Universitat de Girona
Escola Politècnica Superior

Grau en Disseny i Desenvolupament de Videojocs

PROJECTE FINAL DE GRAU

Desenvolupament d'un videojoc d'acció i aventures en 3D

Autora:
Elena Tera Faba

Tutors:
Francesc Xavier Costa
Gustavo Patow

MEMÒRIA

Convocatòria:
Setembre 2023

Departaments:
Informàtica, Matemàtica Aplicada i Estadística
Organització, Gestió Empresarial i Disseny del Producte

Índex

1. Introducció i objectius

1.1. Introducció

Actualment, la indústria dels videojocs està creixent de manera excepcional. Cada cop els jugadors tenen més varietat per a escollir i més oportunitats per a trobar un gènere i estil que s'adeqüi als seus gustos.

D'entre tots, el gènere d'acció-aventures és dels més populars, dins d'aquest hi pot haver altres subgèneres; per exemple, puzles, *shooter*, supervivència... A més, cadascun pot tindre un estil artístic propi d'entre un gran ventall de possibilitats. Aquest és un gran atractiu pels jugadors perquè, per molt que unes bones mecàniques siguin les que majoritàriament decideixen si un videojoc triomfa o no, és l'estètica la que és capaç d'atraure i captar l'atenció des d'un primer moment.

Tot i que, com s'acaba de comentar, hi ha una àmplia varietat de gèneres i estètiques, s'ha trobat una gran mancança d'un estil visual amb components orientals en els jocs d'acció-aventures, que estaria bé explotar. Per aquest motiu, s'ha decidit que l'estètica d'aquest projecte begui en gran part de la cultura japonesa.

1.2. Motivacions

Les motivacions principals per a realitzar aquest projecte han estat principalment les de poder experimentar el procés de creació d'un videojoc i cadascuna de les seves etapes, des de la concepció de la idea fins al seu desenvolupament, incloent tant la part més artística com la més tècnica, i d'aquesta manera poder aprendre la metodologia de treball que es duu a terme actualment a la indústria.

Tot i que el treball es centra sobretot en el modelat, texturitzat i composició de les diferents peces, s'ha volgut posar en pràctica tots els coneixements que s'han adquirit en els diferents àmbits al llarg del grau.

1.3. Propòsits i objectius del projecte

El propòsit principal d'aquest treball és el de desenvolupar un videojoc d'acció i aventures en un entorn natural 3D, en el que hi ha una forta component artística ambientada en la cultura tradicional japonesa. Els diferents *assets* estaran modelats amb *Blender* i texturitzats amb *Substance Painter* per a que, posteriorment, puguin ser integrats en el motor de videojocs *Unreal Engine 5*.

A més, s'implementaran diferents mecàniques pel jugador i *IA*s pels enemics, també es crearan diferents materials, sistemes de partícules i *shaders* per a acabar de completar tota l'estètica de l'entorn.

Per últim, pel que fa a altres elements com la major part de les animacions i els sons, es farà una recerca per a trobar els que més s'adeqüin al joc.

Així doncs, resumint, els objectius principals del projecte són els següents:

- Planificar les diferents fases del desenvolupament i disseny del joc.
- Estudiar els elements i cultura tradicional japonesa.
- Dissenyar, modelar i texturitzar els diferents *assets*.
- Fer recerca d'animacions i sons que siguin coherents amb l'entorn.
- Creació de materials, *VFX* i *shaders.*
- Disseny d'interfícies.
- Integrar els *assets* a l'entorn.
- Implementar les mecàniques del personatge i la IA dels enemics.
- Revisar i finalitzar la documentació desenvolupada al llarg del projecte.

1.4. Distribució de tasques

El desenvolupament d'aquest treball es divideix en quatre grans parts. Cadascuna d'aquestes té un propòsit diferent, però totes elles són molt importants per a que el joc es senti complet. Tot i així, se li han donat diferents pesos, ja que s'ha preferit aprofundir en la component artística.

Essent així, les diferents tasques s'han distribuït de la següent manera:

Estètica	60%
Narrativa	5%
Mecàniques	10%
Tecnologia	25%

Com s'ha comentat en seccions anteriors, s'ha volgut dedicar més temps a l'estètica, el que inclou el modelat, texturitzat, creació de materials, interfícies, a més de la composició de tots aquests elements.

Per altra banda, a la narrativa se li ha donat poc pes, tot i així, se li proporciona un sentit, de forma molt breu, a l'objectiu del joc per a que el jugador l'acabi de comprendre.

Pel que fa a les mecàniques, el disseny és senzill, però coherent amb el que es vol transmetre i la simplicitat de l'entorn. A més s'ha preferit aconseguir que aquestes no siguin excessivament complicades d'implementar, per a així poder-se centrar en altres aspectes.

Per acabar, la tecnologia també té bona part del pes del projecte, ja que se li ha dedicat molt de temps a la implementació de les mecàniques, la IA dels enemics i fer que les diferents funcionalitats que es troben al llarg del joc funcionin correctament.

2. Estudi de viabilitat

És necessari fer un estudi de viabilitat previ al disseny i desenvolupament del projecte per a poder assegurar-se de que es tenen els recursos suficients (programari, hardware, personal, temps...) per a dur-lo a terme sense problemes.

2.1. Viabilitat tecnològica

2.1.1. Hardware

Per a la realització d'aquest treball s'ha utilitzat el següent hardware:

- Ordinador:
  - Processador: AMD Ryzen 7 5800X 8-Core
  - Memòria RAM: 16GB
  - Targeta gràfica: NVIDIA GeForce RTX 3070

- Tauleta gràfica Wacom Bamboo
- Comandament Xbox X|S

Ja es disposava de tots aquests recursos un cop iniciat el projecte, el que vol dir que no ha suposat cap inversió inicial en aquest aspecte.

2.1.2. Software

El software utilitzat al llarg del desenvolupament ha estat el següent:

- Unreal Engine 5.0.3: motor de joc utilitzat durant el desenvolupament.
- Blender: eina gratuïta de modelat 3D.
- Substance Painter: programa de texturitzat.
- Tree It: programa gratuït per a la generació d'arbres en 3D.
- Adobe Photoshop: software d'edició fotogràfica que s'ha fet servir per a la creació de menús i textures.
- Clip Studio Paint: eina d'il·lustració, utilitzada principalment per a la realització d'esbossos.
- PureRef: programa per a facilitar la visualització d'imatges de referències pel modelat o dibuix.

Totes les llicències han estat adquirides prèviament, i altres s'han aconseguit gràcies a identificar-se com a estudiant, així doncs el cost del programari és nul.

2.2. Recursos Humans

Per a dur a terme aquest projecte també s'han de tenir en compte els múltiples rols que han de desenvolupar les diferents persones d'un equip, en aquest cas, es fa de manera individual, així que les tasques assignades als diferents perfils es realitzaran per una sola persona que ha d'assumir els següents rols:

- Dissenyador
- Artista 3D
- Programador/ tester

2.3. Viabilitat econòmica

Tot i que els recursos tecnològics ja s'havien adquirit prèviament i no han suposat cap cost addicional al desenvolupament del projecte, a la següent taula es pot veure el cost aproximat del hardware i programari:

Hardware	**Cost**
Ordinador	2179,21€
Tauleta gràfica Wacom	60€
Comandament Xbox Series X|S	60€
Llicència Adobe Photoshop	290,17€
Llicència Clip Studio Paint	20€
Total	**2609,38€**

Per altra banda, tal i com s'ha comentat en els apartats anteriors, el desenvolupament del projecte ha estat realitzat per una sola persona, així que el cost dels recursos humans és nul, de totes maneres s'ha fet una estimació aproximada pels diferents rols, cadascun d'aquests està simplificat, és a dir, assumirien tasques que es podrien encomanar a més treballadors:

Rol	**Cost/hora**
Dissenyador	16€/h
Artista 3D	13€/h
Programador/tester	13€/h
Total	**66€/h**

2.4. Estudi de mercat

Abans de començar el desenvolupament i el disseny del projecte, cal fer una recerca dels jocs d'aventura més populars que presenten una estètica oriental per a poder estudiar i analitzar diferents aspectes de la competència. Gràcies a això, és possible comparar i tractar de millorar el que es troba actualment al mercat.

2.4.1. Estat de l'art

S'han trobat diferents videojocs que compleixen amb el gènere i l'estil desitjats, alguns dels més populars són el Ghost of Tsushima, Ōkami i Aragami 2.

- Ghost of Tsushima
  Ghost of Tsushima s'ha convertit en un joc de referència pel gènere, tant per les mecàniques de lluita, com pels impressionants escenaris (Figura 1). És un joc que porta al jugador al Japó feudal, concretament a l'illa de Tsushima i li fa posar-s'hi en la pell d'un samurai que ha de lliurar diferents batalles contra els mongols per a recuperar l'illa.

Figura 1. Ghost of Tsushima

- Ōkami
  Per altra banda, Ōkami és un altre dels videojocs d'acció-aventura que es situen al Japó, l'estil és molt característic, ja que simula la tècnica sumi-e (tècnica de pintura amb tinta negra), però amb molt més color, com es pot veure a la Figura 2.
  En aquest joc es juga com a Amaterasu, la deessa del sol, que adopta la forma d'un llop blanc que ha de salvar la seva terra de les mans d'un monstre que l'està destrossant.

Figura 2. Ōkami

- Aragami 2
  Per acabar, Aragami és un joc de sigil en tercera persona, en el que el jugador controla a un guerrer que ha d'enfrontar a diferents exèrcits invasors. Tot i que

aquest té un fort component de sigil, és un bon exemple de videojoc d'acció que es desenvolupa en una zona plena d'elements de la cultura japonesa, com es pot apreciar a la Figura 3.

Figura 3. Aragami 2

2.4.2. Comparació de l'estat de l'art

Una vegada s'han identificat al mercat alguns dels jocs que compleixen amb el gènere i estil artístic desitjats, s'ha fet una comparació entre ells per a poder identificar els punts forts del projecte proposat, que s'ha anomenat Hana.

	Ghost of Tsushima	**Aragami 2**	**Ōkami**	**Hana**
Gràfics	9	7	7	8
Jugabilitat	8	7	8	6
Diversió	8	7	8	7
Abast del públic	6	6	8	9
Fantasia	6	8	9	9
Estètica japonesa	9	9	9	9
Gènere Acció-Aventura	9	7	9	9
Puzles	6	7	8	7
Representació femenina	7	6	6	8

Com mostra la taula anterior, pel que fa als gràfics, es creu que es pot aportar una estètica estilitzada, lluny del realisme del Ghost of Tsushima i més semblant als altres dos jocs, tot i que amb les seves diferències i millores.

Seguidament, la jugabilitat és l'aspecte que pitjor nota té, però com s'ha estat comentant en els apartats anteriors, l'atractiu principal del joc és el disseny de l'entorn.

Pel que fa a la diversió, tots els jocs, essent majoritàriament del mateix gènere, tenen unes mecàniques i estils molt diferents, però amb l'exploració, enfrontaments amb enemics i puzles definits al Hana, es creu que no hi ha lloc per l'avorriment.

A continuació, com s'ha pogut comprovar segons el PEGI dels diferents jocs, tant el Ghost of Tsushima com l'Aragami, estan enfocats per a un públic més adult,

majoritàriament, pel seu grau de violència. El Hana és el que poc abastar més públic, tant infantil com adult.

Per altra part, s'ha afegit la fantasia com a punt fort en el projecte, ja que tant els enemics com alguns elements que es mostraran més endavant, no són gens realistes i es creu que juga a favor d'aquest treball, ja que es complementa perfectament amb l'idea d'estil i públic objectiu.

Respecte l'estètica japonesa i el gènere, tots compleixen amb nota alta aquests dos aspectes, ja que la recerca ha estat a partit d'aquests, tot i així, com s'ha comentat anteriorment, l'Aragami està molt més lligat al gènere del sigil.

Pel que fa als puzles, tant el Ghost of Tsushima com l'Aragami no es defineix per incloure'n (al menys no són gaire importants), en canvi l'Ōkami sí que presenta algun trencaclosques al llarg del joc, a l'igual que el projecte proposat.

I per últim, encara que actualment el mercat ja compte amb molts més videojocs amb protagonistes femenines, encara és necessària més representació. Per això, com està previst que el joc també arribi a un públic infantil, les nenes es podran veure representades en aquests personatges.

Així doncs, amb els resultats obtinguts i les conclusions extretes, s'ha comprovat que la proposta cobreix algunes mancances actuals en el mercat i es veu factible seguir endavant amb el projecte.

2.5. Públic objectiu

Per a poder dissenyar un joc adequadament és necessari saber quines són les necessitats, gustos i preferències del públic objectiu al que va dirigit, d'aquesta manera es pot adaptar i enfocar tot el desenvolupament de manera adient per a captar l'atenció d'uns jugadors en concret.

