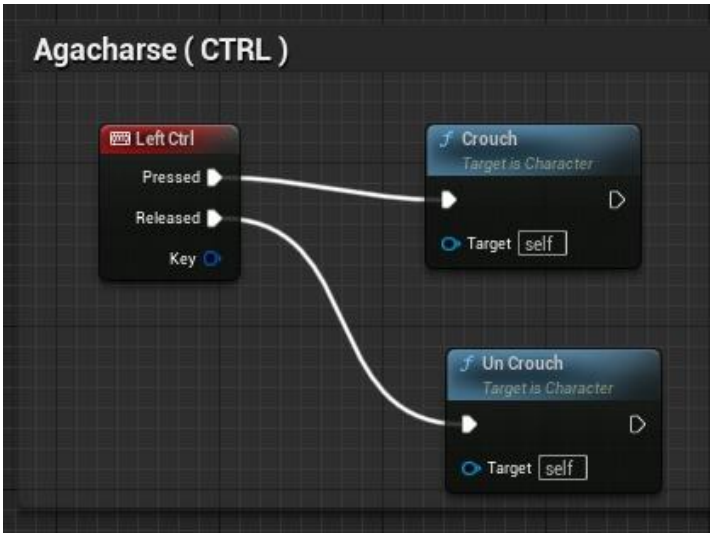
TP — Herramientas y Motores de Trabajo
Adriel Fontanet

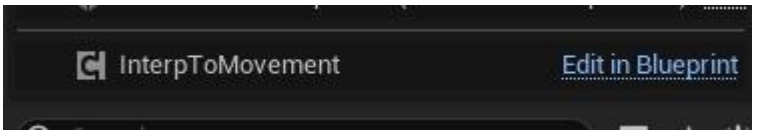
Mecánicas para FPS
1) Doble Salto y Agacharse ✅
2) Tocar un Botón Abre la puerta ✅
3) Plataformas Móviles ✅

- Doble Salto
    - Literalmente ya había una función especializada que me dejaba cambiar la cantidad de saltos que había así que añadí también la posibilidad de Agacharse.

- Agacharse
    - Para agacharse el personaje se utilizan estos blueprints, las funciones de Crouch y Un Crouch ya estaban disponibles por default así que directamente use esos blueprints de Unreal, dentro de las mismas pude configurar la altura que tendría el colider del personaje al estar agachado. Estas funciones las vincule a la tecla Left CTRL y ya estaba funcionando.



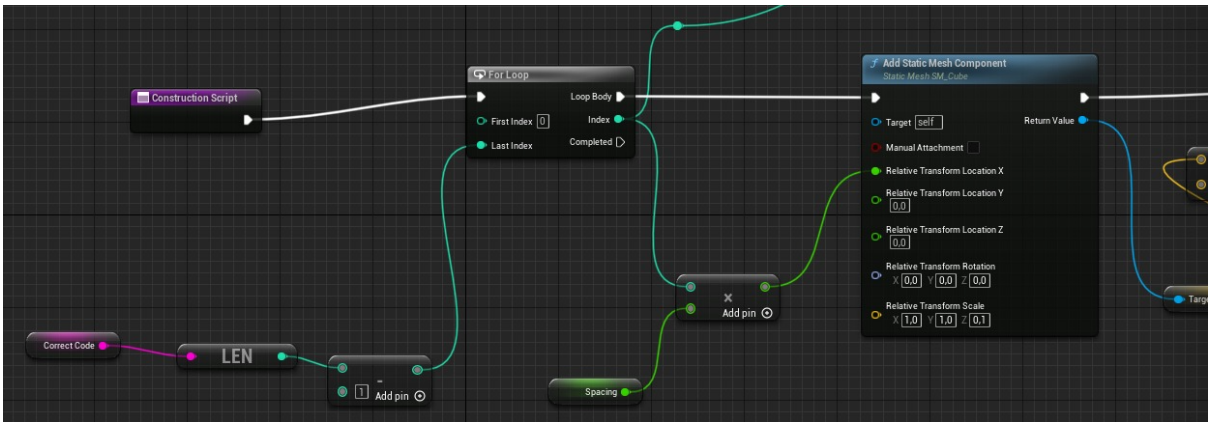
- Plataformas Móviles
    - Generar un movimiento constante con plataformas creadas a través de Blueprints me fue imposible, así que tuve que utilizar un componente que ya tenía Unreal y modificarlo para que sirva mi propósito.
    - El Componente es InterpToMovement nos deja cambiar la posición del actor elegido además nos deja mover el actor elegido por medio de vectores, lo cual lo hace un componente muy cómodo de usar.



