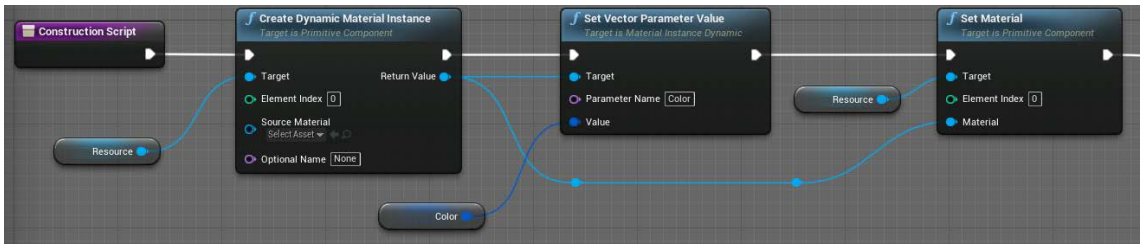
Índice

Recursos

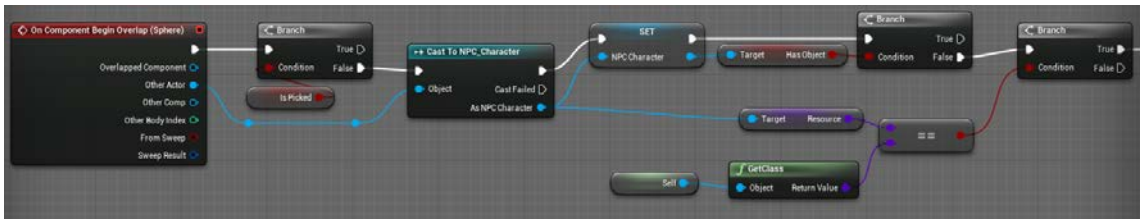
Hay tres tipos de recursos: verde, naranja y azul.

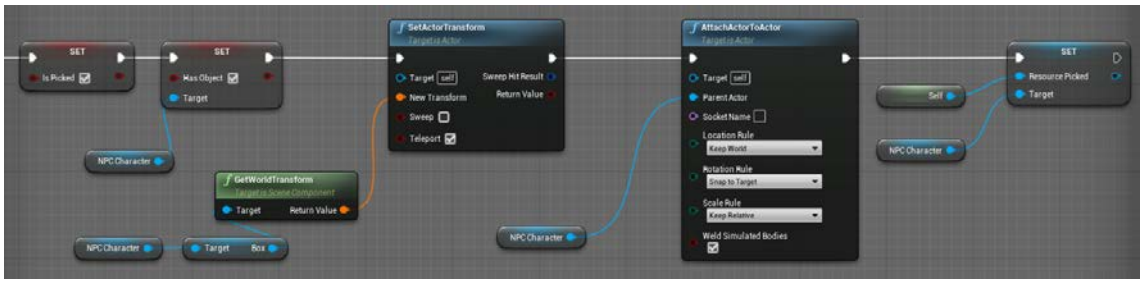
Hay una clase base llamada “ResourcesBase” de la que heredan el resto. Tiene varias variables: el color, el nombre y un bool para comprobar si un NPC lo ha cogido.

El color del recurso cambia cuando se cambia la propiedad del color.



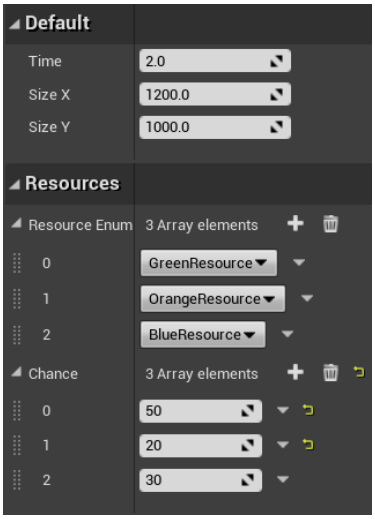
Los recursos tienen una esfera alrededor, y si el NPC entra en ella se activará el código de recoger. Primero comprueba que el tipo de recurso que puede recoger el NPC que ha pasado es del mismo tipo. Si es el caso, y el NPC no está llevando ningún objeto, el recurso se colocará en la espalda del NPC (se convierte en un hijo del NPC para que así se mueva con él) y el bool que determina si alguien lo ha cogido cambiará para que no lo busquen otros.