2.5.1. Tipologia de jugadors

Segons la classificació establerta per Richard Bartle, com es pot veure a la Figura 4, existeixen quatre perfils de jugadors que es poden classificar de la següent forma:

- *Achievers*: són jugadors competitius que gaudeixen superant reptes difícils, tant imposats pel propi joc com per ells mateixos.
- *Explorers*: els hi agrada explorar tot el món i l'entorn en detall, esbrinar tots els secrets que s'amaguen i saber cada possible drecera.
- *Killers*: aquests jugadors gaudeixen interactuant amb altres, però estan més enfocats a competir entre ells que no pas a col·laborar. Volen destacar i controlar el joc, fan el possible per guanyar.
- *Socializers*: a diferència dels *killers* aquests jugadors prefereixen jocs on puguin col·laborar amb altres. Són bons i els hi agrada treballar en equip.

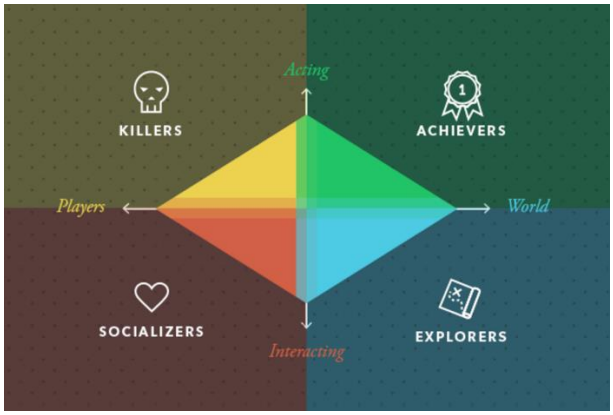

Figura 4. Tipologia de jugadors segons Richard Bartle

Tenint això en compte, es pot afirmar que aquest projecte està dirigit als *explorers*, ja que com l'objectiu principal és el disseny i modelat de l'entorn s'espera que els jugadors el puguin gaudir explorant-lo.

2.5.2. Perfil del jugador

El perfil del jugador ideal és el que li agraden els gèneres d'aventura i acció. No cal que aquest tingui una gran experiència de cap tipus, ja que no es trobarà amb massa dificultat al llarg del recorregut. Per altra banda, es creu que l'edat idònia seria, com a mínim, a partir dels 10 anys.

I per acabar, tal i com s'acaba de comentar en l'apartat anterior, pertany al perfil d'*explorers* segons la classificació de Bartle.

2.5.3. Motivacions, necessitats i emocions del jugador

Un videojocs poden transmetre una àmplia varietat d'emocions i sentiments als jugadors, tant positius, que són els que s'intenten potenciar i complir, com negatius, que es tracten de minimitzar. Algunes de les emocions positives que es poden sentir mentre es juga a aquest projecte són el desig/contemplació, l'expectació, l'alleujament i la satisfacció.

2.5.4. Arc emocional

Al començament del joc, el jugador pot sentir expectació i contemplació provocada per l'entorn, el que farà que vulgui continuar explorant i descobrint cada racó del mapa. Quan s'hagi d'enfrontar als enemics i al puzle que l'espera pel camí, i sigui capaç de superar-los, tindrà la una sensació d'alleujament i satisfacció, que tornarà a experimentar quan arribi al destí del joc.

3. Planificació

Per a poder tenir un control de les diferents tasques i el temps que s'ha dedicat a cadascuna d'elles, s'ha creat un diagrama de Gantt (Figura 5), on s'ha definit, de manera aproximada, què s'ha de fer i en quant de temps.

Primer de tot, es defineix la idea per a poder començar el disseny d'una forma més segura i estable. Seguidament, es dissenya de manera molt bàsica i esquemàtica l'entorn, per així poder integrar el personatge i implementar les mecàniques amb una base.

A continuació, es dissenyen i implementen els enemics, amb això acabat, ja es pot testejar el combat.

Després es creen les interfícies, que proporcionen feedback al jugador.

Tot seguit, es modelen els *assets*, la part més important del projecte; es creen diferents materials i efectes que acompanyen l'entorn, i s'integra tot.

Finalment, s'implementen els sons i es genera la il·luminació, elements que fan que el videojoc quedi més rodó.

Pel que fa a les dates, no s'han pogut complir totes, algunes tasques han portat menys dies dels esperats i altres més. De totes maneres el que sí s'ha fet ha estat seguir l'ordre marcat.

Tasca	Data Inici	Dies	Data Fi
Definició de la idea			
Recerca d'informació	07/01/2023	1	08/01/2023
Esbossos	08/01/2023	2	10/01/2023
Disseny bàsic de l'entorn			
Modelat del terreny	10/01/2023	1	11/01/2023
Blocking	11/01/2023	3	14/01/2023
Disseny del personatge			
Concept, modelat i texturitzat	14/01/2023	30	13/02/2023
Recerca i modificació d'animacions	13/02/2023	7	20/02/2023
Modelat, texturitzat arma	20/02/2023	4	24/02/2023
Animació arma	24/02/2023	3	27/02/2023
Implementació mecàniques			
Implementació mecàniques	27/02/2023	20	19/03/2023
Disseny dels enemics			
Concept, modelat i texturitzat	19/03/2023	15	03/04/2023
Creació d'animacions	03/04/2023	2	05/04/2023
Implementació enemics			
IA	05/04/2023	20	25/04/2023
Sistema de partícules	25/04/2023	1	26/04/2023
Interfícies			
Disseny i implementació dels menús	26/04/2023	4	30/04/2023
Disseny i implementació del HUD	30/04/2023	2	02/05/2023
Disseny final de l'entorn			
Modelat i texturitzat dels assets	02/05/2023	70	11/07/2023
Sistemes de partícules	11/07/2023	4	15/07/2023
Creació de materials y shaders	15/07/2023	10	25/07/2023
Composició i modificacions inicials	25/07/2023	7	01/08/2023
Implementació de sons			
Recerca i integració	01/08/2023	2	03/08/2023
Generació de la il·luminació	03/08/2023	2	05/08/2023
Documentació	07/01/2023	240	04/09/2023

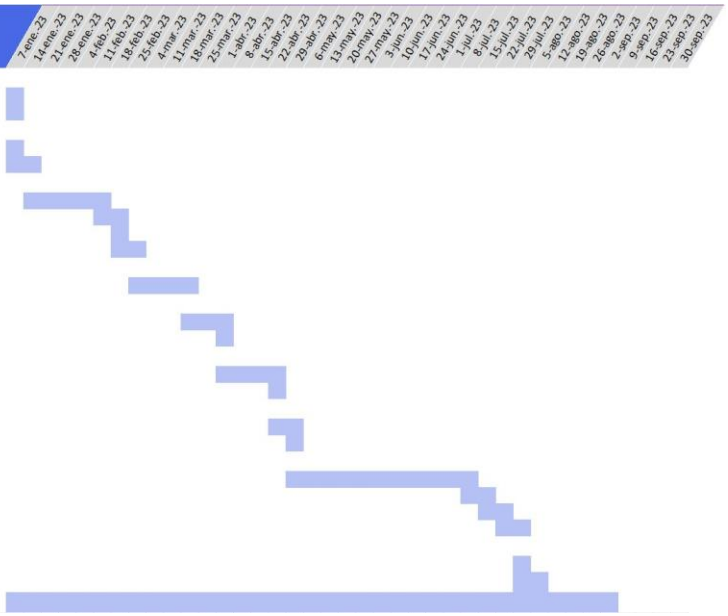

Figura 5. Diagrama de Gantt

4. Marc de treball i conceptes previs

Tal i com s'ha comentat en l'Apartat 2.2 el projecte s'ha dut a terme de forma individual, essent una sola persona la que ha realitzat totes les tasques esmentades anteriorment.

Com s'ha volgut dedicar la major part del temps en el disseny i creació de l'entorn amb components de la cultura japonesa, s'ha cregut oportú fer recerca de diferents d'aquests elements per a aprendre el significat i la importància. D'aquesta manera el videojoc adquireix coherència visual i estètica.

4.1. Conceptes previs

4.1.1. Cultura japonesa

Uns dels elements que destaquen en el videojoc són les màscares. Aquestes es tallaven en fusta i s'utilitzaven durant el període Jomon (14500-300 a.C) en rituals religiosos per a cobrir el rostre dels difunts. Posteriorment, al llarg del període Kofun (250-538 d.C) i l'arribada del Budisme al Japó, es van començar a utilitzar per a celebrar festivitats i altres cerimònies.

N'hi ha de molts tipus, però s'han de destacar dues en concret, la màscara Kitsune (Figura 6) i la màscara Oni (Figura 7). La primera representa una de les guineus més famoses de la mitologia japonesa i és típica dels festivals de final d'estiu. Per altra banda, l'altre màscara mostra a uns dels dimonis/monstres (*yokai*) més populars del país.

Figura 6. Màscara Kitsune

Figura 7. Màscara Oni

Per altra part, seguint el amb el tema de les festivitats, un altre element que destaca són els fanalets de paper (Figura 8) que són purament decoratius. Aquests es poden trobar en molts esdeveniments (casaments, funerals, festivals...) i en l'entrada d'alguns bars i tabernes.

Figura 8. Fanalet de paper

Una altra de les figures més populars són les guineus, especialment la que anomenen Kitsune (Figura 9), com s'acaba de comentar. Segons la mitologia japonesa, la seva funció principal és la de protegir els boscos, santuaris i poblats. A més, es creu que posseeix el poder d'adoptar la forma humana d'una dona jove.

Figura 9. Estàtua Kitsune

A part, de les estàtues de guineus, un altre element que es troba majoritàriament en els santuaris són les llanternes de pedra (Figura 10).

Figura 10. Llanterna de pedra

Per altre part i també relacionat amb els santuaris, es troben els arcs torii (Figura 11), una de les estructures més conegudes de l'arquitectura japonesa que, per norma, acostumen a indicar que més endavant es pot trobar un temple, ja que es creu que aquests portals són una frontera entre el profà i el sagrat.

Figura 11. Arc Torii

Per acabar, un dels elements més característics de la primavera al Japó i un atractiu turístic, són els arbres de cirerers en flor, també coneguts com sakura (Figura 12). Aquests són tan populars per la gran quantitat de varietat que hi ha i les celebracions al llarg de l'estació de floració, i la seva simbologia, ja que representa la bellesa i la fugacitat de la vida.

Figura 12. Cirerers

Gràcies a haver fet aquesta recerca, s'han pogut agafar referències i idees per a la creació de l'entorn i alguns elements estètics dels personatges.

4.1.2. Runtime Virtual Texturing

El Runtime Virtual Texturing (RVT) és una eina que inclou l'Unreal Engine. Aquesta eina permet renderitzar materials complexos pel terreny de manera més eficient. La utilització està molt recomanada per a la renderització de materials procedurals i composats per capes. En el cas d'aquest projecte, per a poder usar l'RVT amb la finalitat que es desitja; la qual consisteix en aplicar el mateix color que posseeix el terreny a la vegetació i a una part d'alguns objectes; cal crear diferents components (*volumes*) a l'escena i escalar-los per a que coincideixi amb la grandària del mapa. A més, és necessari generar textures per a cadascun dels volums, ja que aquestes textures contenen la informació del terreny. Un cop s'han creat aquests elements i la informació guardada en ells, ja es poden utilitzar per a crear els materials finals amb els que s'aconsegueix el resultat desitjat.

4.2. Referències

Diferents videojocs han servit com a referència per a la creació de l'estil visual i algunes de les mecàniques del projecte:

- Kena: Bridge of Spirits
  El Kena és un videojoc d'acció-aventura amb alguns puzles en 3D en el que et poses en la pell d'una guia espiritual que ajuda a diferents esperits atrapats en un poblat. Com l'entorn a dissenyar és natural i estilitzat, i tot el món del joc del Kena es situa en zones muntanyoses, rocoses i amb molta vegetació, tal i com es pot apreciar a la Figura 13, ha estat una gran font d'inspiració per agafar idees de com composar l'escena.

Figura 13. Kena: Bridge of Spirits

- The Witness
  The Witness és un joc de puzles en 3D on s'han d'explorar diferents localitzacions i anar resolent els trencaclosques que proposen.
  En aquest cas, no s'ha agafat cap idea de la jugabilitat i mecàniques, si no de la composició de colors i la vegetació que envolten totes les zones del món del joc, com es pot veure a la Figura 14. El contrast i la saturació de la flora és el que s'ha pres com inspiració.

Figura 14. The Witness

- Rime
  Per últim, Rime és un videojoc del gènere d'aventures i puzle, en el que s'han de superar els diferents obstacles per a arribar al punt més alt del mapa.
  De la mateixa manera que els jocs anteriors, Rime ha servit de referència per l'estètica estilitzada, la varietat de vegetació i les estructures rocoses que es troben al llarg del joc, com es mostra a la Figura 15 .

