

EQS Farming



Ignacio Tapia Marfil

Diseño y desarrollo de videojuegos y entornos virtuales.

29/01/2025

Índice.-

| | |
|------------------------|----|
| Índice.- | 1 |
| Descripción General.- | 2 |
| Descripción Técnica.- | 3 |
| RECURSOS:- | 3 |
| SPAWNER:- | 5 |
| ALMACENES:- | 7 |
| RECOLECTORES:- | 9 |
| EXTRAS:- | 17 |
| Diario de Desarrollo.- | 19 |
| Bibliografía.- | 20 |

Ignacio Tapia Marfil

Diseño y Desarrollo de Videojuegos
Tecnologías de Desarrollo de Videojuegos y Entornos Virtuales III



Descripción General.-

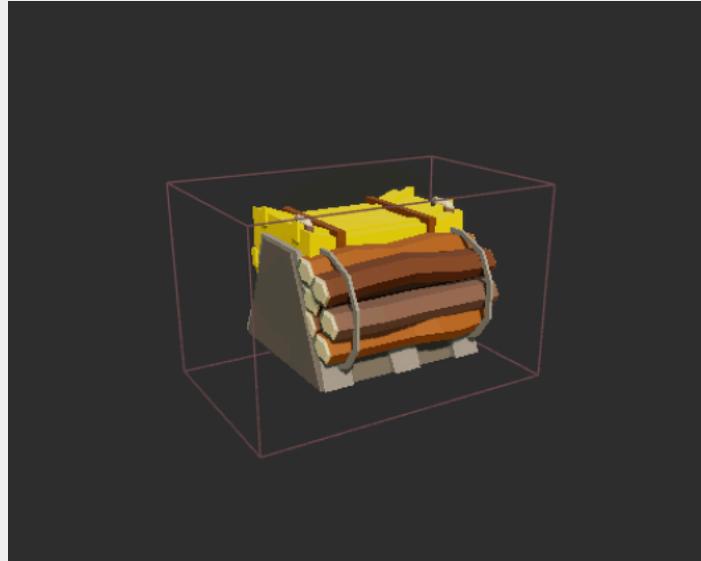
Este proyecto ha sido realizado para la asignatura de Tecnologías de Desarrollo de Videojuegos y Entornos Virtuales III. Trata de realizar una IA que recolecte recursos y los deposite en un almacén según especificaciones que le damos. Para esto, vamos a usar las EQS de Unreal Engine, junto a los Behaviours Trees y un IA controller.

PUNTOS PRINCIPALES:

- **Recursos:** diferenciándolos en tres tipos, en este caso, madera, alimento y piedra.
- **Spawner:** se encargan de generar los recursos. El tiempo y tipo de recurso son editables (si se genera más de un tipo de recurso en un spawner, habrá que elegir el porcentaje de aparición que tendrá cada uno de los recursos).
- **Almacenes:** se depositarán los recursos colocándose en el espacio en una matriz de 3 dimensiones. El tipo de recurso que se guarda en el almacén es editable, pudiendo guardar más de un tipo de recurso en cada uno.
- **Recolectores:** irán a recoger el recurso más cercano a ellos del tipo que tienen asignado. Cuando tengan el recurso, irán a depositarlo al almacén más cercano donde se pueda poner dicho recurso. El tipo de recurso se puede cambiar, pudiendo poner los tres tipos de recurso.

Descripción Técnica.-

RECURSOS:



Todos los recursos son el mismo blueprint, que cambian su tipo en función de un enumerador creado.

| Description | | Resources Type |
|------------------|-------|----------------|
| Enum Description | | |
| ▼ Enumerators | | |
| Display Name | Stone | |
| Display Name | Wood | |
| Display Name | Food | |

Según este enumerador, la Mesh del recurso cambia.



Ignacio Tapia Marfil

Diseño y Desarrollo de Videojuegos
Tecnologías de Desarrollo de Videojuegos y Entornos Virtuales III

UC3M

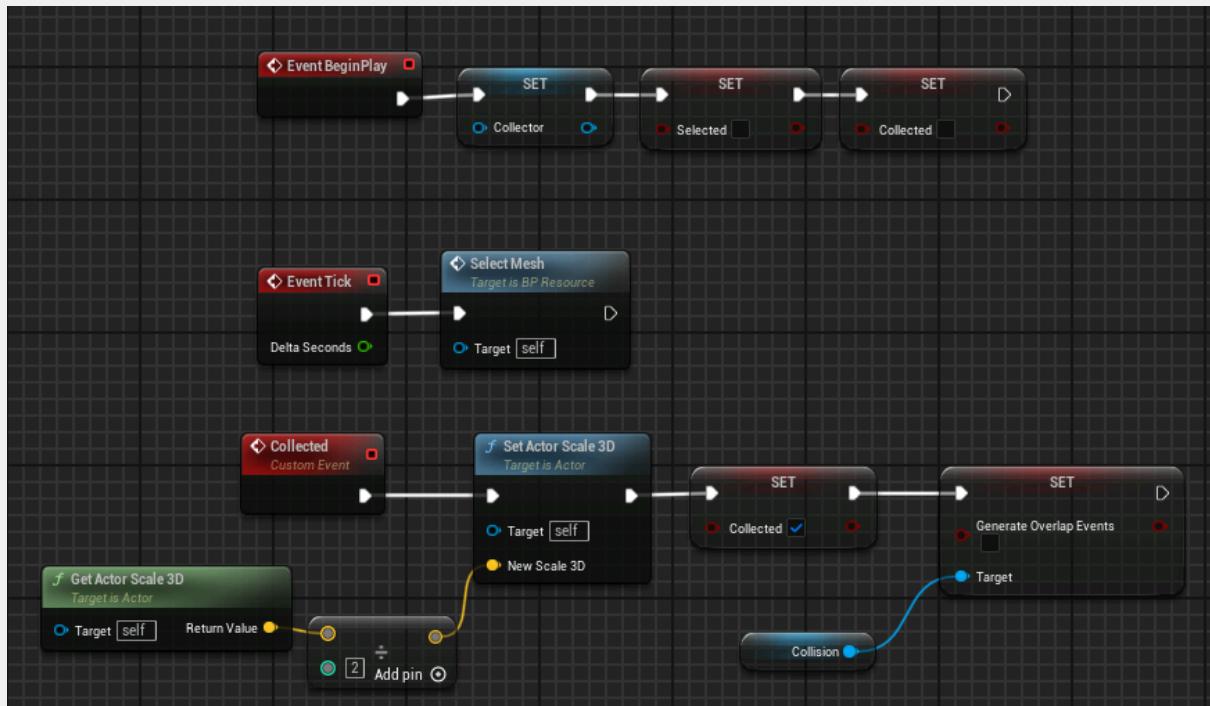
El blueprint también consta de dos variables Bool y una del tipo del actor del recolector.