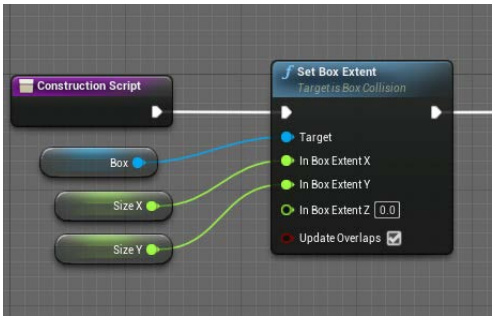


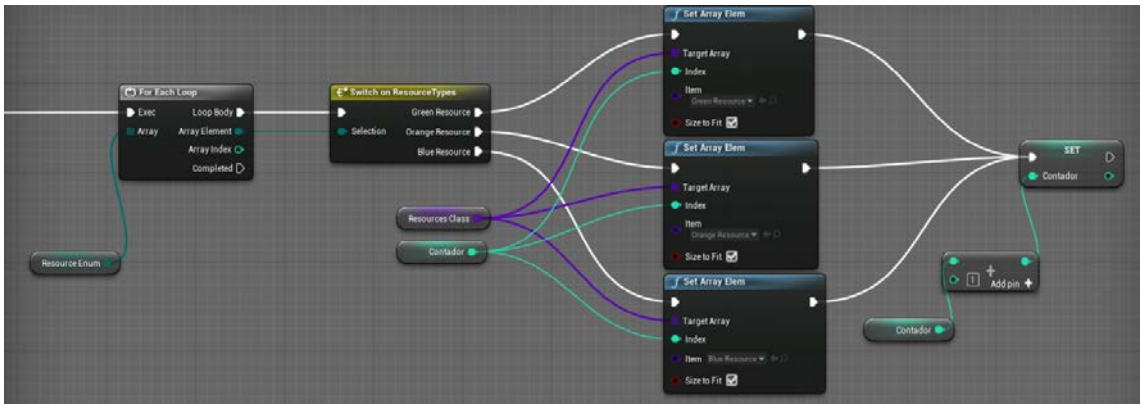
Punto de Spawn

El spawner es una caja invisible en la que su interior se generan los recursos. A través del editor se pueden modificar varias variables del spawner:

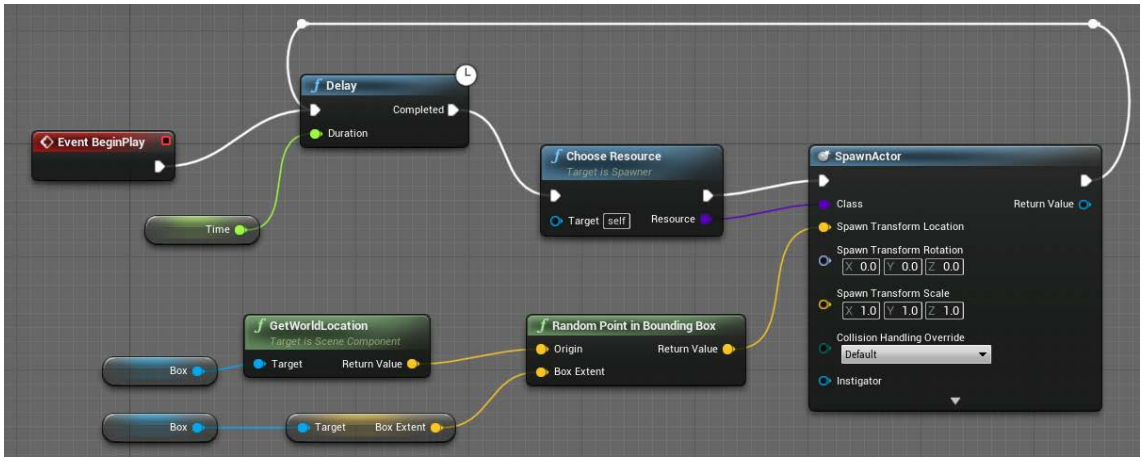


El tiempo (“Time”) que tarda entre generar recursos, el tamaño tanto en el eje X como en el eje Y (“Size”) de la caja, los tipos de recursos que generan y las posibilidades de que genere cada uno. Estos dos son arrays, y se pueden quitar o añadir recursos, haciendo que se dejen de producir algunos recursos si se quitan del array. La selección de los recursos se produce gracias a un “Enumeration” que tiene tres elementos, uno por cada tipo de recurso.