Figura 15. Rime

5. Disseny del videojoc

5.1. Mecàniques

5.1.1. Espai de joc

S'ha definit l'espai de joc com a lineal i bidireccional (Figura 16). El camí a seguir està completament marcat i el jugador no pot accedir a cap zona que no estigui contemplada. Per altra banda, és possible que aquest arribi al final del joc sense l'objecte clau per a poder acabar, així que té la possibilitat de tornar enrere per aconseguir-lo, no hi ha res que li ho impedeixi.

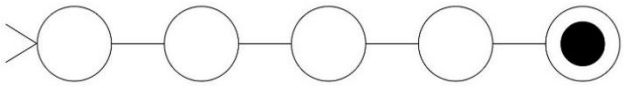

Figura 16. Espai de joc lineal

Per a accedir a segons quin subespai, no en tots els casos, és necessari que el jugador tingui un objecte o compleixi unes condicions. Quan el jugador comença la partida necessita recollir l'arma per a poder continuar pel camí. Un cop l'agafa i desbloqueja la zona, pot decidir lluitar amb el primer grup d'enemics o continuar. Si lluita i guanya, aconsegueix l'objecte necessari per acabar el joc, la pedra màgica. Combatre amb el segon grup, l'ajudarà a recuperar una vida, però no és obligatori per a avançar. Quan es troba en la següent zona, necessita matar a l'últim enemic per a poder activar el puzle que es troba distribuït per tota l'àrea. Un cop el completa de forma correcta, es desbloqueja el pas a la zona final. En aquesta última no hi ha cap enemic, però si el jugador cau a l'aigua perd vides ràpidament. Un cop arriba al punt final, ha d'entregar la pedra obtinguda anteriorment. Si no la té, no pot acabar el joc, i haurà de tornar a la segona zona del joc, arriscant més vides amb l'aigua i el grup d'enemics inicials.

A continuació es mostra el pseudocodi amb les condicions necessàries per a canviar de zona i completar el joc:

```
téArma = fals
téPedra = fals
vida = 5
grupEnemics = [1,2,3]

Mentre vida > 0
Si téArma
      Entra a la nova zona
      Lluita
      Si grupEnemics == 1 i enemicsVius == 0
            téPedra = true
      FSi
      Avança
      Lluita
      Si grupEnemics == 3 i enemicsVius == 0
            Desbloqueja puzle
            Si la seqüència del puzle és correcta
                  Entra zona final
                  Si téPedra
                        Joc acabat exitosament
                  FSi
            FSi
            FSi

Sinó
      Prohibit el pas
FSi
FMentre
```

5.1.2. Reptes i objectius

Tal i com s'ha comentat en la secció anterior i es mostra en el diagrama de la Figura 17, l'objectiu principal és portar la pedra màgica a la zona final. Per aconseguir-ho, el joc obliga al jugador a matar a un grup d'enemics per obtenir-la, matar a un altre grup per a desbloquejar un puzle que bloqueja l'accés; i un cop activat, desxifrat i completat, ja es pot accedir a aquesta última zona esmentada.

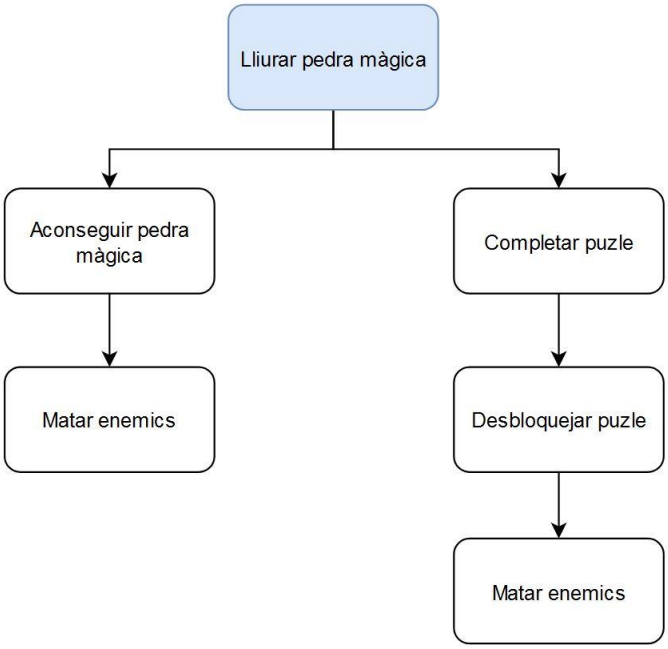

Figura 17. Jerarquia de reptes

5.1.3. Objectes

El jugador pot interaccionar de diferents maneres amb els següents elements que es mostren a continuació:

- El pètal que es mostra a la Figura 18, permet que el jugador recuperi una vida quan aquest passa per sobre.

Figura 18. Pètal

- L'arma (Figura 19) és fonamental per a que al jugador pugui avançar, atacar i bloquejar els atacs dels enemics. Quan es comença a jugar s'ha d'agafar de sobre d'una plataforma que està just al davant d'on s'inicia el joc.

Figura 19. Arma

- Els petits esperits (Figura 20) que es troben al llarg del mapa ajuden al jugador amb pistes. Aquestes es mostren quan el personatge s'apropa a ells.

Figura 20. Esperit ajudant

- L'estàtua de *Kitsune* que es mostra en la Figura 21 és curativa, li retorna tota la vida al jugador. Aquest pot tornar tantes vegades com vulgui per recuperar-se, però només ho pot fer cada 5 minuts. Aquesta estàtua es diferencia de les altres per la brillantor de les seves esquerdes (que desapareixen quan el jugador es cura) i la seva grandària respecte les altres guineus.

Figura 21. Estàtua curativa

- L'objectiu del joc és recuperar la pedra màgica que li falta al Gran Esperit (Figura 22) per a ser alliberat. Aquesta es troba en la segona zona del joc, on està el primer grup d'enemics. Un cop els venç, la pedra es mou fins la posició del jugador. Quan col·lisionen, la pedra es destrueix i, internament, s'indica que l'ha aconseguit i la té en el seu poder. Això es veu reflectit a la interfície d'usuari.

Figura 22. Pedra màgica

- La peça del puzle que es mostra en la Figura 23 es troba just abans de la zona final, junt amb altres 5 peces més quasi idèntiques. Inicialment no es pot resoldre, primer s'ha de vèncer l'enemic, després, un cop desbloquejades (Figura 24), s'han d'accionar en l'ordre correcte; que està indicat pel número de flors que tenen sobre. Per a fer-ho possible, el jugador ha de colpejar-les. Segons es vagi encertant l'ordre, les esferes blaves s'il·luminen i eleven (Figura 25).

Figura 23. Puzle bloquejat

Figura 24. Puzle desbloquejat

Figura 25. Puzle activat

- El portal és l'entrada a la zona final: fins que no es completa el puzle correctament, aquest no deixa passar al jugador. Quan s'intenta, es mostra un efecte que indica que no es pot continuar (Figura 27), en canvi, just s'encerta amb l'última peça del puzle, la boira que es veu a la Figura 26 desapareix i es mostra un altre efecte que indica que ja es pot continuar (Figura 28).

Figura 26. Portal bloquejat

Figura 27. Portal amb efecte de bloqueig

Figura 28. Portal desbloquejat

- Les pedres amb forma cilíndrica es troben a la zona final del joc formant un camí a través de l'aigua. Algunes d'aquestes tenen esquerdes, com es pot veure a la Figura 29, que és un indicador de que desapareixen als pocs segons d'haver-les trepitjat. Això fa que el jugador caigui a l'aigua i perdi vida.

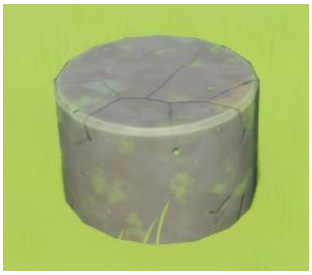

Figura 29. Pedra inestable zona final

- En la Figura 30 es pot veure la plataforma final on es troba el Gran Esperit. Quan el jugador s'apropa, pot interactuar amb ell. Sí té la pedra màgica, es col·loca on li correspon i s'acaba el joc, i sinó, es mostra un missatge indicant que ha d'anar a buscar-la.

Figura 30. Plataforma del Gran Esperit

5.1.4. Accions

Les accions que pot dur a terme el jugador són les següents:

- Atacar: permet al jugador vèncer als enemics i resoldre el puzle (s'han d'accionar diferents peces).
- Bloquejar: dona la possibilitat de bloquejar els projectils dels enemics.
- Córrer: facilita l'esquiva i el moviment per l'entorn.
- Saltar: permet al jugador no caure a l'aigua en la zona final, saltant d'una roca a una altra.
- Interactuar amb l'entorn
    - Agafar l'arma: a l'inici del joc, el jugador pot interactuar amb l'arma per a obtindre-la i ser capaç de, posteriorment, lluitar.
    - Recuperar vida: a través d'alguns objectes, el jugador por recuperar una o més vides.
    - Col·locar pedra màgica: en la zona final del joc, es pot entregar la pedra màgica, en cas que s'hagi recuperat.

5.1.5. Economia

L'economia d'un videojoc pot estar composada per diferents elements que generen, modifiquen i consumeixen recursos, aquests es classifiquen en quatre tipus:

- ***Sources***: Creen recursos del no res i dels guarden en una determinada entitat.
- ***Drains***: Consumeixen recursos.
- ***Converters***: Transformes recursos d'un tipus en un altre.
- ***Traders***: Mouen el recurs d'una entitat a una altra.

En el joc es poden trobar dos *sources* que proporcionen vida. La vida la pot donar algun grup d'enemics al morir, o amb una estàtua curativa. Per altra part, també hi ha dos elements *drains*, un és l'atac de l'enemic i l'altre l'aigua que es troba en la zona final, ambdós tenen la funció de drenar la vida del jugador.

A continuació es mostra un diagrama d'aquesta economia:

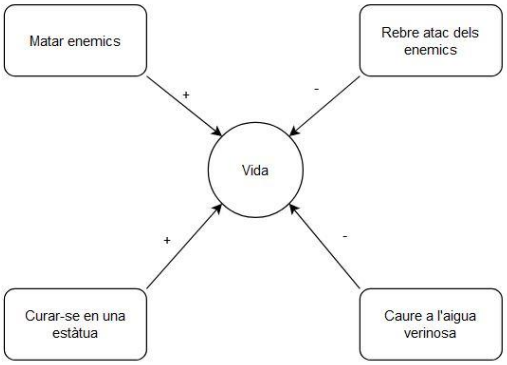

Figura 31. Diagrama de l'economia

5.1.6. Game balancing

El disseny del combat, tot i ser senzill, s'ha de balancejar, ja que el jugador no té manera de millorar l'atac de l'arma o l'àrea de protecció. L'únic que millora és l'experiència del jugador en combat. Això sí, quan fa una de les dues accions i colpeja als enemics, aquests es desplacen lluny del personatge, per donar un avantatge al jugador. Pel que fa als enemics, aquests es divideixen en tres grups. El primer consta de 3 enemics, és el més gran, però només tenen 2 punts de vida, així que moren fàcilment. El segon grup l'integren 2 fantasmes amb 3 punts de vida cadascun, aquests són més lents a l'hora d'atacar, i, tot i que es van transportant a altres posicions al voltant del jugador, dona el suficient temps a reaccionar. Per últim, el tercer grup només inclou 1 fantasma que té 5 vides i s'assembla molt als del primer grup, però els atacs són dirigits, així que moltes vegades esquiven el bloqueig.

5.2. Narrativa

5.2.1. Sinopsi

El jugador es posa en la pell de la Hana, una noia que desperta en un bosc infestat d'enemics fantasma. Normalment, es troba protegit pel Gran Esperit, però cada cert temps, aquest necessita descansar per a recuperar el poder que ha gastat al llarg d'un gran període. Cada vegada que això succeeix, s'escampen per tot el bosc tres pedres que formen part de l'esperit i que li permeten fer servir tot el seu poder.

Els enemics que es mantenien fora del bosc, gràcies a la protecció que aquest rebia, aprofiten per a tornar a prendre el control de la zona i expulsar als petits habitants que hi viuen. Mentre l'esperit està en repòs, el bosc es va debilitant, fins al punt en que pot arribar a morir.

És per això que el bosc necessita l'ajuda de la Hana per a recuperar les pedres màgiques que obren l'esfera on es troba el Gran Esperit. Un cop les tingui totes, serà capaç d'alliberar-se i atorgar pau i protecció durant un altre llarg període.

5.2.2. Personatges

Hana

La Hana (Figura 32) és la protagonista del joc, el seu nom significa flor en japonès. Ella és una noia jove, d'estatura mitjana (1.66m), cabells curts i blaus. Vesteix amb un vestit amb estampat de flors de cirerers i un llaç al voltant de la cintura. Porta una màscara de guineu que li tapa el rostre i unes branques amb flors li envolten el cos.