Las dos Bool son:

- Selected, se marca cuando un recolector la marca como objetivo.
- Collected, se marca cuando un recolector la recoge.

La variable del tipo del actor del recolector marca el recolector que irá a recoger ese objeto.

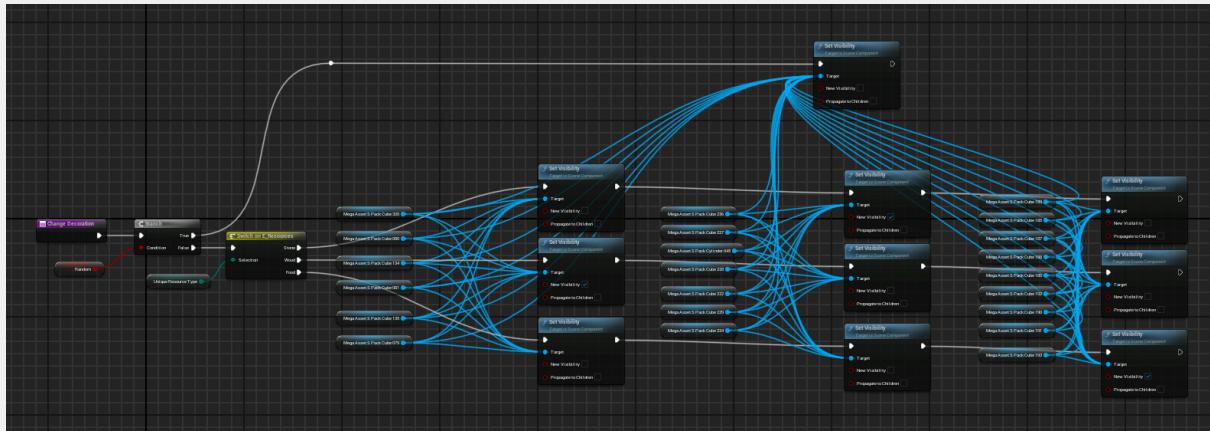
Cuando se recoge el objeto, se llama a un evento que reduce el tamaño del objeto y lo marca como recogido.



SPAWNER:

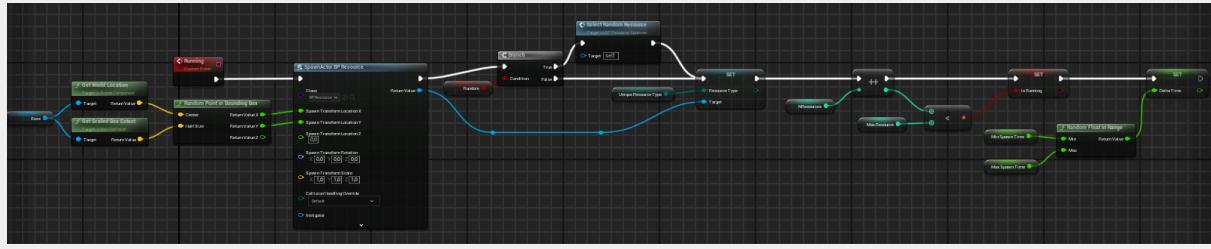


Los cuatro tipos de spawner (madera, alimento, piedra y varios) son un mismo blueprint, que según el tipo de recurso que generen tendrán un aspecto u otro. Si generan más de un tipo de recurso no tendrán ningún tipo de decoración.

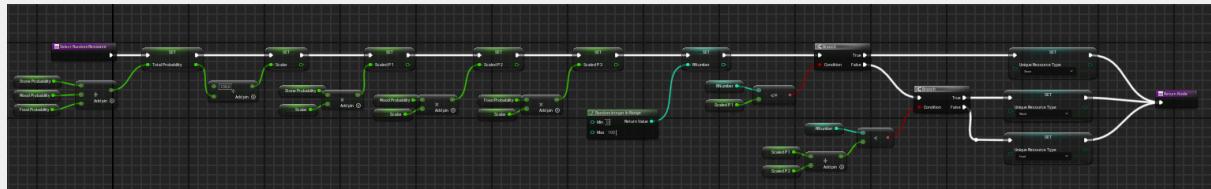


Los recursos se generarán en intervalos aleatorios, entre un tiempo máximo y un mínimo que podremos ajustar.

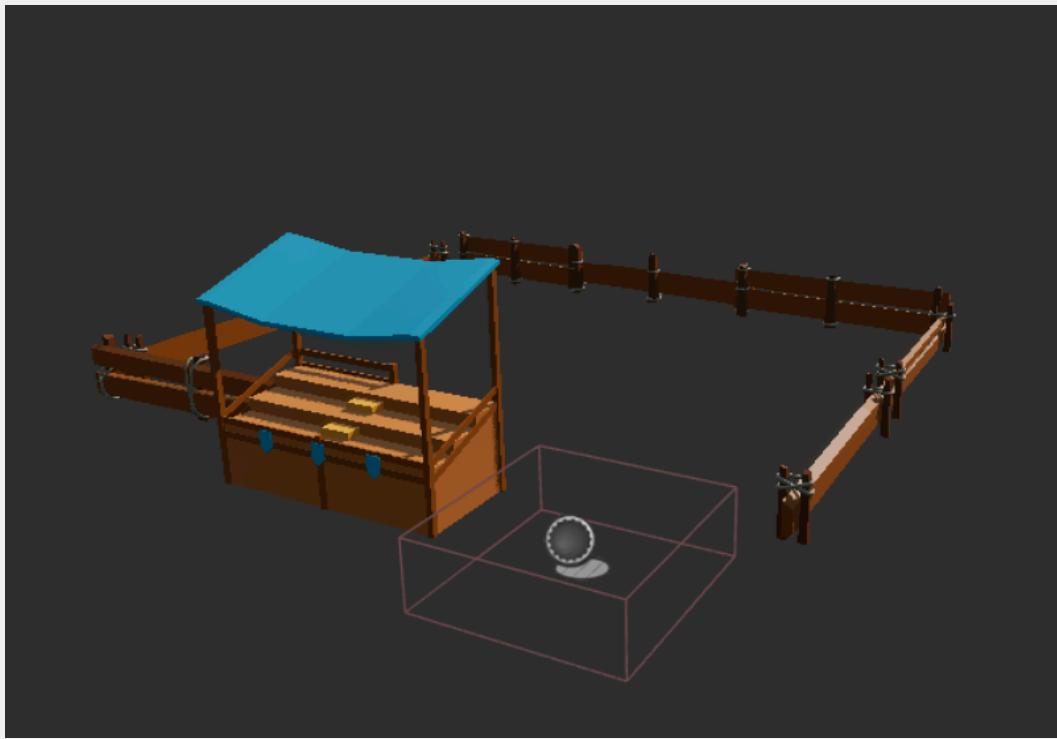
Cada spawner tiene un máximo de recursos que puede generar.