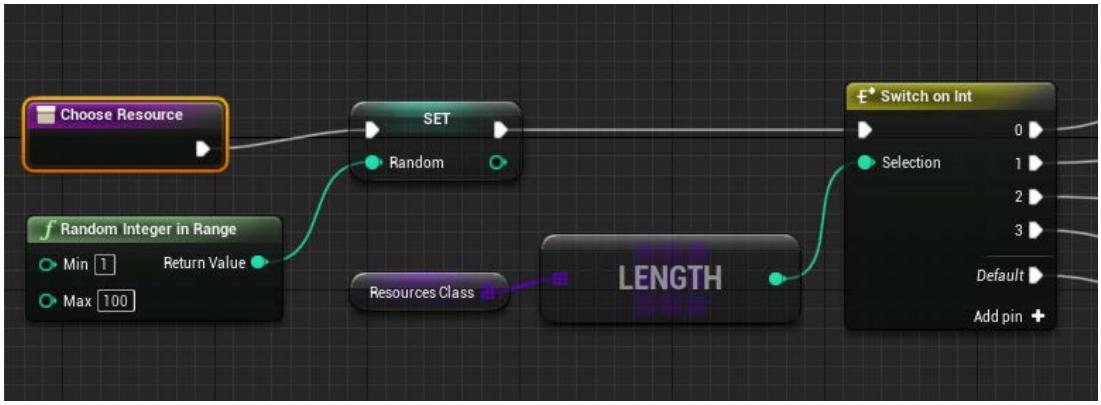


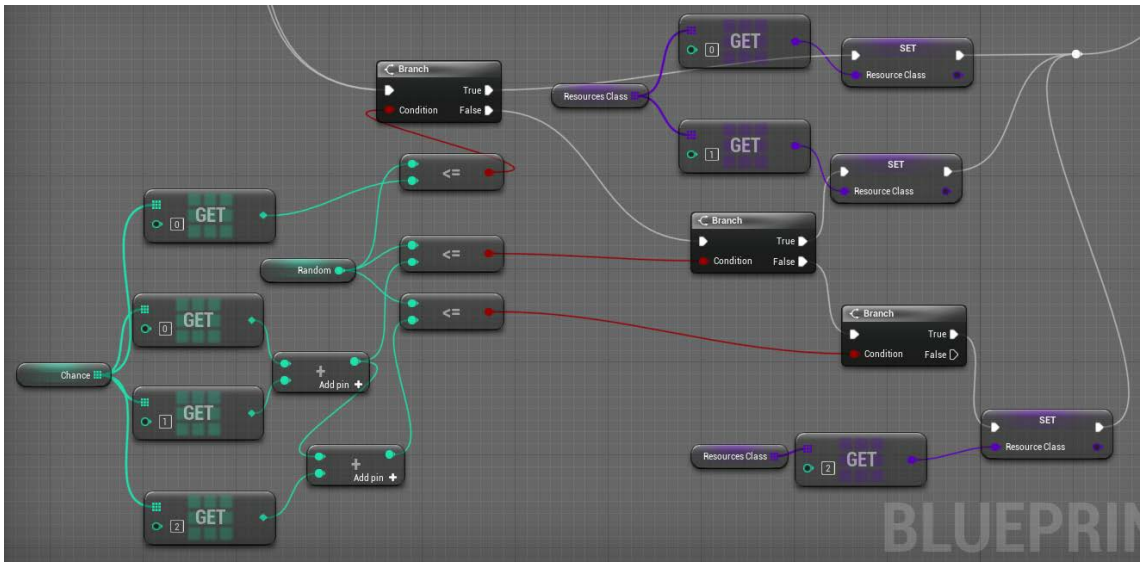
Para la generación de recursos se necesita elegir un tipo de recurso y una posición. Para la posición se usa una función llamada “Random Point in Bounding Box” que coge un punto aleatorio de la caja del spawner. Para elegir el recurso se creó una función llamada “Choose Resource” que devuelve el recurso elegido dependiendo de las probabilidaddes.



La función de Spawn se llama a si misma con un Delay en el que se introduce el tiempo que se haya establecido para que así se creen en el tiempo que quiera el usuario.

Choose Resource es la función que elige el resource dependiendo de la probabilidad establecida. Tiene un Switch que mira la longitud del array de recursos, ya que cuantos menos hay, menos comparaciones hay que hacer.



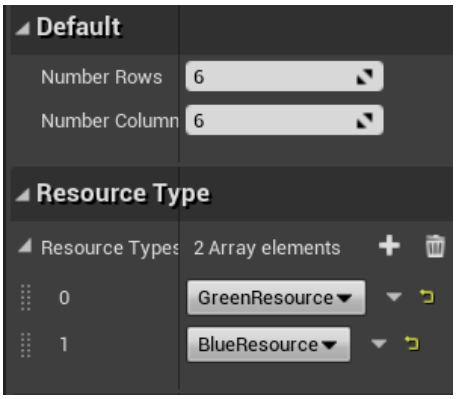


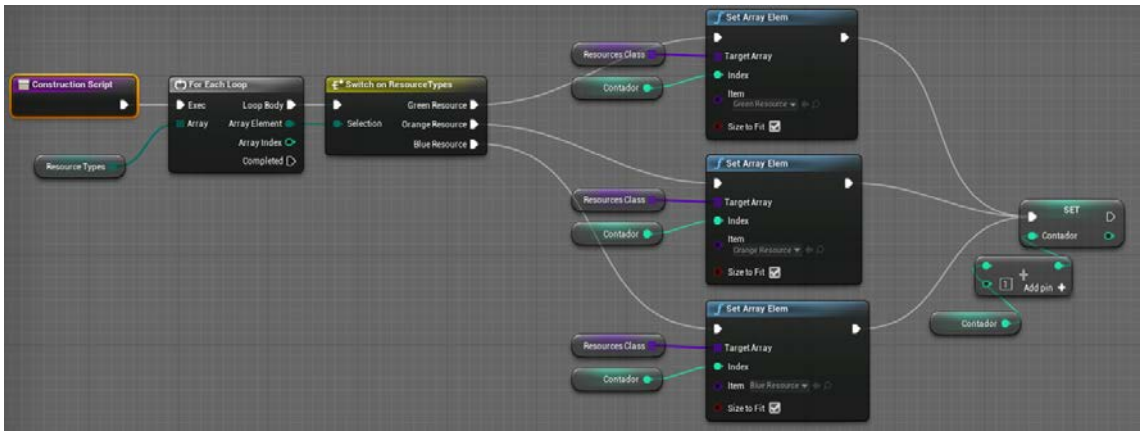
Esto son los nodos que deciden entre los tres tipos de recursos. Si en el array hay tres tipos de recursos, se tendrán que hacer tres comprobaciones. Para dos recursos es básicamente lo mismo, pero con una comprobación menos, y si es solo uno, solo hay que hacer una.

El sistema funciona comparando los porcentajes con un numero aleatorio del 1 al 100. Si el número que ha salido es menor al primer porcentaje, se creará el recurso número 1, si no se comprobará si es menor que el primer porcentaje más el segundo (ya que sabemos que no es menor que el primero, hay que sumar), si esto se cumple, se crea el recurso número 2. Luego, si no se cumple, se hace lo mismo con el porcentaje número tres (que es la suma de todos) y si es correcto se genera el recurso. Si se ponen porcentajes que no suman 100% no siempre se crearán recursos.