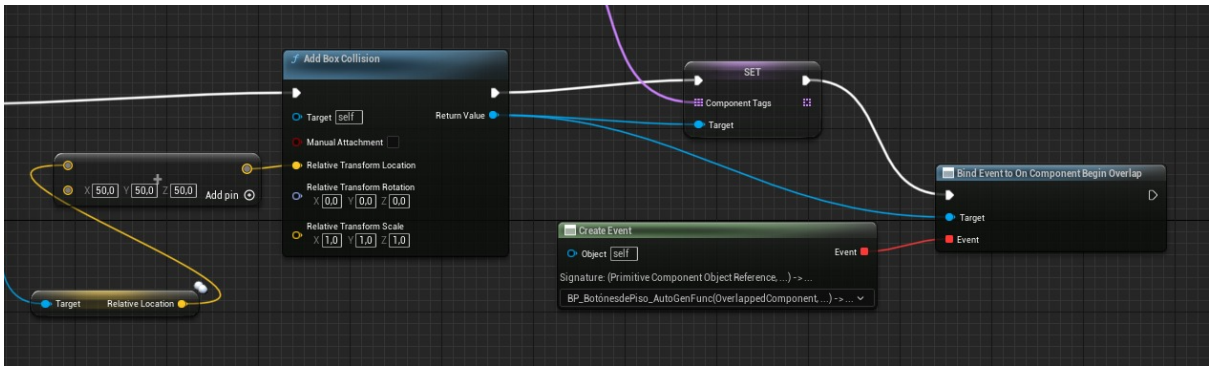
    - Al usar este componente, si lo que queremos es que la acción se loopee infinitamente, hay que poner la opción “Ping Pong” que hace que la primera acción de movimiento se vuelva a generar pero al revés y luego loopee estas dos acciones.

- Tocar un Botón abre la puerta
    - Ojala esta mecánica, hubiera sido tan simple como las demás pero tuve que crear dos funciones distintas simplemente para poder hacer esto de forma rápida y no tener que copiar los blueprints una y otra vez
    - Primero, hice con blueprints la morfología de las placas de piso para poder crear varías al mismo tiempo y para esto necesite crear varios componentes que significan varias cosas
    - Spacing: Espaciamiento que le dice a los blueprints cuanta distancia tiene que haber entre plataforma y plataforma (Si no estuviera esto, las tres plataformas estarían una arriba de otra)

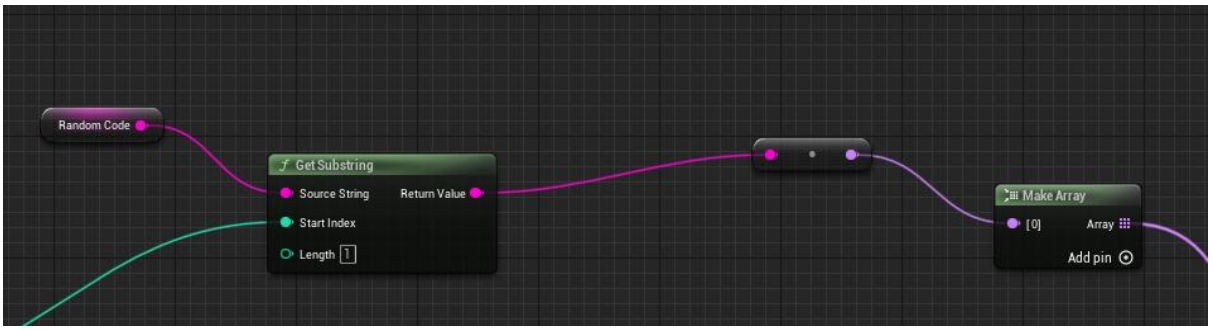
    - CorrectCode: Componente que es el que ayuda a que automáticamente aparezcan más plataformas al alargar el código, con este código, RandomCode randomiza el mismo.
    - InputCode: El código que se pone por el jugador, esto nos ayuda a resetear el puzzle cuando se hace mal y también a que se valide que se hizo correctamente el puzzle
    - RandomCode: Componente que genera un código Aleatorio (Por problemas con el feedback al jugador, tuve que Hardcodear un código específico que es primero el medio, segundo el de la izquierda y tercero el de la derecha)
    - Door: Cree el componente Door, para que la función pueda entender que Door es un actor y la deje en modificable, para que pueda elegir que actor era catalogado como Door en el editor en vez de en la pantalla de Blueprints