Si el spawner genera más de un tipo de recurso, se pondrá el porcentaje de aparición de cada uno de los recursos que queramos que aparezcan, y cada vez que se genere uno se seleccionará un tipo siguiendo dicho porcentaje.



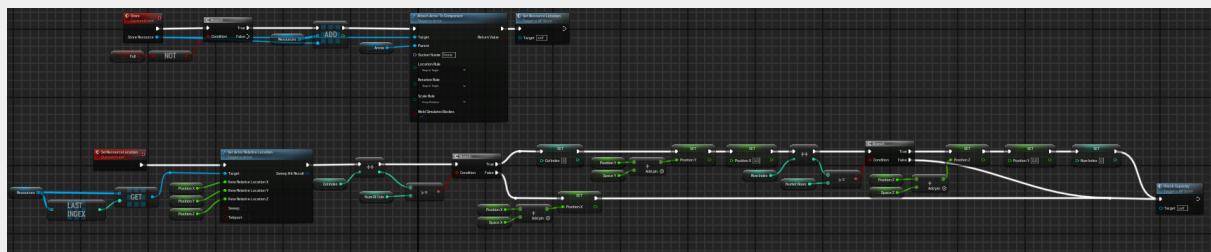
ALMACENES:



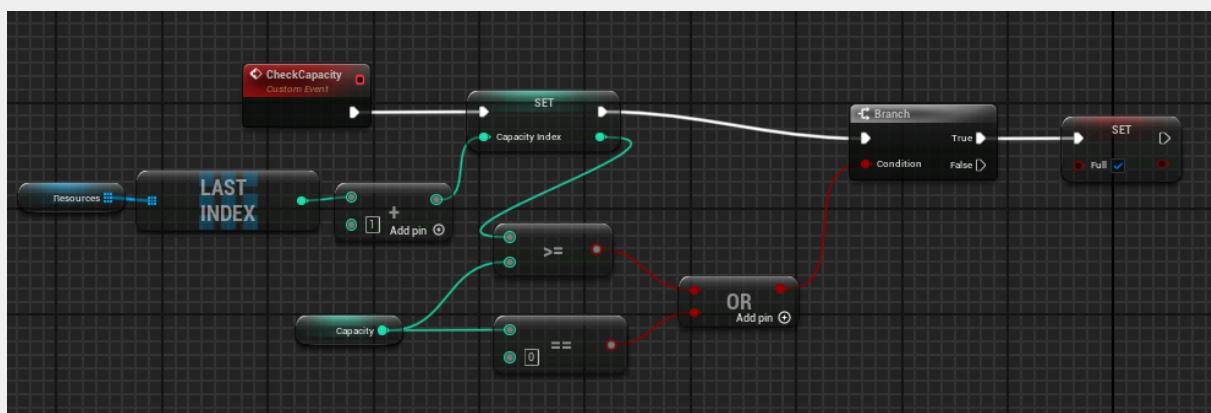
Los almacenes son iguales independientemente del recurso que se pueda depositar en ellos. Los recursos que se pueden dejar en cada almacén se elegirán con un array del enumerador de los recursos.

| Resource Type | 3 Array elements | ⊕ | ⊖ | ↶ |
|---------------|------------------|---|---|---|
| Index [0] | Stone | ▼ | ▼ | ↶ |
| Index [1] | Wood | ▼ | ▼ | ↶ |
| Index [2] | Food | ▼ | ▼ | ↶ |
| Resources | 0 Array element | | | ⊕ |

Cuando se deposita un recurso en el almacén este se attacha a un componente dentro del almacén, y se coloca en su posición correspondiente, según cuantos elementos haya en el almacén.



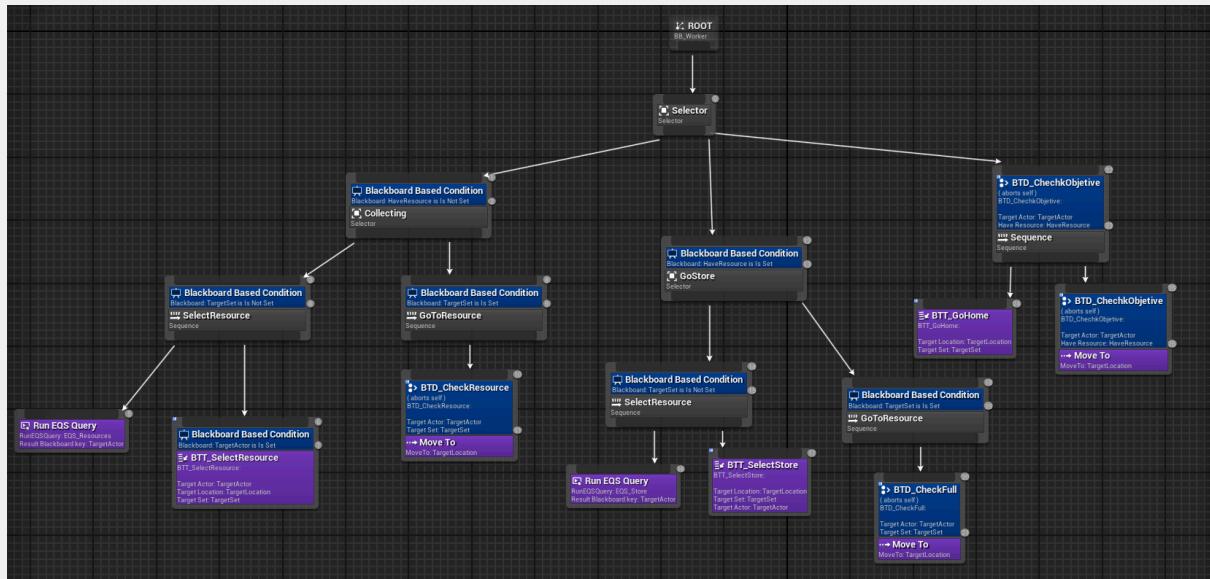
Después de colocarlo se revisa si el almacén ha llegado a su máxima capacidad (variable editable), y si ha llegado a esta capacidad se bloqueará el almacén para que no se puedan depositar más objetos.