Almacén

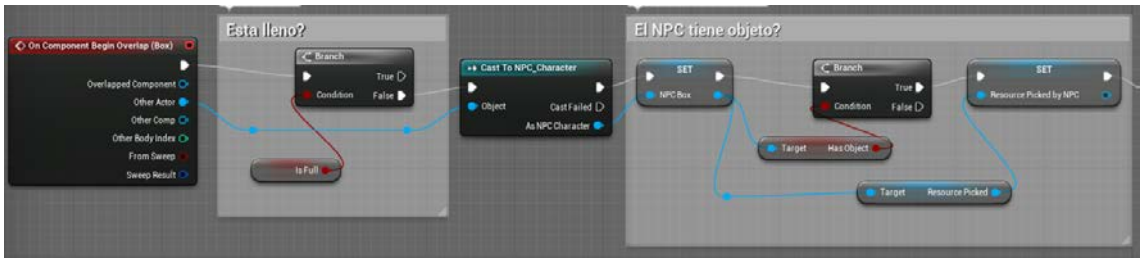
En los almacenes se puede configurar el número de filas y columnas de recursos que pueden almacenar. Aparte hay un array de los tipos de recursos que pueden almacenar. La selección de los recursos funciona con la enumeración como en el spawner.



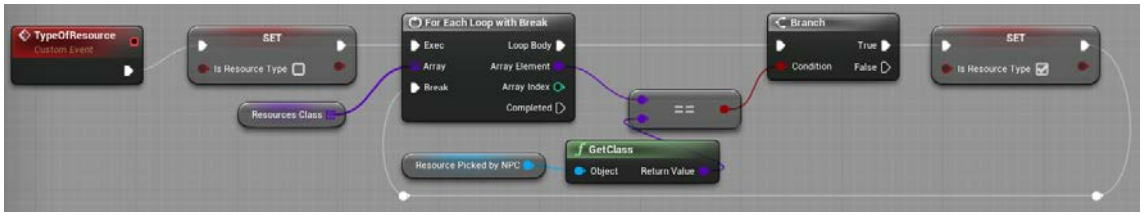


Los almacenes funcionan de forma similar a los recursos, son una caja en la que, si entra un NPC con un recurso del tipo correcto, el recurso se colocará en la posición correcta.

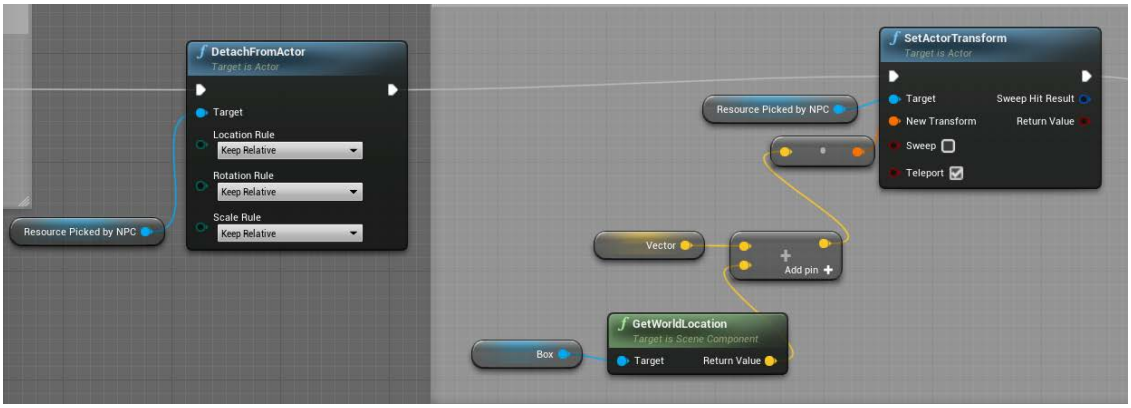
Lo primero que comprueba un almacén cuando un NPC entra en él, es si el propio almacén esta lleno. Hay un bool que cambia cuando el almacén se llena. Tras esto comprueba que el NPC tiene un objeto.



Una vez se sabe que no está lleno y que el NPC tiene un objeto, hay que comprobar si ese recurso es del tipo compatible con los que puede almacenar. Para esto hay un evento que comprueba si el tipo del recurso está en el array de recursos del almacén.



Si el recurso es del mismo tipo entonces el recurso dejará de ser hijo del NPC y se colocará en la posición correcta en el almacén.

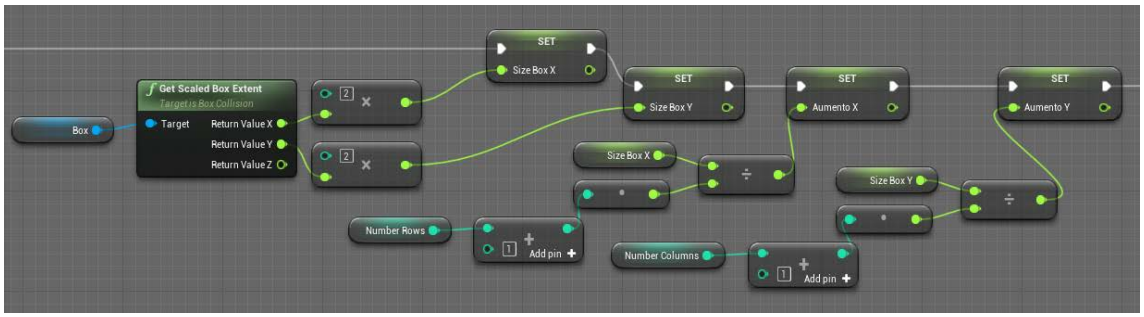


El vector es relativo a la caja que es el almacén, por eso se suma a la posición de este.

