barra de necesidad (o aumentará la barra de aburrimiento según cómo se mire).



Se avanza hasta el último nodo en el que se comprueba si el NPC es el criminal, en caso positivo, se buscará un objeto robable próximo y se colocará en la mano del skeletal mesh del ladrón, dándole 10 segundos al jugador para encontrarlo.

**Variables y funciones**



**Byte Personality** – Implica la personalidad del NPC.

**Byte(Array) affinity** – Los valores de afinidad con cada personalidad.

**Bool isTalking** – Indica si está en una conversación o no.

**Bool calculateWander**- Indica si ha pasado el tiempo necesario para calcular el nuevo wander. Existe un tiempo de espera entre wanders para hacer el movimiento más realista.

**Float Timer –** Ayuda a controlar el tiempo.

**Float timerTrigger –** cada cuanto se resetea el timer.

**String social,Introvert,Naughty,Explorer,Curious;** - Strings guardados para facilitar la limpieza de los nodos.

**Float NeccesitiesDecreaseRate** – Rate por el que bajarán las necesidades

**Float NeccesitiesBar** – Barra de necesidades

**Bool isCriminal** – Booleano que muestra si es criminal o no.

**NPC\* fellowPartner**- Referencia del NPC con el que está interactuando.

**Bool objectLocated** – Booleano para saber si el criminal posee el objeto.

**SteleableObject\* steleable\_Object** – referencia del objeto robable.

**CameraPawn**

Clase encargada de ser el player que observa a los NPCs y tiene la opción de acusarles para pillar al criminal.

**Resumen**

Como cámara de seguridad tendrás que encontrar al ladrón 10 segundos antes de que se consiga llevar la obra de arte. Es más que un juego una fase alpha, debido a que existen algunos bugs aún. Al ser mi proyecto de la asignatura de proyectos pretendo llevarlo más allá y creo que el sistema de personalidades es un buen comienzo, debido a que aunque ahora sólo exista la interacción de la charla, en un futuro las personalidades tendrán sus propias interacciones y maneras de reaccionar a cada cosa, lo que le dará al juego mucha más solidez.