RECOLECTORES:



Los recolectores se manejan a través de un behaviour tree, que es el que controla toda la lógica de sus acciones.



Ignacio Tapia Marfil

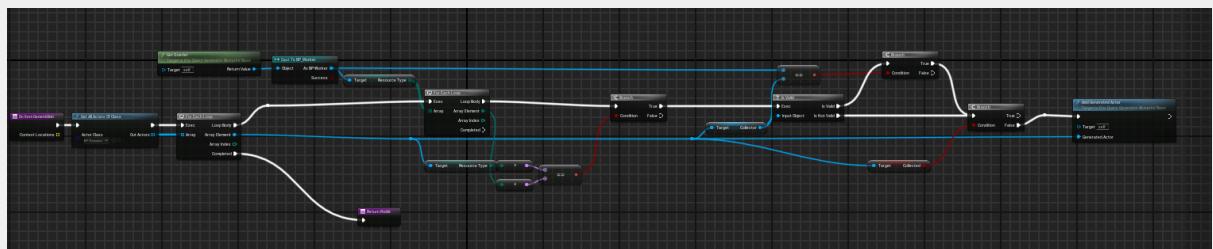
Diseño y Desarrollo de Videojuegos
Tecnologías de Desarrollo de Videojuegos y Entornos Virtuales III

UDC

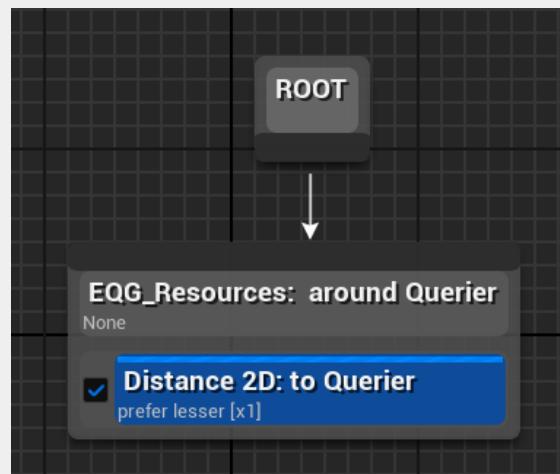
El árbol se separa en tres ramas principales. Si no tiene algún recurso encima, si sí lo tiene y una tercera rama, que mira si hay algún recurso o almacén disponible para ir, tenga o no recurso encima.

1. La primera rama es aquella en la que todavía no tiene el recurso, se divide en dos ramas. Una en la que no tiene todavía el recurso objetivo seleccionado, y en la que ya lo tiene seleccionado.
 - 1.1. No objetivo: primero inicia la EQS Query, que se encarga de encontrar el recurso más cercano del tipo de recurso asignado a este recolector.

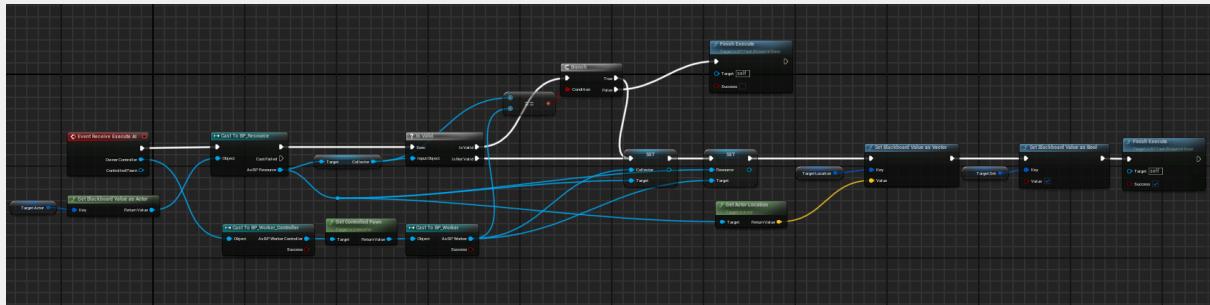
Esta Query genera puntos de detección mediante un generador personalizado, que detecta todos los recursos del entorno del tipo de recursos asignados a ese recolector.



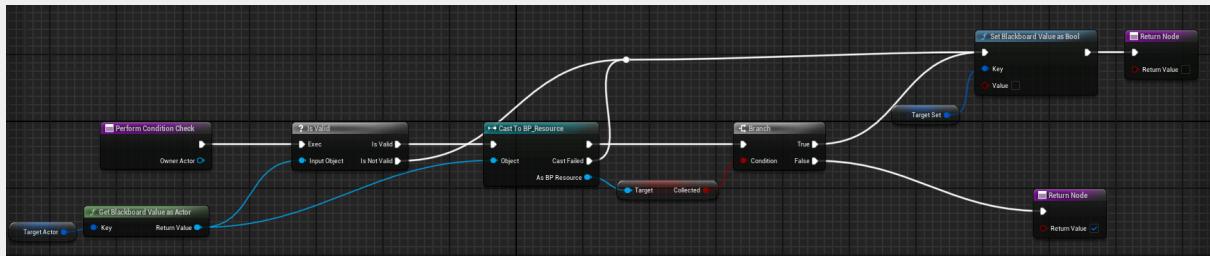
Después de esto, devuelve el más cercano al propietario de esa Query.