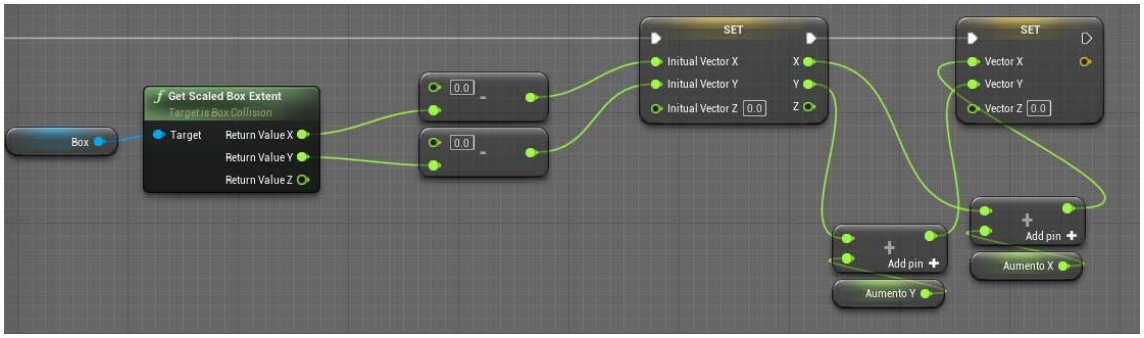
Calculo de las posiciones

El cálculo de las posiciones está basado en intentar que todos los recursos entre si estén a la misma distancia y así ocupen la superficie uniformemente.

Lo primero que hace es guardar el tamaño del almacén. Se guarda el tamaño del eje X y el tamaño del eje Y. Luego al tamaño del eje X se divide entre el número de filas más uno, y en el eje Y se hace lo mismo, pero con las columnas. Se guarda este valor en dos variables de tipo float, “AumentoX” y “AumentoY”.

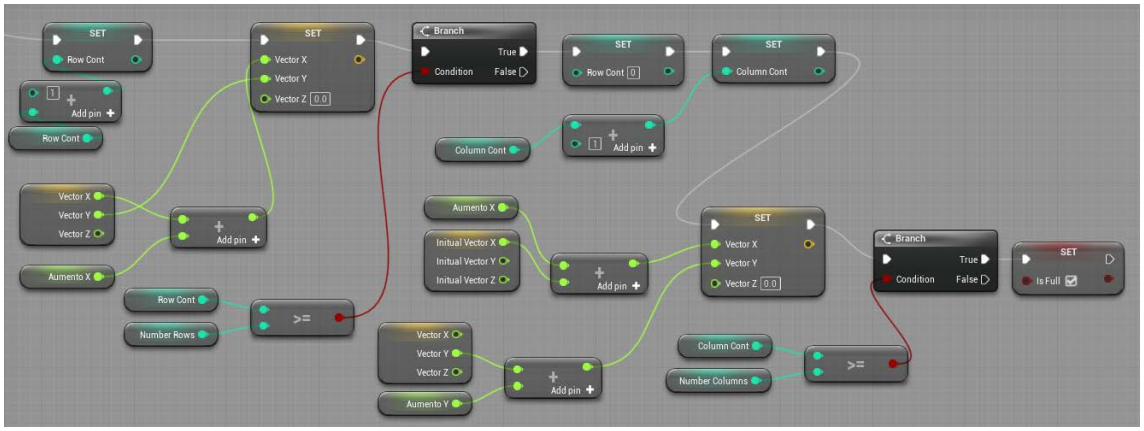


Después se almacena en un vector los valores del almacén con valor negativo (para tener el punto inicial en el que habría que empezar) y se guarda en otro vector este vector, pero sumando los aumentos en cada eje. Esta será la posición que ocupe el primer recurso.



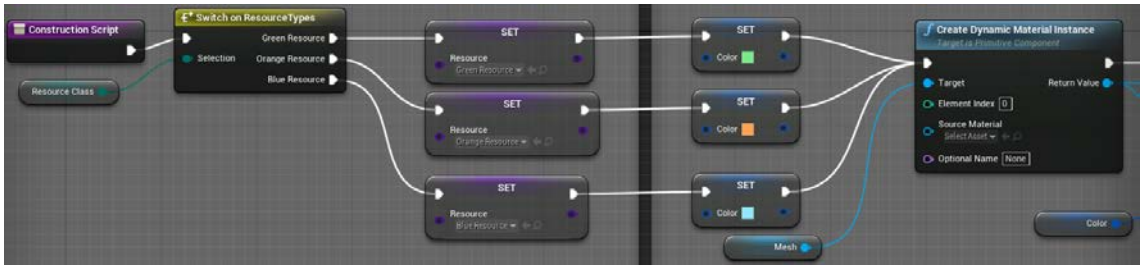
Una vez se coloca un recurso, para calcular la próxima posición se suma al eje X el aumento. Hay un contador para comprobar el número de recursos que hay en cada eje. Cuando se aumenta la posición en el eje X se aumenta el contador de filas. Una vez este contador llegue al número de filas máximas, pasará a la siguiente columna reiniciando la posición del eje X y aumentando el Y, y aumentará el contador de las columnas.

Una vez llegue al máximo de columnas y filas, se establecerá que está lleno el almacén.