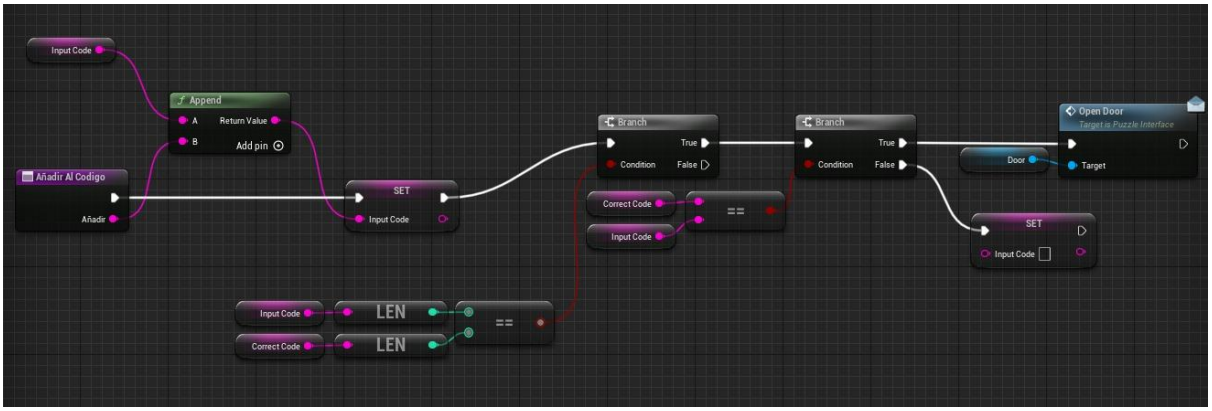
    - Acá podemos apreciar que gracias al Blueprint For Loop, multiplicando el Spacing y al CorrectCode (Además el Correct Code en vez de estar conectado directamente, esta anexado a un -1 porque sino el For Loop hace una plataforma demás) crea Static Meshes con la posición que le decimos y la forma que le explicamos (En este caso una placa fina)



- A los Static Meshes les añadimos unos Box Collisions que son más grandes que las placas, así podemos hacer Overlap con el personaje jugable y así activar un evento
- A esto le sumamos el SET que va a estar conectado a El Get Substring y el Make Array (Para que se creen automáticamente las placas con sus Box Collisions que queremos según el RandomCode, acá podría también ir CorrectCode, pero pongo RandomCode por si me equivoque al hardcodearlo directamente no funcione nada y se exactamente cual es el problema)



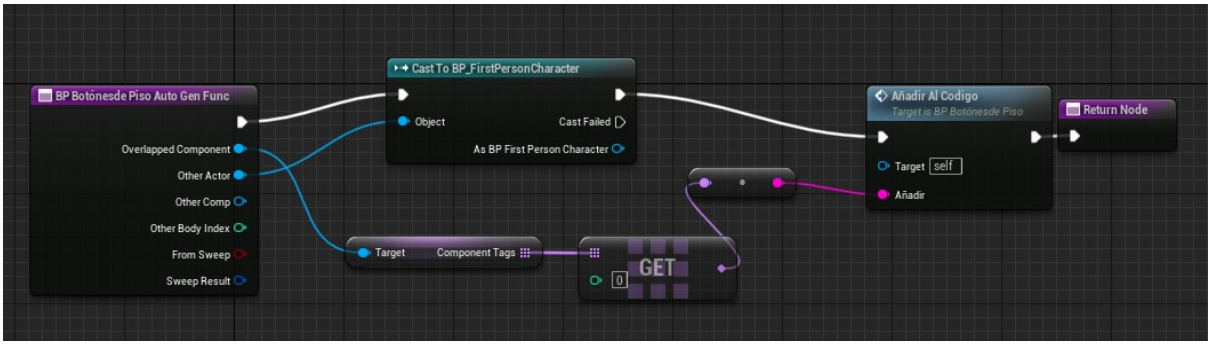
- El Evento de overlap está vinculado a una funcion llamada AÑADIR AL CODIGO



- Añadir al código utiliza varios Blueprints y componentes ya explicados para generar la acción de que el código que inputea el jugador, sea el mismo que el que hardcodie anteriormente.
- Utilizo un branch, que como condición tiene que inputCode (el código ingresado por el jugador) y Correct Code (El código correcto) sean el mismo.

Si son el mismo, se crea otro branch que vuelve a verificar lo mismo, solo que esta vez si no se cumple la condición, se resetea el código. Para que si el jugador se equivoca al inputear las tres placas, este código se resetea. Si no existiese esto al equivocarse una vez sola, habría que resetear completamente el nivel.

- Si este branch da True, se activa la función OPEN DOOR y el target es DOOR, esto nos deja seleccionar un actor dentro del editor, para que este sea Door y nos deje cambiarlo de lugar.



- Para qué AÑADIR AL CÓDIGO entienda que el OVERLAP tiene que crearse a partir del evento real de que un jugador de primera persona se pare arriba del Collider Box de los Botones de piso, necesitamos estos blueprints, que literalmente hacen que Botónes de piso entiendan que es evento debe activarse cuando FirstPersonCharacter (El actor pawn que utiliza el jugador) está encima de las placas.