Una vez tengamos este recurso, llamamos a una tarea que va a setear todas las variables necesarias dentro del recolector, el blackboard y el recurso seleccionado. Antes de colocar estas variables, se comprueba que el recurso no tenga ya un recolector asignado, si es así, devuelve una ejecución errónea y se volvería a lanzar el árbol.

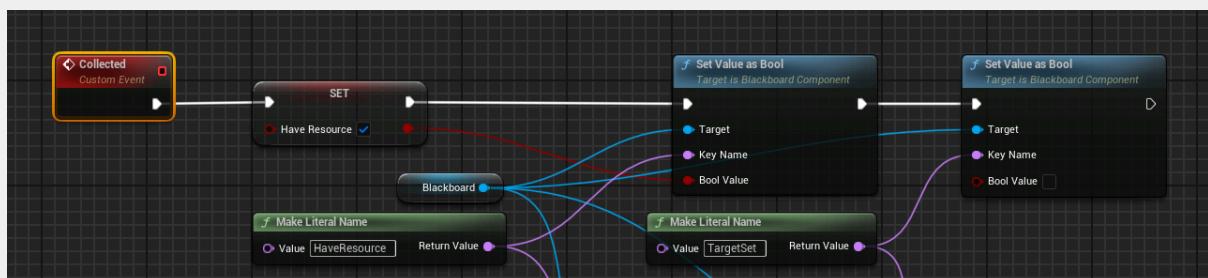
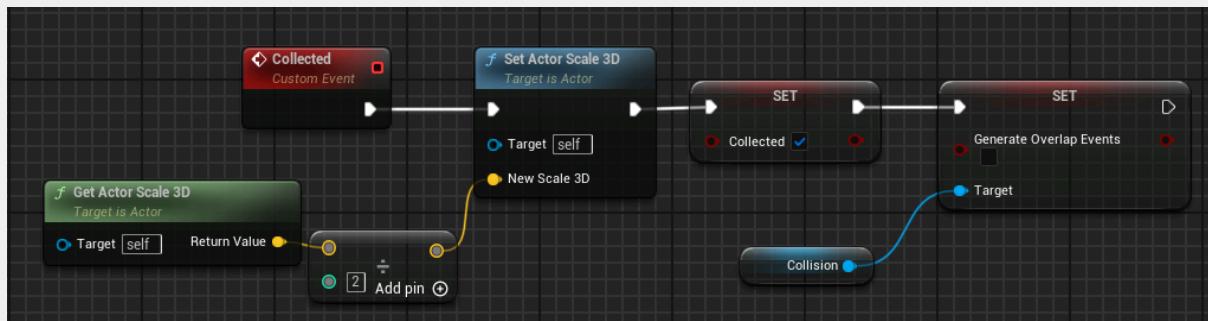
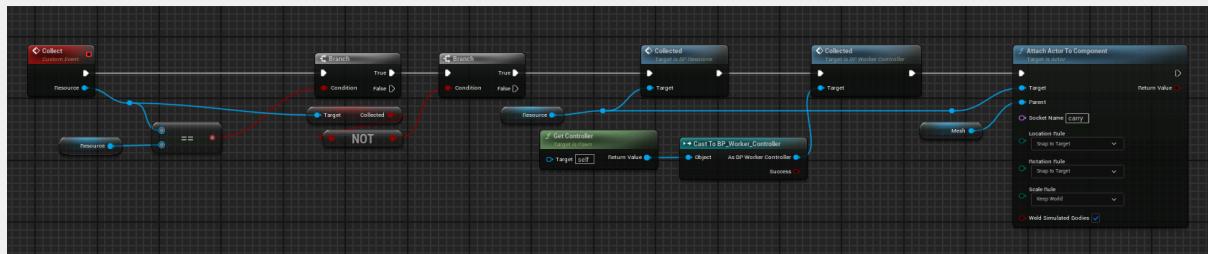
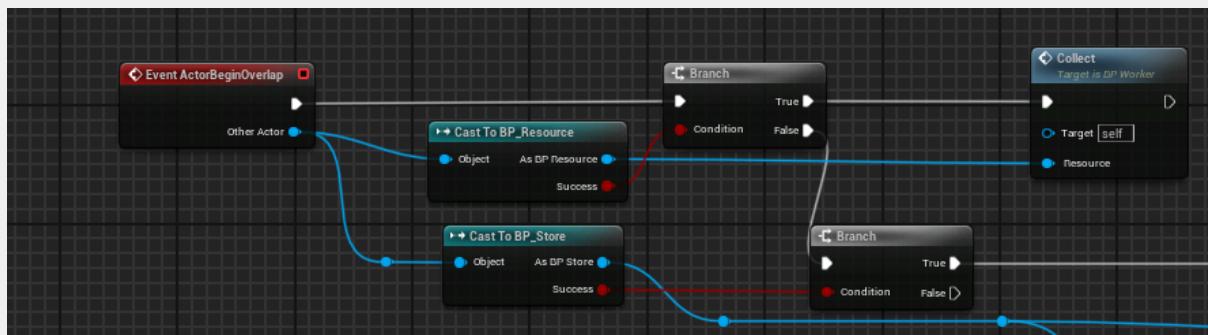


- 1.2. Si objetivo: llama a un move to para que el recolector se mueva hacia el recurso . Este nodo tiene un decorador que va a estar revisando todo el rato que el recurso al que está yendo no haya sido recogido antes de que este recolector llegue. También comprueba si el recurso ha desaparecido o da algún tipo de error, de ser así, devuelve una ejecución errónea y se volvería a lanzar el árbol.



Una vez llega al recurso que tiene seleccionado como objetivo, el recolector lo recoge. Se marca el recurso como recogido y se llaman a las acciones de recoger del recurso y del controller del recolector. Después, el recurso se attacha al recolector en un socket determinado.

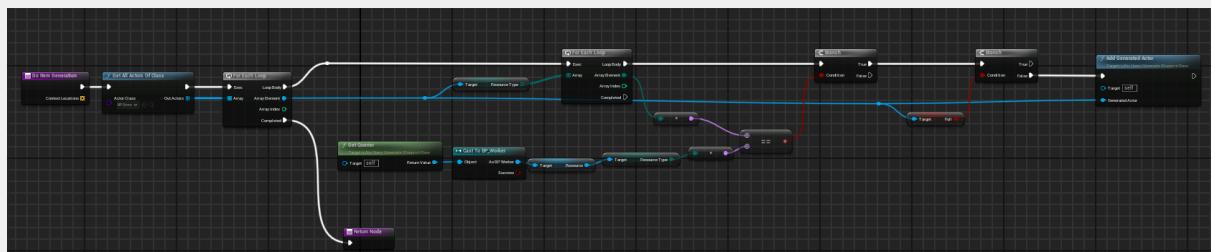
En el collected del controller se cambian las variables del blackboard, para que este pueda seguir con su ejecución.