Agentes

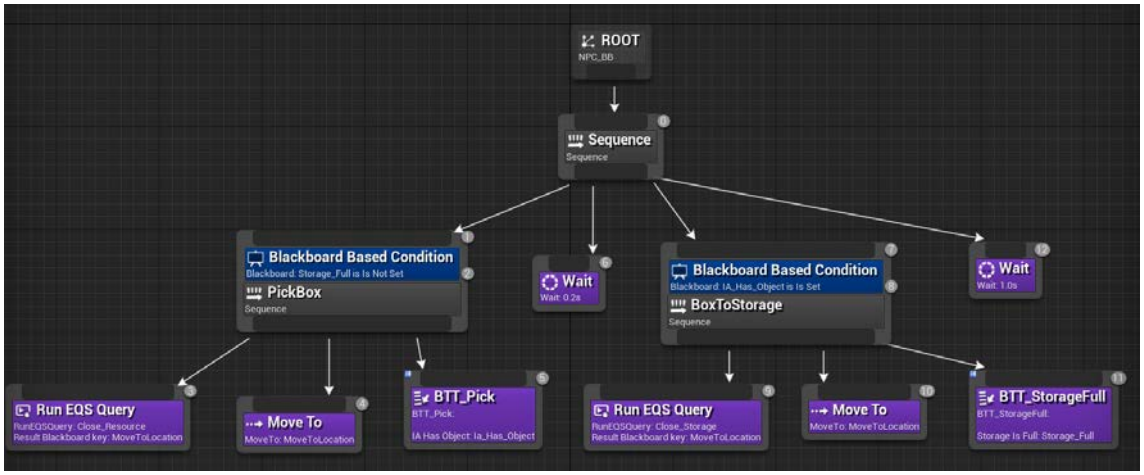
Los NPCs tienen una variable publica (la enumeración de tipos de recurso) para establecer el tipo de recurso que pueden recoger (se guarda la clase en una variable privada). Dependiendo del recurso, cambiará el color del NPC.



Aparte de esto el NPC tiene una variable de tipo bool para ver si tiene un objeto o no, que se usa para cuando pasa por encima de un almacén y no tiene objeto que el almacén no tenga que hacer más comprobaciones. También se usa en los recursos para que no coja uno si ya tiene.

EQS

Los NPCs siguen el siguiente árbol de comportamiento:



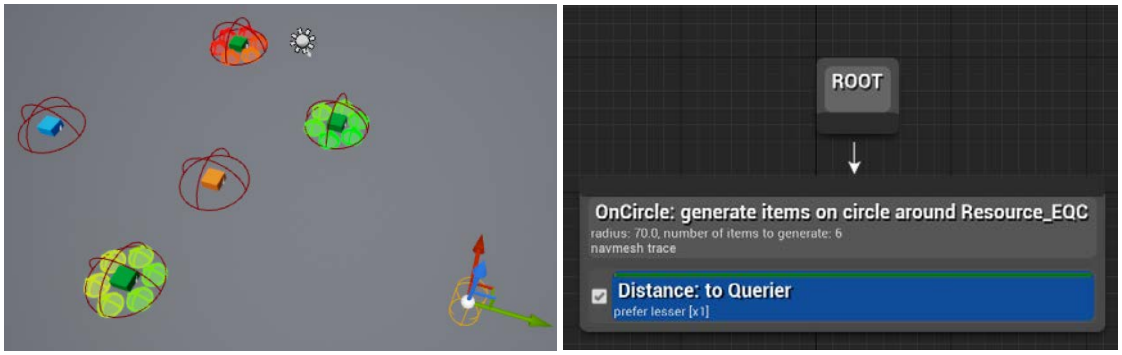
Primero buscan los recursos más cercanos, se mueven hasta el objeto y lo recogen.

Una vez tienen el objeto buscarán el almacén más cercano del tipo de recurso que han recogido e irán a dejarlo en él.

Para acceder a la primera rama se debe cumplir la condición de que no estén llenos los almacenes del tipo de recurso que el NPC puede llevar. Esta condición es un bool del Blackboard que se cambia en la acción llamada “BTT_StorageFull”, la cual se llama después de dejar un objeto en un almacén.

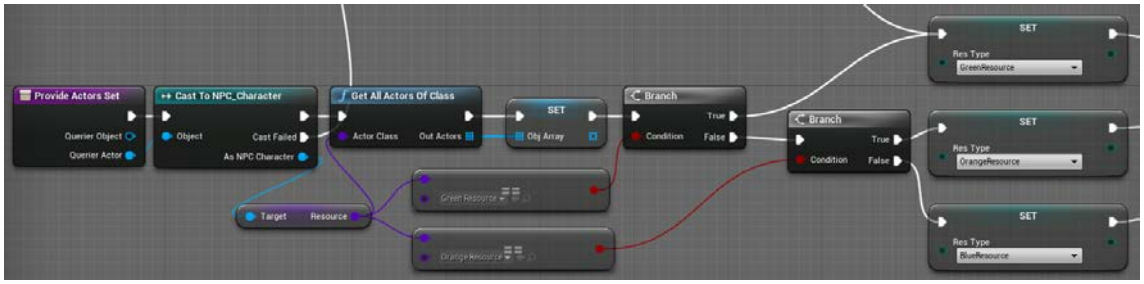
En la primera rama, lo primero que se ejecuta es la Query “Close_Resource” que busca el recurso del tipo que puede coger más cercano al NPC.

Lo que hace esto es generar un grid circular alrededor de los objetos que el NPC puede recoger. Luego para decidir cuál es el mejor, usa la distancia del NPC a los puntos del grid.