Figura 32. Hana

Fantames

Els fantasmes (Figura 33) no tenen nom i són relativament petits, mesuren uns 0.7m. Tots porten màscares i els rodegen focs que utilitzen com a projectils per a atacar a qualsevol ésser viu. Els fantasmes són de diferents colors, segons la perillositat.

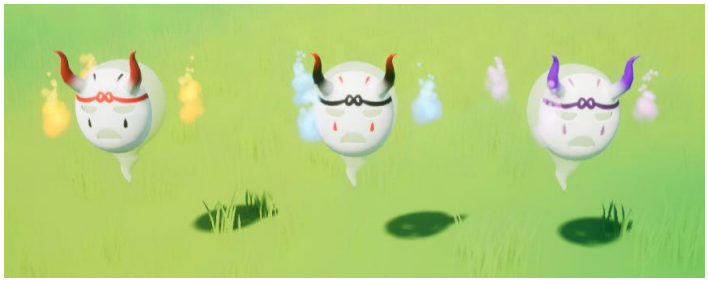

Figura 33. Fantasmes

Gran Esperit

El Gran Esperit (Figura 34) és blanc i brillant, a més li surten tres plantes de diferents colors del cap. Mesura al voltant del mig metre, i tot i que sembli poca cosa, conté un poder enorme. Es diu que fa centenars d'anys, ell també era un petit esperit.

Figura 34. Gran Esperit

Petits esperits

Els esperits que es mostren a la Figura 35 són els habitants del bosc i ajudants del Gran Esperit. Són diminuts, però no tant brillants. Tenen forma de llavors i els hi surt una branqueta amb fulles de la punta del cap.

Figura 35. Esperits

5.3. Estètica

5.3.1. Paleta de colors

Com s'ha decidit crear un entorn natural amb elements orientals, la paleta de colors és la següent:

Figura 36. Paleta de colors

El colors principals que es troben a l'entorn són els verdosos i groguencs, per la vegetació, i els grisos per les roques. El vermell contrasta molt amb aquest tons, a més de ser un color predominant en molts objectes tradicionals. Per altra part, el rosa també juga un paper important, per la representació dels arbres de cirerers i les seves flors.

5.3.2. Procés de modelat i texturitzat

Per a poder explicar de forma resumida el procés del modelat amb *Blender* i posterior texturitzat amb *Substance Painter*, s'ha escollit mostrar un dels objectes creats.

Aquest model s'ha creat a partir de diferents formes bàsiques, cilindres, cubs, plans, cercles... Com es pot veure a la Figura 37, un cop s'aconsegueix que les diferents malles representin l'objecte desitjat, es pot dir que s'ha obtingut el model *lowpoly,* és a dir, que compta amb un número reduït de polígons.

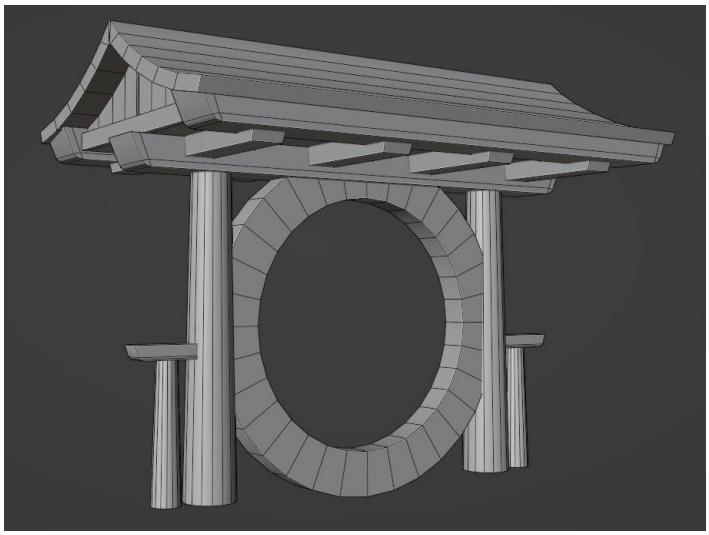

Figura 37. Model lowpoly

A continuació, es duplica l'objecte i se li aplica el modificador *subdivision* per a que subdivideixi la malla en molts polígons, el que s'anomena model *highpoly* (Figura 38). Seguidament, també s'afegeix el modificador *remesh* que refà la malla amb una distribució dels polígons més uniforme. Tot això fa possible que es pugui modelar i esculpir amb més detall.

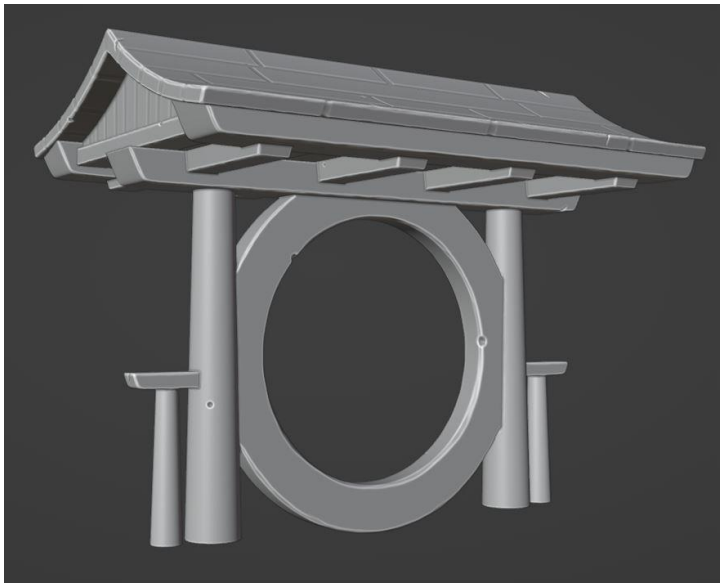

Figura 38. Model highpoly

Abans d'exportar els models, s'han de crear les *UVs*, és a dir, desplegar la malla en un pla 2D, on es projecta cada polígon, això permet el texturitzat i la creació dels diferents materials. Per a aquest model en concret, s'han creat dos, un per l'estructura (Figura 39) i l'altre pel teulat (Figura 40).

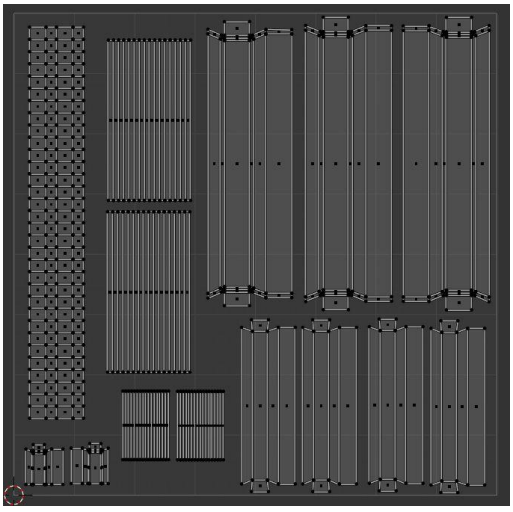

Figura 39. UV de l'estructura

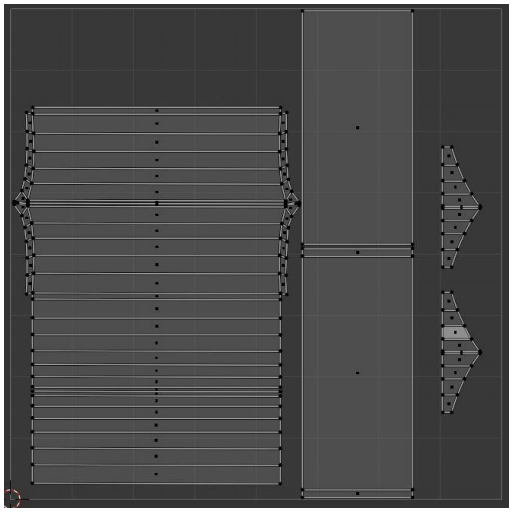

Figura 40. UV del teulat

A continuació, tant el model *lowpoly* com el *highpoly* es porten al programa de texturitzat, en aquest cas, *Substance Painter*. Des d'aquí es fa el *bake* del model (Figura 41), el que vol dir que, utilitzant el model amb pocs polígons com a base, se li apliquen els detalls del que té un alt grau de poligonatge per a que, visualment, sembli que té una major qualitat.

Figura 41. Bake

Un cop que el *bake* s'ha acabat satisfactòriament, es procedeix a texturitzar l'objecte (Figura 42). Per fer-ho es defineixen diferents capes (Figura 43) en les que s'aplica una gran varietat de colors i textures.

Figura 42. Model texturitzat

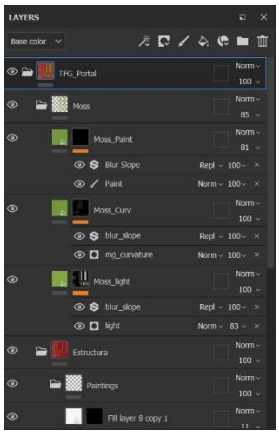

Figura 43. Capes

Tot i que aquest és el procediment estàndard utilitzat en el modelat, cada objecte és diferent i a vegades s'han d'utilitzar altres modificadors. A més, per a models que, per exemple, simulen roques, és més fàcil fer el tractament invers. Primer, crear un model *highpoly* a partir d'una esfera o cub, esculpir-lo i, finalment, amb un modificador anomenat *decimate*, que s'encarrega de reduir el nombre de polígons, ja s'extreu el *lowpoly*.

5.3.3. Disseny de personatges

Personatge

Abans de començar a modelar el personatge, tot i tenir una idea bastant clara del que es volia, es van generar uns esbossos previs, com els que es poden veure a la Figura 44 i Figura 45.

Figura 44. Primer esbós

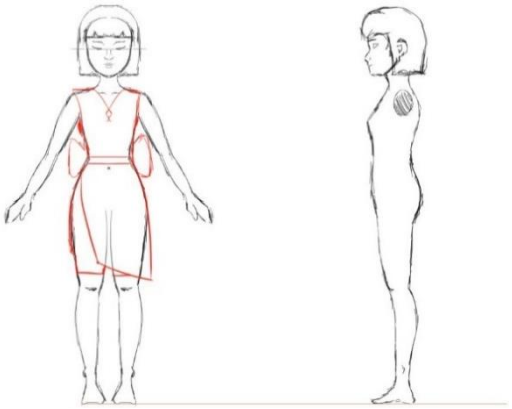

Figura 45. Segon esbós

Com es pot apreciar en la Figura 46, el resultat final ha estat una barreja de les dues propostes inicials. El cos s'ha modelat seguint el segon esbós; la roba s'ha mantingut en els dos, però canviant la cinta del voltant de la cintura, i la màscara, tot i que no surt en la segona proposta, sempre ha estat present, encara que a l'hora de modelar-la el resultat no va ser l'esperat i es va decidir canviar el disseny.

Figura 46. Model del personatge

A part del modelat, també s'ha fet el *rig* (l'esquelet del personatge) com es pot veure a la Figura 47, gràcies a *l'addon AutoRigPro*, desenvolupada per a *Blender*, que ha facilitat la feina. Tot i així, s'han hagut de modificar les *weights* de cada os, és a dir, la influència que n'exerceix cadascun sobre la malla.

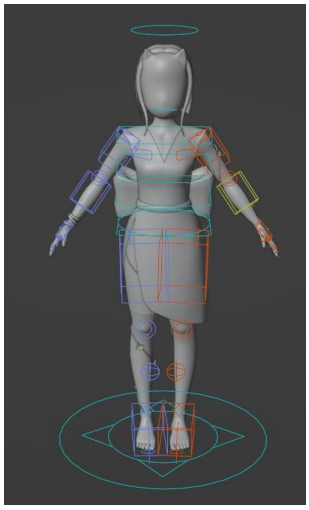

Figura 47. Rig personatge

A part, s'han extret diverses animacions de *Mixamo*; com les de caminar, atacar i saltar; que s'han hagut de modificar per a que es comportessin correctament amb el model. A més, s'ha creat la de bloquejar, ja que no s'ha trobat cap que encaixés amb el que es volia.

Enemics

Per a dissenyar els enemics i fer un primer esbós (Figura 48), s'ha fet una breu recerca sobre els diferents monstres més populars de la cultura japonesa i s'ha decidit optar pel fantasma. Tot i que aquests no es veuen representats amb màscares, es creu que ha estat una bona decisió de disseny per a complementar la que també porta el personatge.