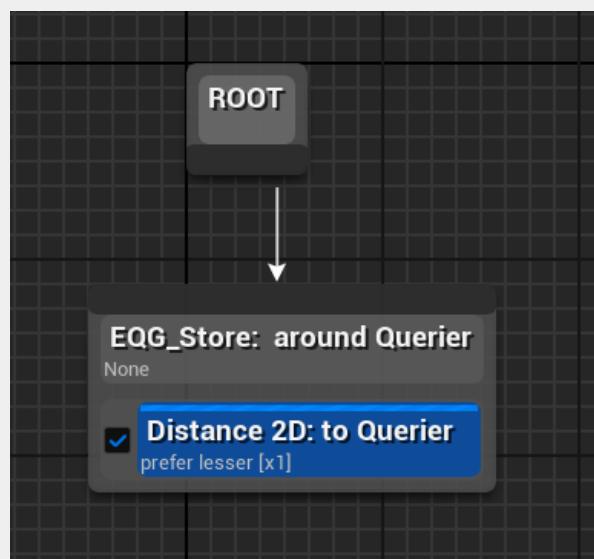
2. La segunda rama, se activa cuando el recolector ya tiene un recurso. Esta también se divide en otras dos ramas, que se separan en sí tiene un objetivo seleccionado o no, pero en vez de con un recurso, con un almacén.

2.1. No objetivo: primero inicia la EQS Query, que se encarga de encontrar el almacén más cercano que acepta el tipo de recurso que tiene encima.

Esta Query genera puntos de detección mediante un generador personalizado, que detecta todos los almacenes del entorno que aceptan el tipo de recurso que tiene el encima el recolector.

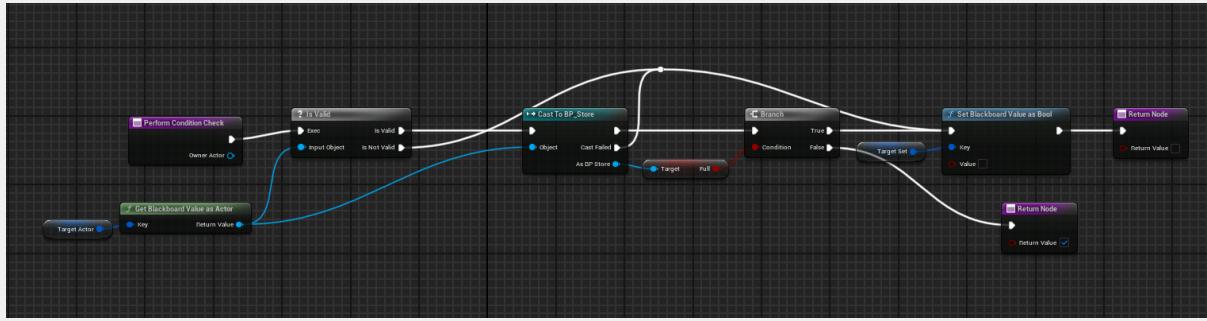


Después de esto, devuelve el más cercano al propietario de esa Query.



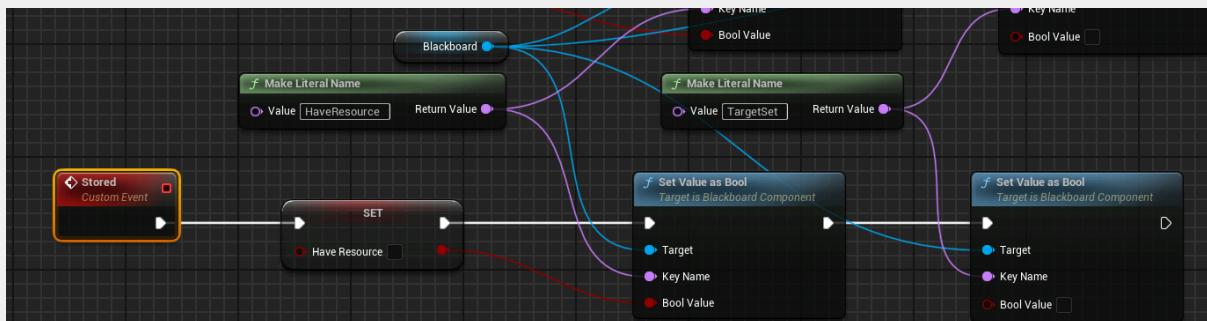
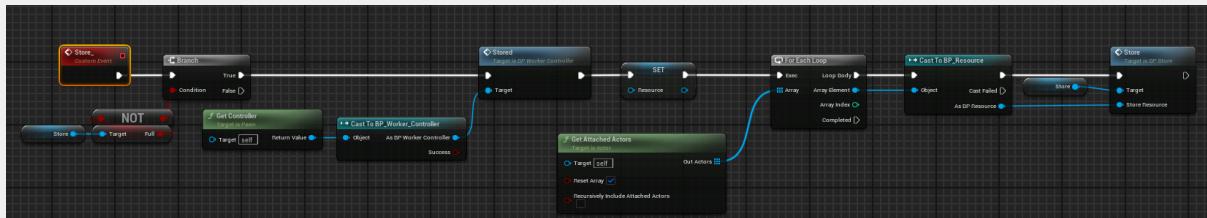
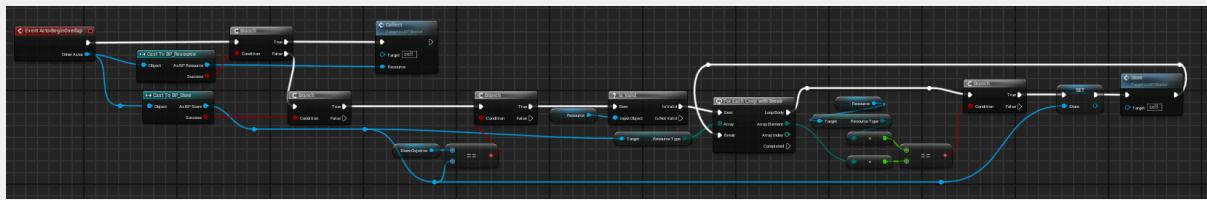
Una vez tengamos este almacén, llamamos a una task, que va a setear las variables, tanto del actor del recolector como del blackboard. Antes de esto, se comprueba si el almacén que ha seleccionado está lleno o no. De estar lleno, se devuelve una ejecución errónea y se vuelve a lanzar el árbol.