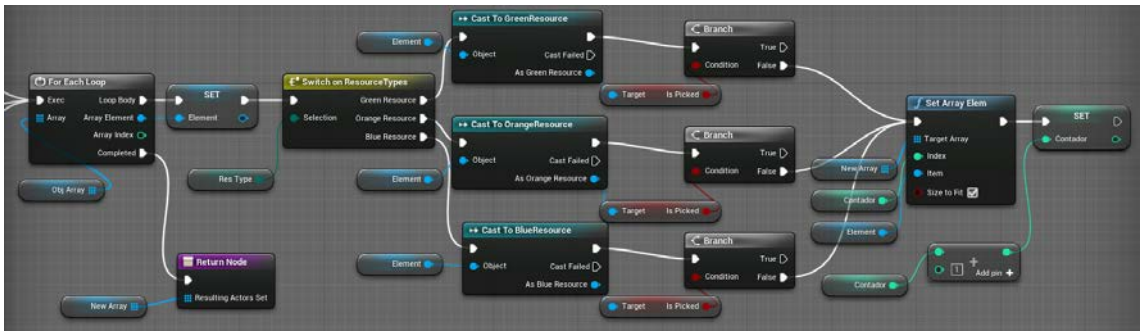


Para poder eliminar de la búsqueda los recursos de otros tipos se usa un contexto personalizado. El contexto busca todos los actores del tipo de recurso que el NPC puede recoger y luego comprueba si han sido recogidos o no (con el bool de si han sido recogidos que tienen los recursos).

El contexto “Resource_EQC” hace un Cast al NPC, luego encuentra todos los actores del tipo de recurso que puede recoger y los guarda en un array.

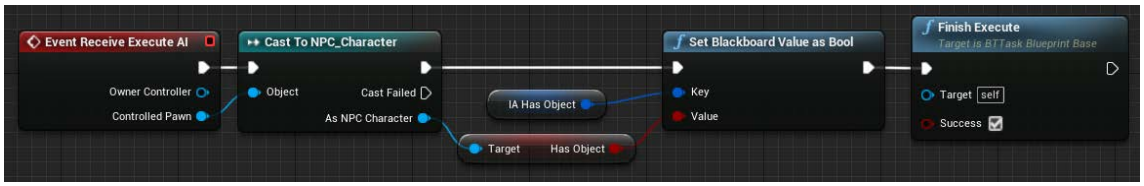


Luego hace un cast al tipo de recurso para comprobar si ha sido recogido. Guarda en un nuevo array los que puede recoger y eso es lo que devuelve.



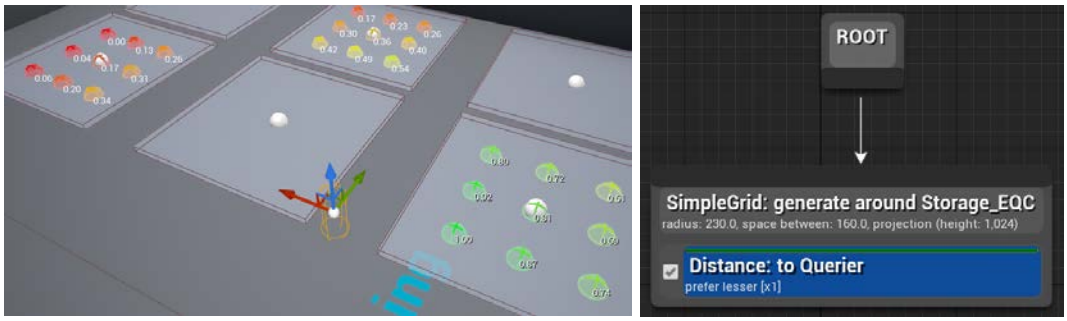
La Query devuelve un vector que se guarda en una variable del Blackboard y luego se usa el “Move to” a esa variable para que el NPC se mueva.

El “BTT_Pick” cambia una variable bool del Blackboard dependiendo de si el NPC tiene un objeto o no.

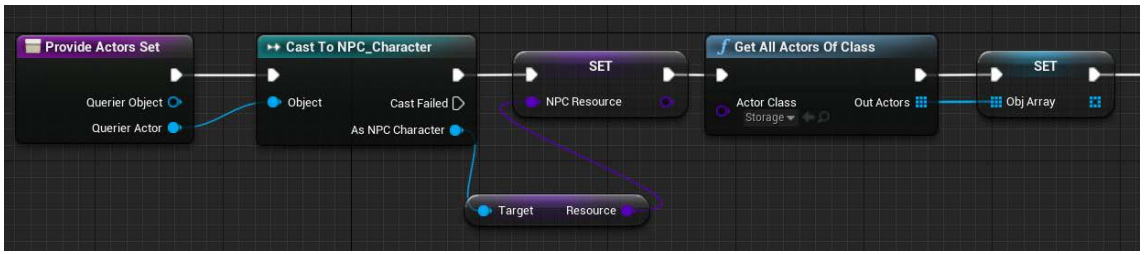


Si la variable del Blackboard que se encarga de saber si el NPC tiene un recurso o no es true (básicamente, si el NPC ha recogido un objeto) entonces el NPC avanzará a la rama de encontrar un almacén, si no repetirá la primera rama (la de recoger un objeto)