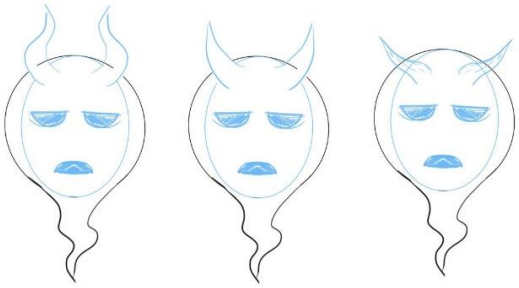

Figura 48. Esbós enemics

Tal i com es veu en la Figura 49, els enemics segueixen l'esbós anterior. A més s'ha decidit optar pel primer disseny de banyes de les màscares, ja que s'assemblen més a la forma que tenen les màscares Oni, mencionades a l'Apartat 4.1. Cadascun té un color per a indicar al jugador que el seu comportament és diferent al dels altres.

Figura 49. Enemics

En aquest cas, el *rig* dels enemics sí s'ha fet a mà, ja que l'*addon* esmentada en la secció anterior només serveix per a models amb forma humana. Tot i així, com no es volien generar animacions complicades (només mou la cua) s'ha fet un esquelet bàsic (Figura 50).

Figura 50. Rig enemic

5.3.4. Disseny d'objectes

Per a la composició i disseny de l'entorn s'han modelat diferents objectes, alguns són només decoratius, i altres compleixen alguna funció i interactuen amb el jugador, com els que s'han comentat en la Secció 5.1.3. Cap d'aquests models s'ha extret de fonts externes, tots són de creació pròpia.

Objectes

A continuació, es mostren tots els objectes modelats (alguns es composen de varis més bàsics), excepte alguns dels que s'han vist en seccions anteriors:

- L'arma del personatge (Figura 51 i Figura 52) és un element molt característic d'aquest. Es creu que, una altra arma com una espasa, llança, o qualsevol altre objecte, no encaixaria tant com el típic para-sol. A més, s'ha escollit el color vermell perquè es creu que a part de contrastar amb els colors de l'entorn, ja es troben altres elements tradicionals en els que el color més típic és el vermell. D'aquesta manera es crea una coherència entre tots els elements.

Figura 51. Paraigües tancat

Figura 52. Paraigües obert

Per a que l'arma es pugui obrir a l'hora de bloquejar, tal i com s'ha fet amb l'enemic, s'ha creat un rig personalitzat. Com es pot veure a la Figura 53, ha estat necessari crear un os per cada vareta, més algun altre pel pal central i l'estructura interna.

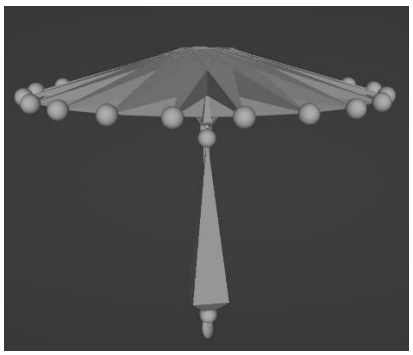

Figura 53. Rig arma

- El pètal amb el que es recupera vida es mostra a la Figura 54. S'ha escollit aquest element, ja que es representa amb una flor de cirerer.

Figura 54. Pètal

- Les pedres màgiques i la plataforma on es troba el Gran Esperit es poden veure a la Figura 55. Aquests elements s'acaben integrant en un de sol, tal i com s'ha vist en apartats anteriors.

Figura 55. Pedres màgiques i Plataforma final

- El Gran Esperit i els ajudants (Figura 56). El primer es troba en la plataforma anterior i els altres esperits per tot el bosc.

Figura 56. Gran Esperit i petits esperits

- Fanal, espelma i el conjunt (Figura 57). Com la flama de l'espelma és molt petita, s'han decidit crear unes textures en blanc i negre, que s'anomenen *alphas* (Figura 58), enlloc d'un sistema de partícules que és més costós. Amb aquestes textures es crea un material que s'aplica a un pla, això s'explica en profunditat en l'Apartat 6.8.5.

Figura 57. Fanalets i els elements que el composen

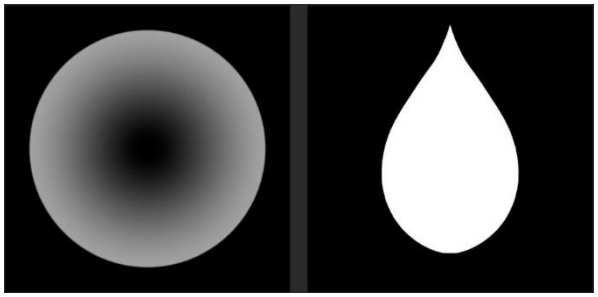

Figura 58. Alpha per la flama de l'espelma

- Kitsunes i pedestals (Figura 59). Aquestes guineus són molt populars en la cultura japonesa, tal i com s'ha explicat en l'Apartat 4.1.

Figura 59. Kitsunes i pedestals

- Estàtua d'unes mans (Figura 60). A sobre d'aquestes es troba la pedra màgica que el jugador ha de recuperar.

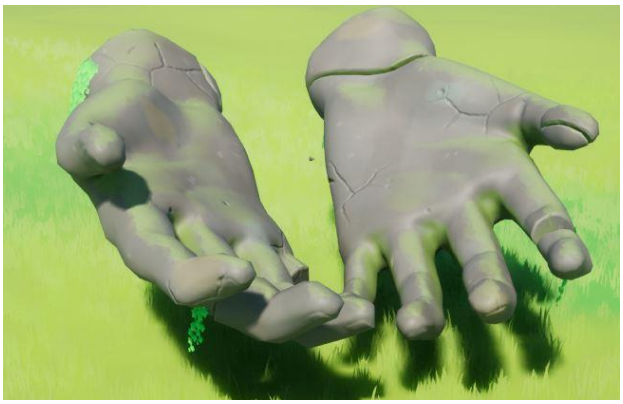

Figura 60. Estàtua de mans

- Diferents arcs torii i el portal (Figura 61) que bloqueja el pas, en el que també s'han utilitzat uns colors i estructures semblants per a poder integrar-ho en l'entorn.

Figura 61. Arcs torii i portal

- Pont amb una estètica semblant als elements anteriors (Figura 62).

Figura 62. Pont

- Altres peces que formen part de l'entorn, com les pedres que composen el camí fins a arriba al Gran Esperit, un dels elements del puzle i la plataforma on es troba l'arma al principi del joc (Figura 63).

Figura 63. Pedres zona final, element del puzle, plataforma inicial

- Altres elements decoratius (Figura 64) que es troben per tot el mapa. Ajuden a que aquest no es vegi tant buit, a més, es complementen amb l'estètica general de l'entorn.

Figura 64. Altres elements decoratius: sabates, gerres, fanals, columnes, pedres i tronc

- Les roques (Figura 65) són una part fonamental de qualsevol entorn natural, però, a més, han estat de gran ajuda per a marcar els límits del mapa i el camí que ha de seguir el jugador.

Figura 65. Roques

Vegetació

- Per a la gespa (Figura 66) s'han modificat diferents plans per a que prenguessin la forma desitjada. No s'ha texturitzat com qualsevol altre model perquè aquest agafa el color del terreny indicant-ho en un material creat a *l'Unreal.* Això s'explica més endavant en l'Apartat 6.7.2.

Figura 66. Gespa

- Per al modelat dels arbres i arbustos (Figura 67) s'ha utilitzat el programa *Tree It*, amb el que es pot generar fàcilment el model i aplicar a cada branca la quantitat de fulles que es vol. Per a aquestes, s'han creat diferents *alphas* (Figura 68), així es dona més varietat entre els diferents elements. Tot i tenir aquesta base generada pel programa, s'ha refet el tronc i les branques al Blender, seguint la forma del primer. Així s'ha aconseguit un model més detallat, però que segueix encaixant amb la distribució de les fulles.

Figura 67. Arbres i arbustos

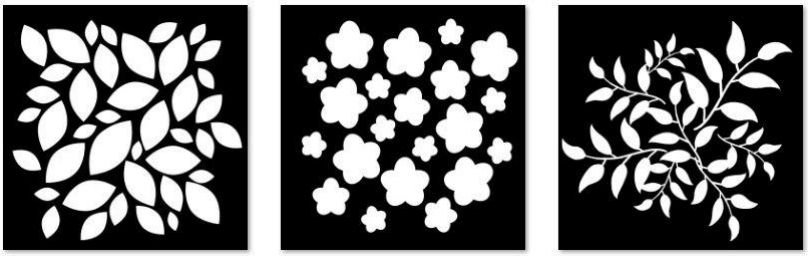

Figura 68. Alphas dels arbres i arbustos

- Pel que fa a la generació de les plantes i les flors que s'han utilitzat per a donar-li color a l'entorn i decorar alguns dels models, primer de tot s'han dibuixat els diferents elements en textures en *Photoshop,* anomenades atles (Figura 69). Aquestes són molt útils, ja que d'una se'n poden extreure diversos models, no cal crear una sola textura per a cadascuna. Un cop generades, des del Blender es creen, a partir d'un pla, la forma de les fulles o flors, a les que se li assigna un espai UV de la textura en qüestió. Amb això ja es poden generar les diferents composicions (Figura 70) que s'importen, posteriorment, al motor.

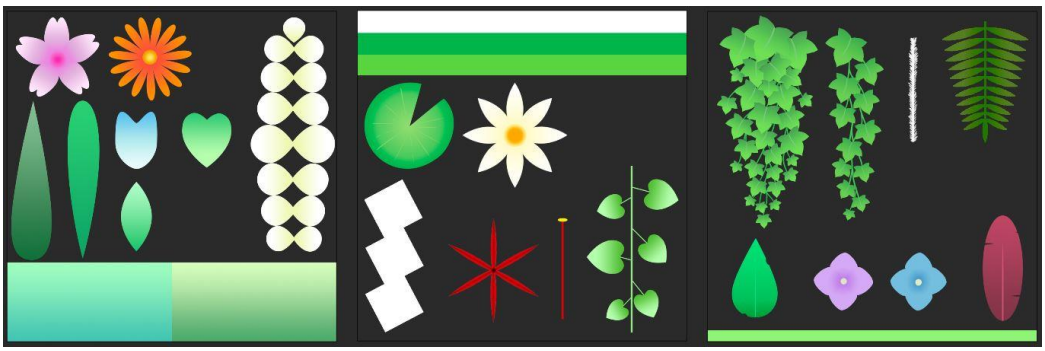

Figura 69. Atles de plantes, flors i altres detalls

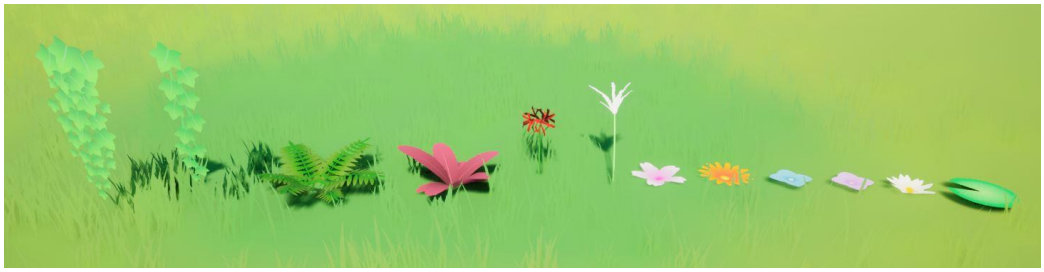

Figura 70. Plantes i flors

5.3.5. Sistemes de partícules

Gràcies al sistema de partícules d'Unreal, tant el bàsic com el Niagara, ha fet possible crear diferents efectes visuals per a alguns models, o per a acabar d'omplir algunes zones de l'entorn. A continuació, es mostren els efectes creats:

- Les flames són un element molt important pels enemics, ja que les utilitzen com a projectils. Com es pot veure a la Figura 71, n'hi ha de tres colors diferents.

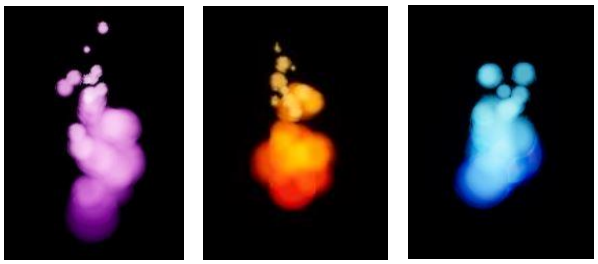

Figura 71. Flames

- La Figura 72 mostra l'efecte que veu el jugador quan vol passar pel portal bloquejat i quan l'ha desbloquejat. S'han creat dos *alphes* per als materials d'aquestes partícules (Figura 73).

Figura 72. Efectes portal

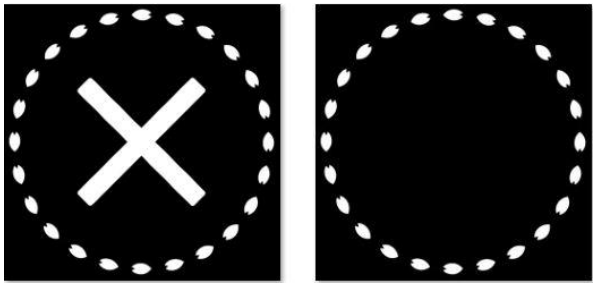

Figura 73. Alpha del portal

- Como que tot l'entorn es situa en un bosc, s'ha decidit crear unes partícules en forma de fulles (Figura 74) que es veuen en algunes zones amb arbres. Aquest cas també ha necessitat un alpha (Figura 75).