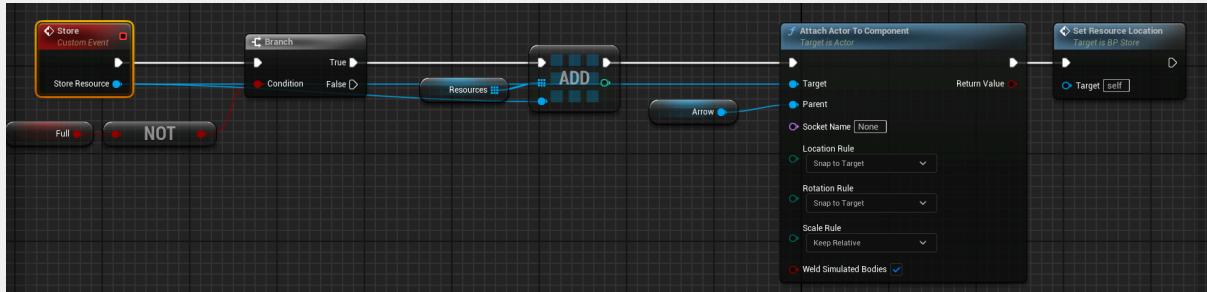
- 2.2. Si objetivo: llama a un move to para que el recolector se mueva hacia el almacén. Este nodo tiene un decorador que va a estar revisando todo el rato que el almacén al que está yendo no se haya llenado antes de que este recolector llegue. También comprueba si el almacén ha desaparecido o da algún tipo de error, de ser así, devuelve una ejecución errónea y se volvería a lanzar el árbol.



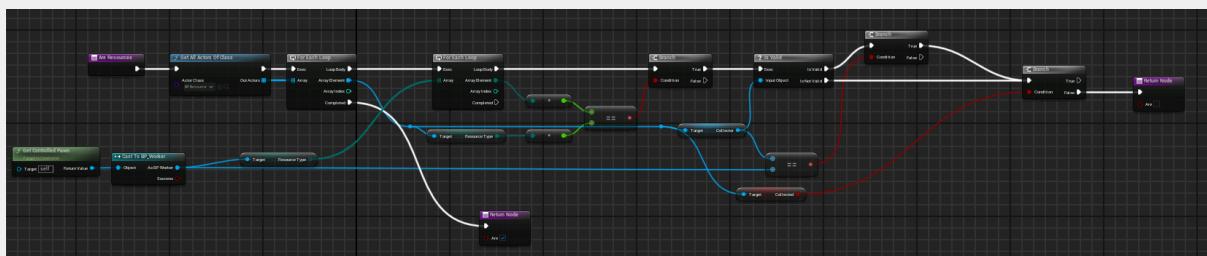
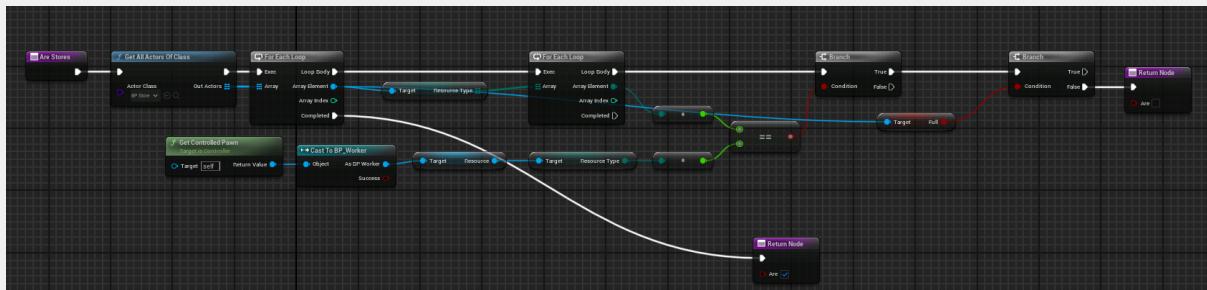
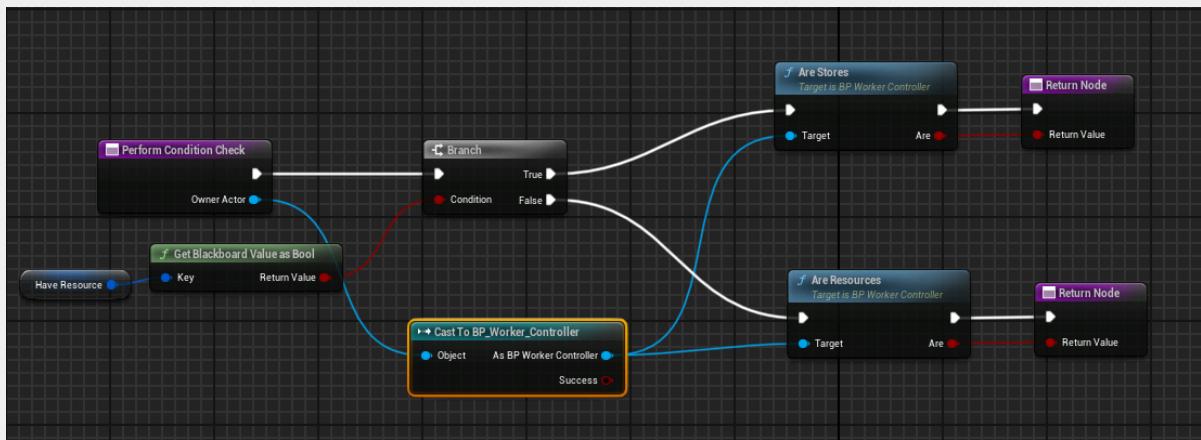
Una vez llega al almacén que tiene seleccionado como objetivo, el recolector deposita el recurso. Se marca el recurso como recogido y se llama a las acciones de recoger del controller del recolector y del recurso. Después, el recurso se attacha al almacén.