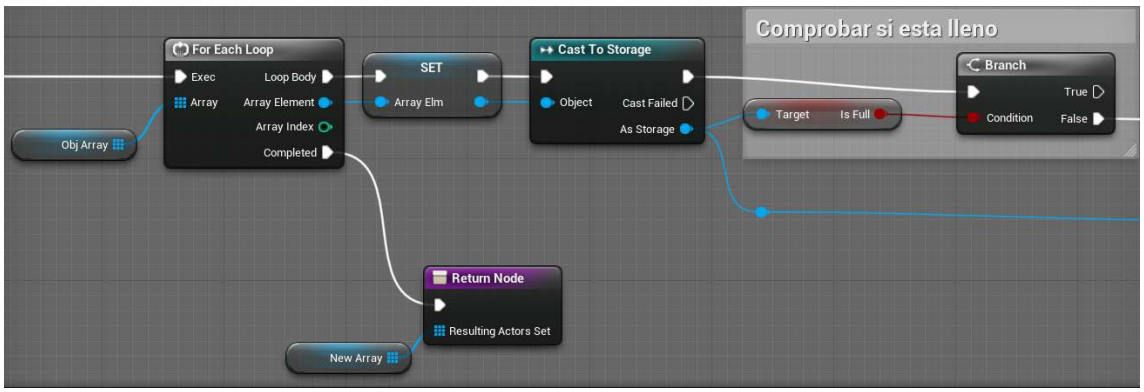
Lo primero que sucede en la rama de encontrar el almacén es la Query “Close_Storage”, que busca el almacén que puede almacenar el tipo de recurso que tiene el NPC



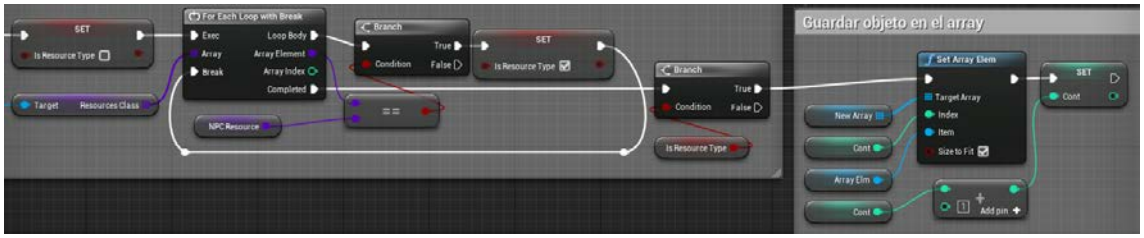
El contexto “Storage_EQC” hace un cast al NPC y almacena el tipo de recurso que puede recoger. Lo siguiente es buscar todos los elementos de tipo almacén y guardarlo en un array.



Después, para cada elemento del array se hace un cast a “Storage” y se comprueba el bool de si está lleno el almacén.

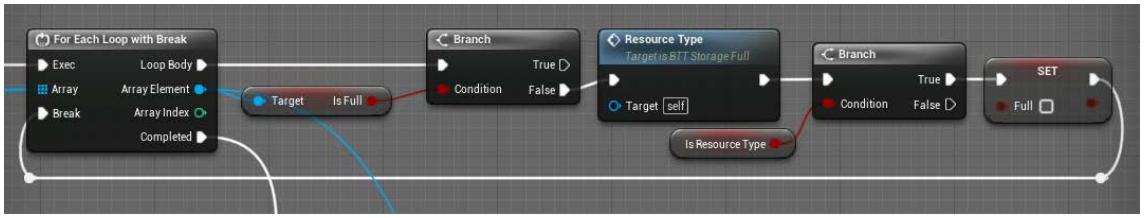


Si el almacén no está lleno entonces hay que comprobar si puede almacenar el tipo de recurso que tiene el NPC. Para ello se comprueba si alguno de los recursos del array es del mismo tipo que el del NPC. Los que son del mismo tipo se guardan en un nuevo array que es el que luego devuelve el contexto

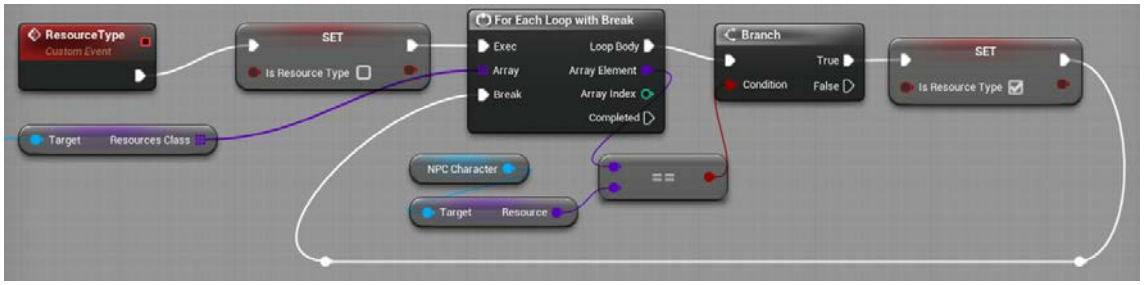


La Query devuelve la posición del almacén más cercano en forma de vector, al igual que en la búsqueda de recursos, y luego se mueve a esa posición con la función “Move to”.

Lo que se quiere saber con el “BTT_Storage_Full” es si todos los almacenes del tipo de recurso que puede recoger el NPC están llenos. Para ello se buscan todos los actores de tipo Storage y se guardan en un array. Luego para cada almacén se comprueba el bool de si están llenos.



Si no están llenos se comprueba el tipo de recurso del almacén a ver si coincide con el del NPC. Se hace con un evento.



Después si alguno ya se sabe que no está lleno, se hace un “break” en el “for each” y se cambia la variable del Blackboard que indica si están llenos los almacenes.