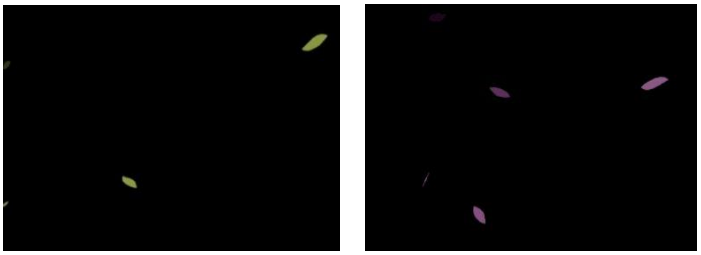

Figura 74. Fulles

Figura 75. Alpha fulla

- S'han afegit cuques de llum (Figura 76) sobre algunes zones amb aigua.

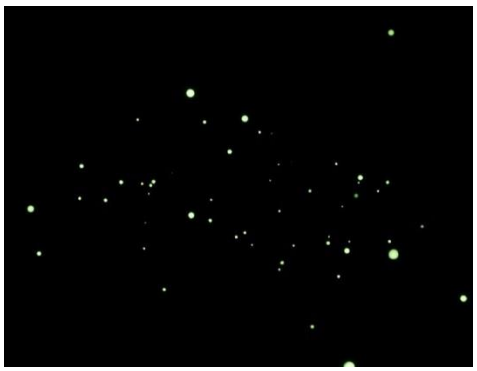

Figura 76. Cuques de llum

- En alguns models del joc, com les pedres màgiques i el pètal, se'ls hi ha incorporat unes petites espurnes (Figura 77) del color de l'objecte.

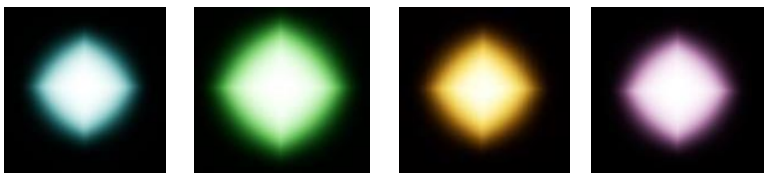

Figura 77. Espurnes

5.3.6. Il·luminació

La il·luminació ha estat un dels apartats més complicats de l'estètica. Al començament del projecte no es tenia clar en quin moment del dia s'ambientaria, ni quines llums s'utilitzarien a l'escena. Finalment, s'ha decidit que, per tal de destacar els colors de la vegetació i els diversos elements, fos de dia.

Per altra part, tal i com es comenta a l'Aparat 9, com a futura millora, s'havia pensat en canviar el moment del dia segons el jugador avança en el joc, d'aquesta manera, començar al matí i acabar de nit. És per això que, en els elements com la plataforma on es troba l'arma, els fanals (els normals i els de paper), l'estàtua curativa i el pont, s'han inclòs diferents punts o focus de llum. Tot i que de dia no s'acaben d'apreciar, també aporten un toc de lluminositat a l'element. A més, aquests punts de llum s'han integrat en *blueprints* pels fanals i el pont, ja que els elements consten de més d'una llum o s'instancien vàries vegades. Així es facilita el treball, pel fet que tots aquests punts es canvien a la vegada.

A la Figura 78, Figura 79, Figura 80, Figura 81 i Figura 82 es poden veure els diferents elements amb les seves llums associades.

Figura 78. Llum de la plataforma inicial

Figura 79. Llum del fanal de pedra

Figura 80. Llums del pont

Figura 81. Llum de l'estàtua

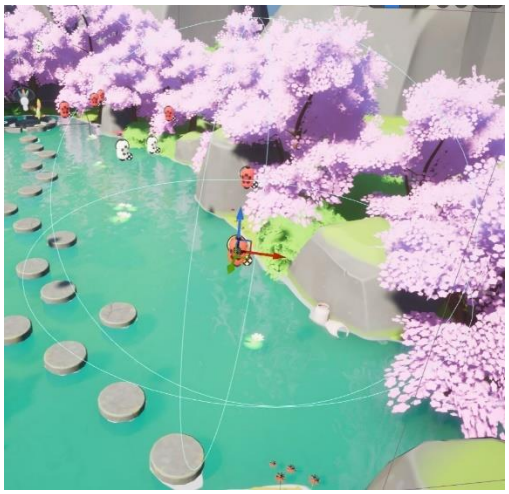

Figura 82. Llum del fanal de paper

5.4. Interfícies

Pel disseny de les interfícies s'han escollit colors i formes que es creu que encaixen i són coherents amb la temàtica oriental. Tots els elements s'han creat utilitzant programes com Clip Studio i Photoshop. No obstant això, s'han extret alguns recursos com la tipografia, textures i pinzells de fonts externes.

5.4.1. Menús

El menú principal (Figura 83) consta de tres opcions: *Start*, per a començar la partida; controls, per consultar els controls tant del teclat com del comandament, i *Quit*, per sortir del joc. Pel que fa al disseny, la imatge de fons s'ha extret generant una composició de diferents elements del joc.

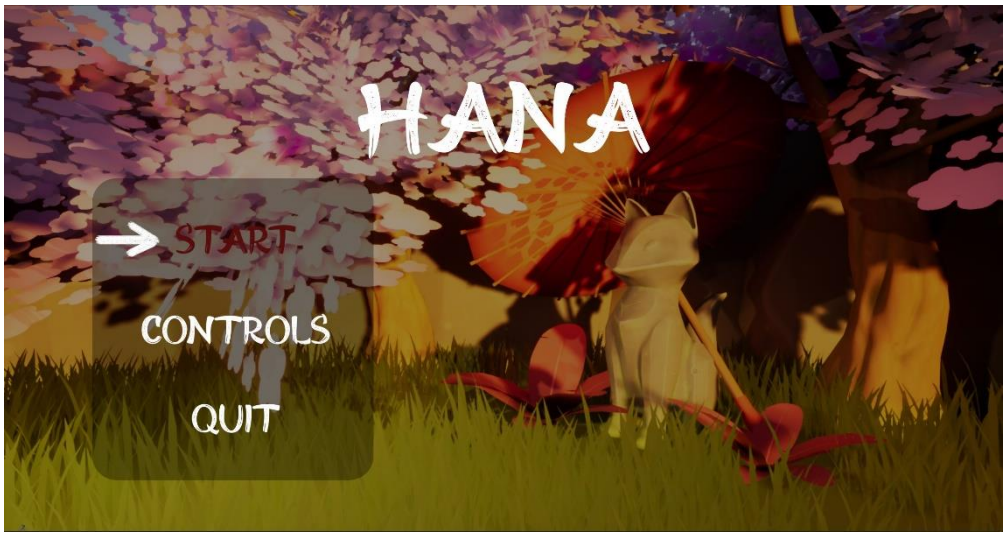

Figura 83. Menú principal

Un cop iniciat el joc, aquest es pot posar en pausa sempre que es vulgui, prement Esc del teclat o el botó de menú del comandament d'Xbox. En aquesta pantalla (Figura 84) es troben diferents opcions, per a continuar jugant, *Resume*; per consultar els controls, *Controls*; i per tornar al menú inicial, *Return to menu*.

Figura 84. Menú de pausa

Com s'ha comentat, els dos menús anteriors consten de l'opció de consultar els controls del joc. Tal i com es pot veure a la Figura 85, aquesta pantalla mostra un llistat de les diferents accions disponibles, tant per teclat com per comandament.

Figura 85. Pantalla controls

Per altra banda, també s'ha dissenyat un menú (Figura 86) per a quan el jugador perd totes les vides, se li permet reiniciar o tornar al menú principal.

Figura 86. Menú game over

Finalment, l'últim menú és el del final del joc (Figura 87), quan s'assoleix l'objectiu. Des d'aquí es dona l'opció de tornar al menú principal.

Figura 87. Menú final del joc

5.4.2. Interfícies d'usuari

La pantalla del joc és senzilla, conté els elements necessaris per a indicar al jugador l'estat del personatge. A la cantonada superior esquerra es mostra la vida actual, es comença amb un total de cinc, com es pot veure a la Figura 88. Cada pètal representa una de les vides disponibles (Figura 89) que, com s'ha comentat en la Secció 0, es poden recuperar de diverses maneres.

Figura 88. Vida completa

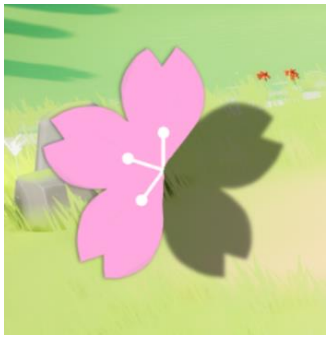

Figura 89. Tres vides

Per altra part, quan el jugador aconsegueix l'objecte necessari per a completar l'objectiu, se li indica amb una imatge d'aquest (Figura 90) a la cantonada superior dreta de la pantalla.

Figura 90. Pedra màgica

Finalment, tot i que no formen part de manera constant a la pantalla del joc, també s'han inclòs missatges, com es pot veure a la Figura 91, que proporcionen ajuda. Aquests missatges es mostren al centre de la part inferior de la pantalla.

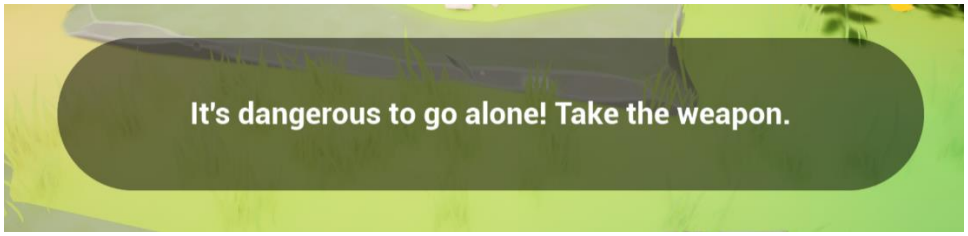

Figura 91. Missatge d'ajuda

5.5. Diagrama global del joc

El següent diagrama (Figura 92) mostra una visió global del joc, on es pot veure com es comuniquen les diferents escenes i menús del joc entre si, incloses les opcions disponibles en aquests últims.

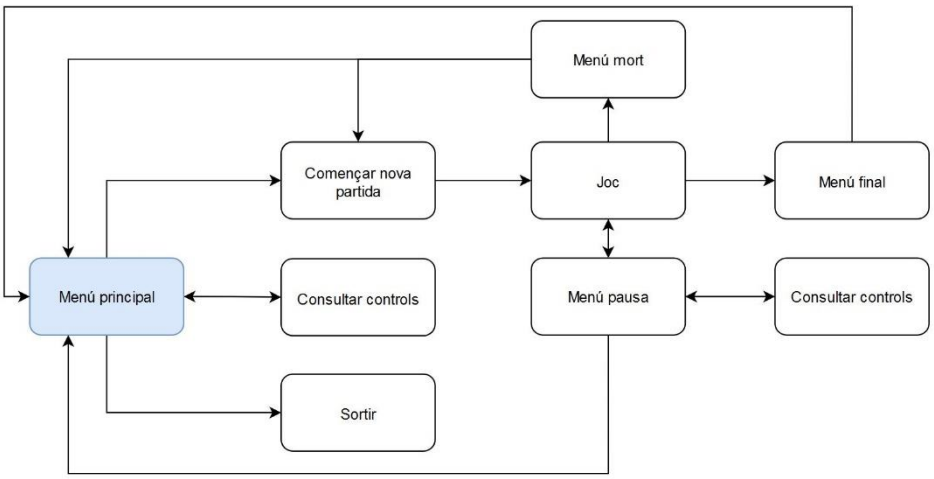

Figura 92. Diagrama global del joc

6. Implementació

6.1. Glossari

A continuació, es mostren diferents conceptes d'Unreal que són útils per a la comprensió de la implementació que s'explica més endavant:

- Mapa: nivell/escena del joc
- *GameMode*: defineix una sèrie de paràmetres i regles per al correcte funcionament del mapa en qüestió.
- *PlayerController*: interfície entre el jugador i el personatge (inputs, comportament...).
- *Blueprint*: script visual basat en grafs de nodes.
- *Blueprint Interface*: es tracta d'una interfície, és a dir, no consta d'una implementació pròpia, sinó que aquesta es defineix en els diferents actors que la criden.
- *Animation Blueprint*: gestiona les animacions de moviment bàsiques del personatge.
- *Actor*: objectes que tenen un comportament implementat.
- *Timeline*: és un node que al que fa referència a una animació que és possible crear des de *l'Unreal*.
- *Delay*: atura l'execució del graf en qüestió durant els segons indicats.
- *Function*: són les funcions que es creen en els *blueprint*. Al contrari que els esdeveniments, a part de tenir valors d'entrada, també poden retornar valors, però no poden utilitzar *delays* ni *timelines*.
- Esdeveniment: són esdeveniments que es creen en els *blueprint.* Poden rebre valors d'entrada, però no de sortida. Això sí, com es creen en el graf principal, són capaços de fer servir *delays* i *timelines.*
- *Event Dispatcher*: *trigger* que s'executa a la vegada en tots els llocs on es pot rebre aquest tipus de peticions.
- *Widget Blueprint*: *blueprint* específic per a interfícies d'usuari.
- *InputAction*: són tots els botons i tecles disponibles que pot prémer el jugador per a executar una acció.
- *Viewport*: pantalla
- *Socket*: son punts que es creen en un os de l'esquelet per a poder accedir a la posició d'aquest des del *blueprint*.