En el collected del controller se cambian las variables del blackboard, para que este pueda seguir con su ejecución.





3. La tercera rama cuenta con un decorador que primero comprueba si ya tiene recurso encima o no. Después, llama a unas funciones del controller, que comprueban que si no tienes recurso exista alguno disponible para ir a por él, y si ya tienes recurso, exista algún almacén para ir a dejarlo.



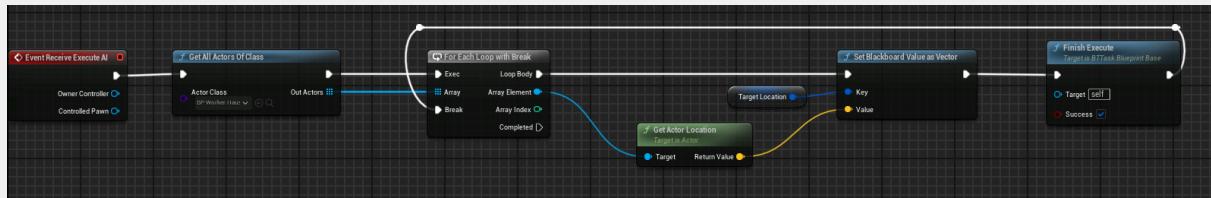
De no haber almacén o recurso disponible, este decorador deja paso y se deja paso a dos nodos.

Ignacio Tapia Marfil

Diseño y Desarrollo de Videojuegos
Tecnologías de Desarrollo de Videojuegos y Entornos Virtuales III



- 3.1. Un task que busca el punto de descanso y pasa su localización a las variables del blackboard.



- 3.2. Un nodo move to, que con un decorador, el mismo que el de la secuencia padre, para que, si mientras se mueve a la zona de descanso hay disponible algún recurso o almacén, se relance el árbol.

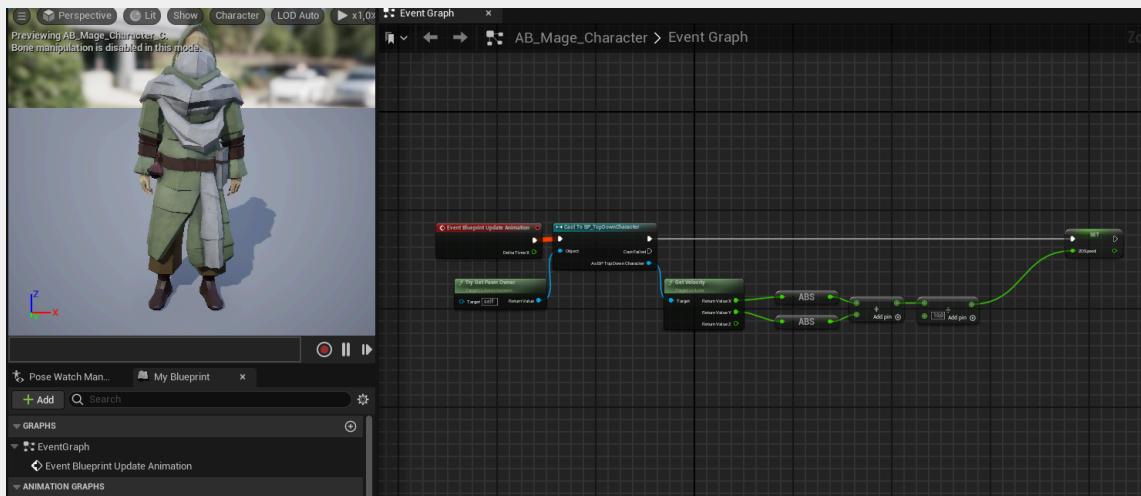
EXTRAS:

Los apartados que se han añadido a mayores a este proyecto son:

- Paquete de Assets de decoración medieval, tanto para el entorno como para los spawners, recursos y almacenes.



- Character mesh personalizado, sacado del fab de UE para los recolectores y el player. Así como animation blueprints, tanto para los AI characters como para el player hechos por mi, con animaciones de mixamo.



- Top down controller de los niveles básicos de unreal, al que le he añadido la mesh y las animaciones personalizadas, además, de la capacidad de recolectar recursos de la misma forma que los recolectores, pero pudiendo recoger cualquier tipo de recurso.

Ignacio Tapia Marfil

Diseño y Desarrollo de Videojuegos
Tecnologías de Desarrollo de Videojuegos y Entornos Virtuales III

UDC



- Punto de descanso, representado con una casa, al que los recolectores van cuando no les es posible hacer otra cosa, es decir, o no tienen recurso y no hay ninguno en el nivel que puedan recoger, o tienen recurso y no hay almacén disponible para poder dejarlo.



Diario de Desarrollo.-

Este proyecto realizado en UnrealEngine 5 no ha supuesto un reto muy grande a pesar de que era la primera vez que utilizaba las herramientas de las query.

El principal reto al que me he enfrentado con este trabajo es a entender de 0 el funcionamiento de estas query y como poder modelarlas a mi gusto. Al final he conseguido tanto saber crear detectores de objetos específicos, como detectores a ciertas distancias y otras muchas cosas.

También decidí ampliar mi conocimiento en cuanto al funcionamiento de las IA de Unreal Engine ahondando más en la parte del behavior tree y el uso de task y decorators personalizados para realizar tareas.

También y debido a que este proyecto lo realicé en navidad, donde disponía de más tiempo, decidí dedicarle más horas y sobre todo centrarme mucho en corregir todos los errores de comportamiento posibles que tuviera la IA, realizando muchas pruebas y arreglando todos los pequeños errores que me iba encontrando. Además de esto también decidí añadir un poco de ambientación así como algún apartado extra que creo que le añaden profundidad al proyecto.

En resumen, este proyecto aunque no ha sido uno de los que más esfuerzo me ha llevado si que me ha ayudado a entender mucho mejor algunas herramientas que tiene este motor además de darme pie a aprender más sobre el pulido de mecánicas y comportamiento de IA.

Bibliografía.-

[Environment Query System in Unreal Engine](#)

[Behavior Tree in Unreal Engine - Quick Start Guide](#)

[Fab](#)

[Mixamo](#)

[Chat Blackbox](#)

[ChatGPT](#)