6.2. Mapes

El joc només consta de dos mapes: el *MainMenuMap*, que mostra el menú principal i les seves opcions, i el *MainMap*, on es desenvolupa tota la partida, ja que no consta de més nivells.

Per a cadascun es defineix un *Game Mode*, el *MenuGameMode* i el *BP_ThirdPersonGameMode* respectivament. Ambdós comparteixen el mateix *PlayerController*, que en aquest cas s'ha anomenat *CharacterController*. La diferència entre ells, és que, a part d'estar relacionats a un mapa diferent, el *MenuGameMode* no consta d'un *DefaultPawnClass,* tal i com es pot veure a la Figura 93, ja que en el menú

d'inici no hi ha cap personatge que es pugui controlar. En canvi, l'altre sí que té assignat el *blueprint* del personatge (Figura 94).

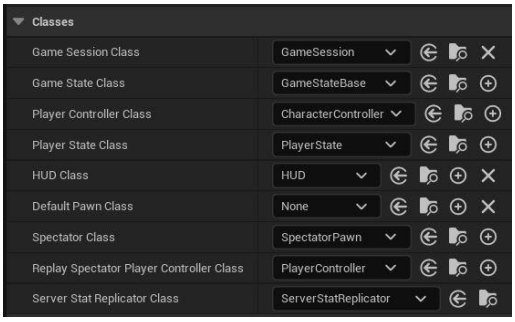

Figura 93. MenuGameMode

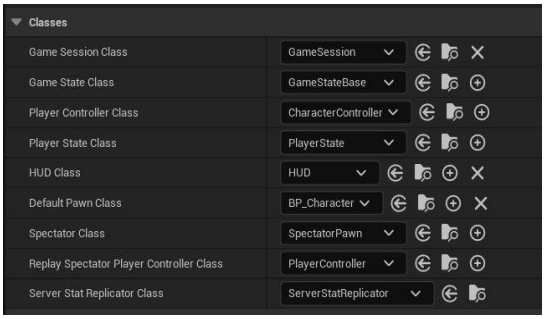

Figura 94. ThirdPersonGameMode

6.3. Personatge

La implementació del personatge es troba al *blueprint BP_Character*. Aquest es basa en el *BP_ThirdPersonCharacter* que ja ve per defecte a l'hora de crear el projecte. A continuació, com es pot veure a la Figura 95, es mostra una visió general del *Blueprint* que s'explica en detall en els següents apartats.

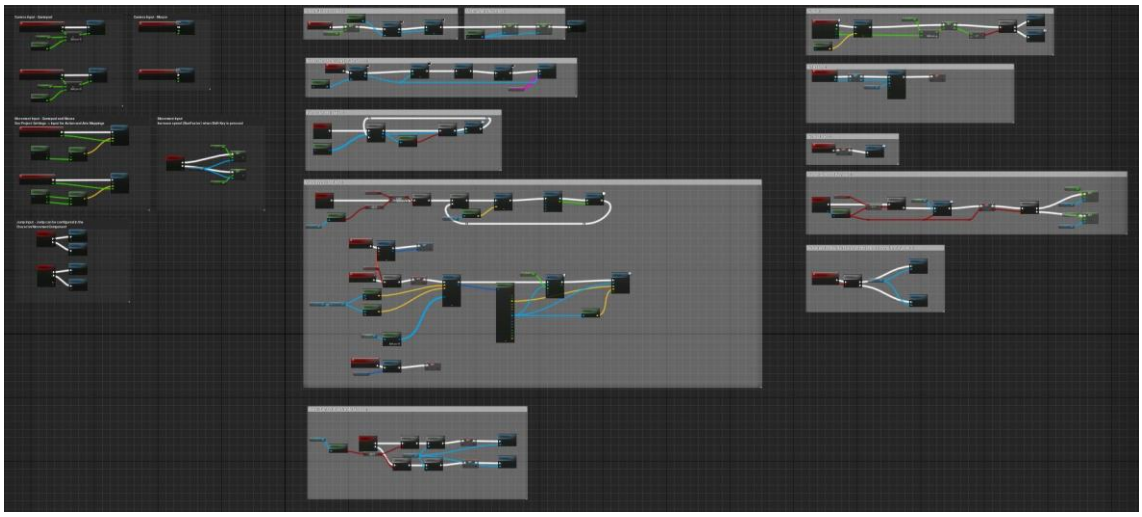

Figura 95. BP_Character

6.3.1. Moviment del jugador

El moviment i la càmera del jugador no s'han modificat en excés respecte el que ja estava implementat per defecte. Pel que fa a la càmera, aquesta consta de quatre esdeveniments que permeten moure-la en totes les direccions, tant amb el ratolí com amb el comandament, com es pot apreciar a la Figura 96.

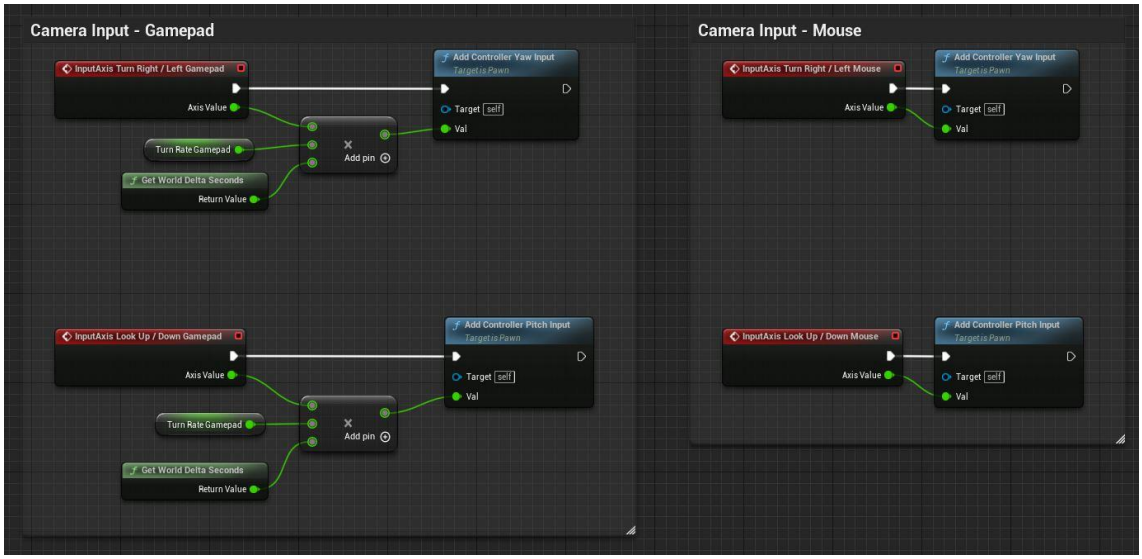

Figura 96. Càmera del jugador

En el cas del moviment passa el mateix, els esdeveniments que es mostren a la Figura 97, excepte l'*InputAction Run,* són els predeterminats per a controlar el personatge en tercera persona. L'altre que s'acaba de mencionar, s'ha afegit per a permetre que el jugador pugui córrer, canviant el valor del paràmetre *MaxWalkSpeed* per *RunSpeed*, al qual se li ha assignat un valor superior al de la variable *Speed*. Per a iniciar l'acció de córrer, el jugador ha de mantenir premut el *Shift* del teclat o moure el *joystick* esquerre del comandament al màxim.

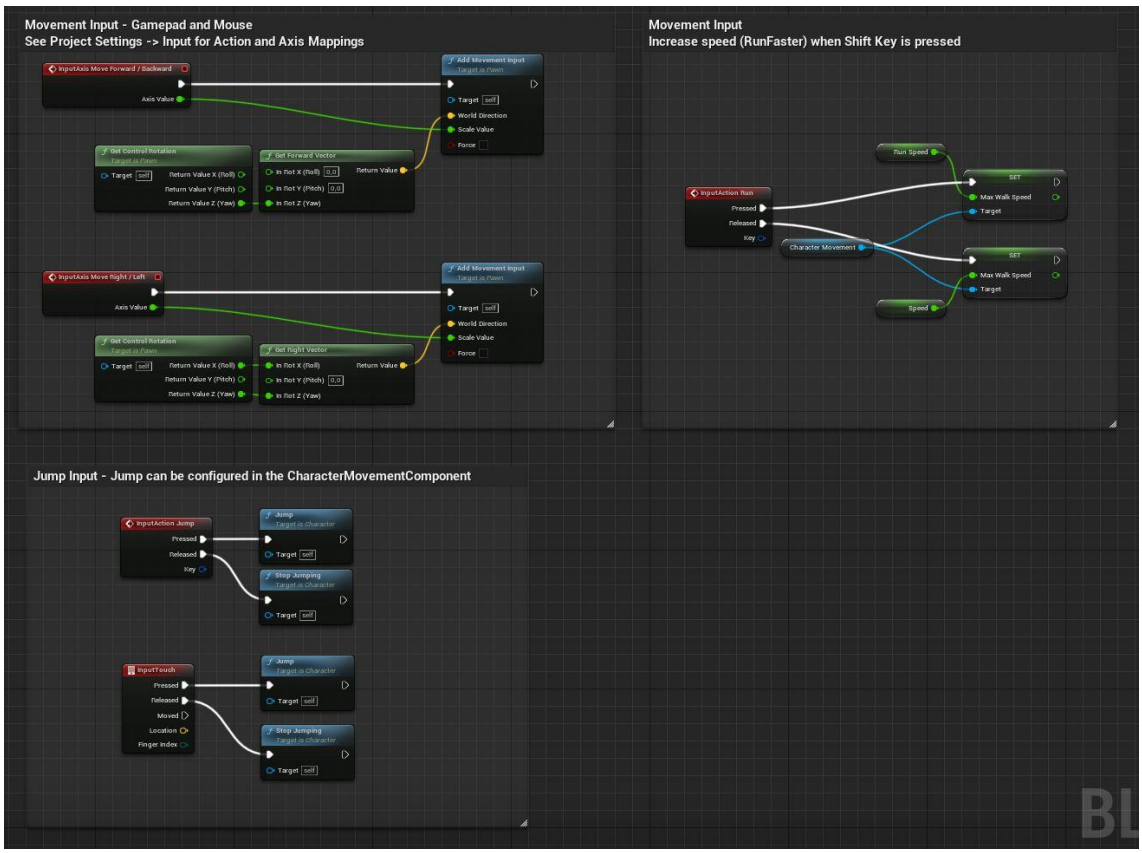

Figura 97. Moviment del jugador

A més, s'ha creat l'esdeveniment *ChangeMovement* (Figura 98), en el qual s'activa o desactiva el moviment del personatge segons convingui. Aquest es crida en ocasions com a l'inici del joc, on es posa en context al jugador, i al final, on s'entrega l'objecte clau.

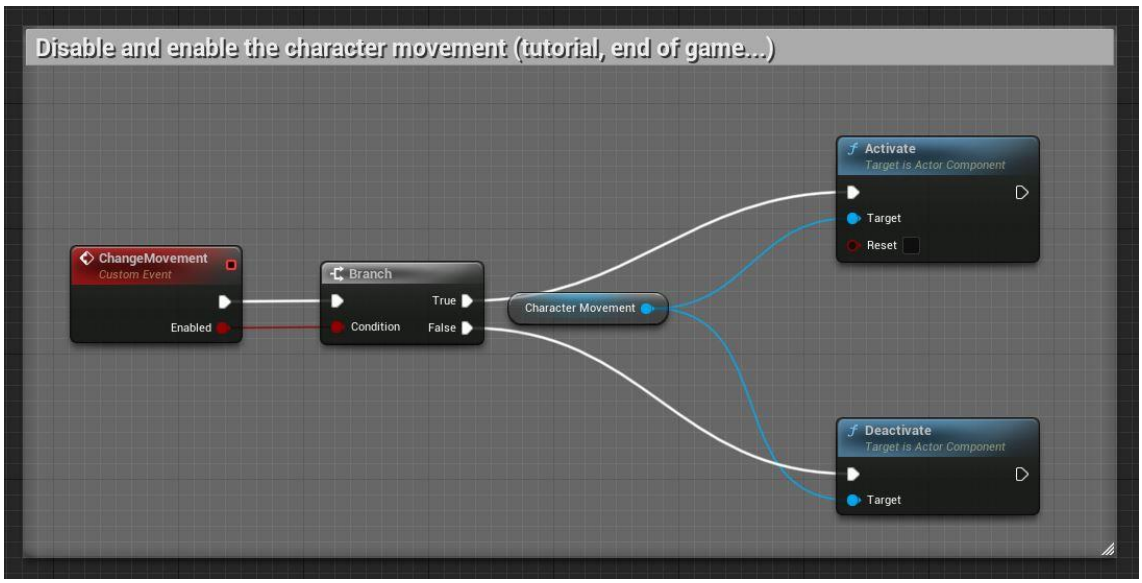

Figura 98. Esdeveniment que activa o desactiva el moviment

6.3.2. Canvi teclat-comandament

Per a millorar l'experiència del jugador amb el joc, s'ha utilitzat l'esdeveniment *AnyKey* (Figura 99), que comprova quina tecla o botó s'ha premut. D'aquesta manera es pot saber si en mig de la partida s'ha canviat el dispositiu d'entrada i s'està jugant amb teclat o amb el comandament. Primer de tot, amb la variable booleana *LastInputIsGamepad,* inicialitzada a *false* per defecte, es comprova si coincideix amb la funció *IsGamepadKey,* si és així, no cal canviar res, però si no, es crida l'*event dispatcher Update Interact Text*, que actualitza els missatges on surten indicacions dels controls que ha de prémer el jugador; guarda el nou valor a la variable; i es canvia el valor de *MaxWalkSpeed* segons el dispositiu d'entrada, ja que, com s'ha comentat abans, es comporten diferent.

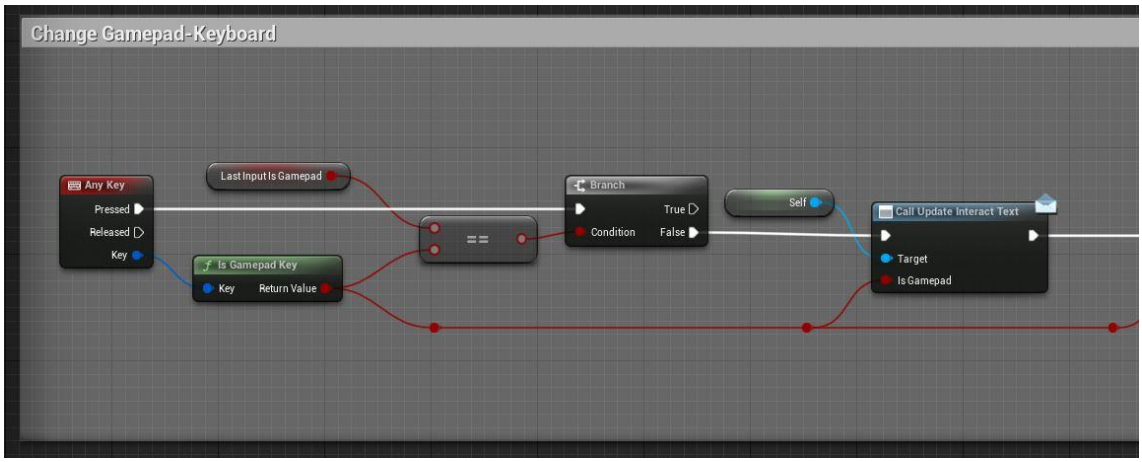

Figura 99. Esdeveniment AnyKey (primera part)

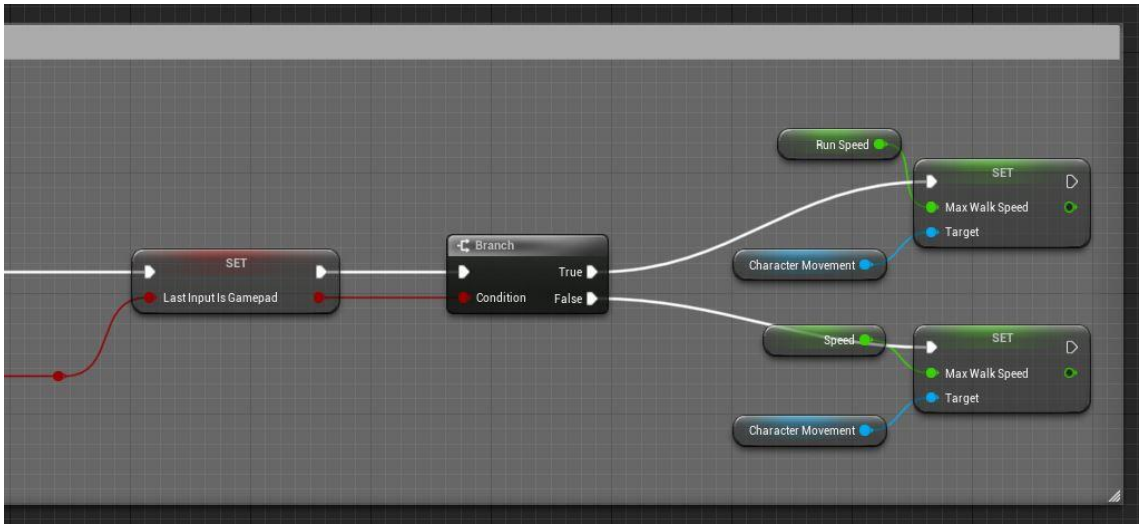

Figura 100. Esdeveniment AnyKey (segona part)

6.3.3. Inicialització

El primer que s'executa al *Blueprint* és l'esdeveniment *BeginPlay* (Figura 101 i Figura 102), on es defineix la salut màxima a l'iniciar la partida; es crea el *widget WBP_HUD* (Apartat 6.6.2), que s'afegeix al *viewport*; es defineix l'angle mínim i màxim en que el jugador pot moure verticalment la càmera, i es crida a l'esdeveniment *IntroTutorial* (Apartat 6.3.4).

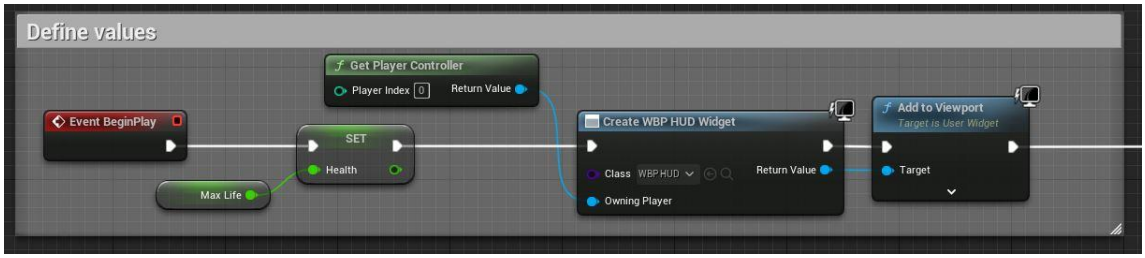

Figura 101. BeginPlay (primera part)

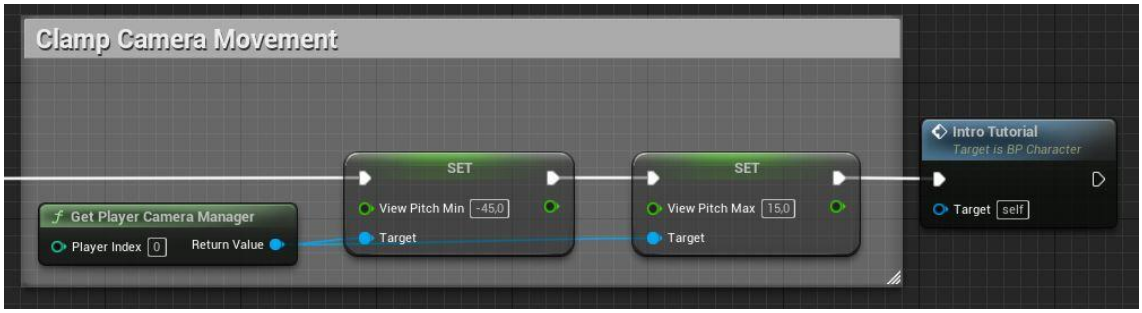

Figura 102. BeginPlay (segona part)

6.3.4. Tutorial

IntroTutorial (Figura 103 i Figura 104) s'encarrega de crear i afegir el *widget WBP_SpiritTutorial* (Apartat 6.6.2), que mostra diferents missatges sobre el context en el que es troba el personatge; restringeix el moviment per a que el jugador no es pugui moure fins que s'acabi l'explicació; reprodueix un petit so amb la veu del Gran Esperit, i

crida a l'esdeveniment StartTutorial del *widget,* al qual se li passa un *array d'strings amb* les diferents parts del missatge (Figura 105).

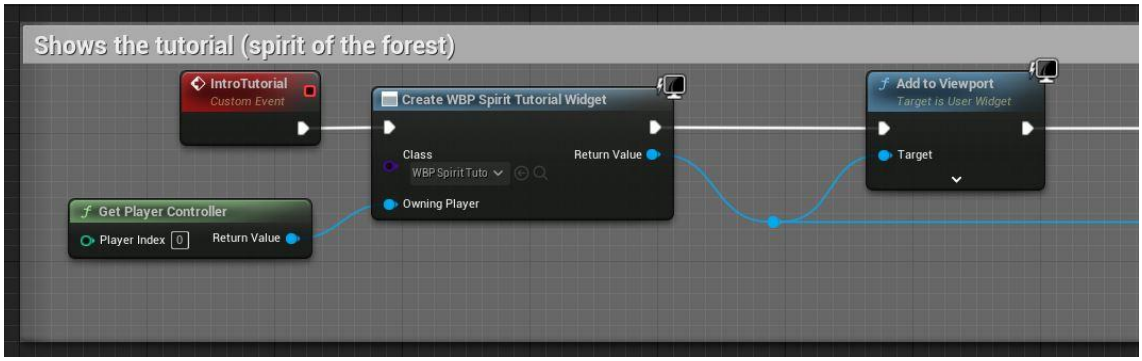

Figura 103. IntroTutorial (primera part)

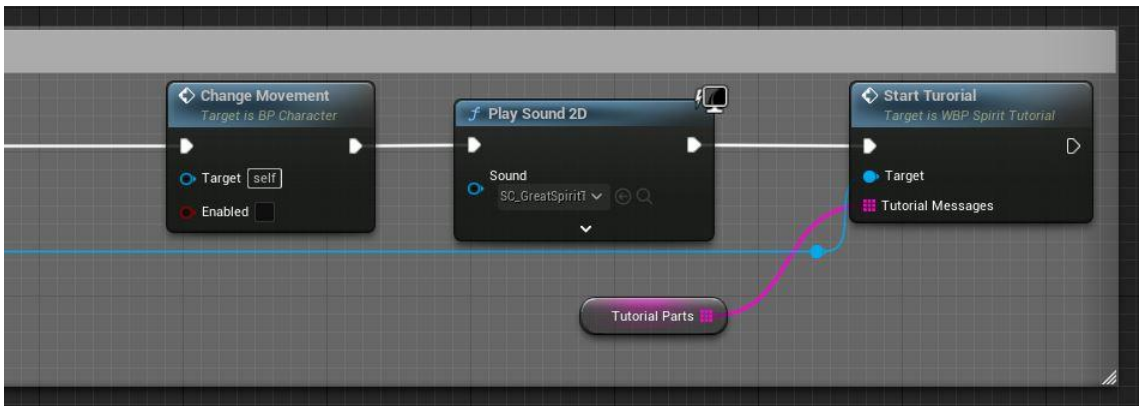

Figura 104. IntroTutorial (segona part)

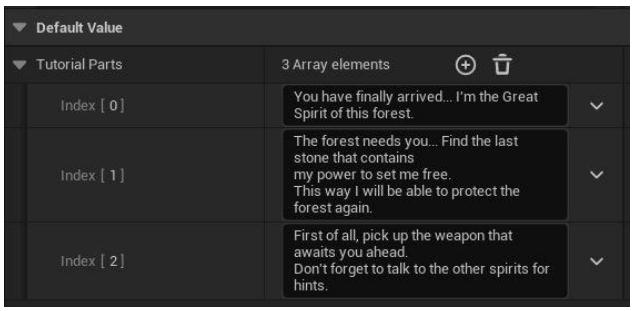

Figura 105. Array amb els missatges del tutorial

6.3.5. Interacció

S'han implementat diferents objectes que interactuen amb el jugador, on es necessita prémer una tecla (E) o botó (B) en específic per a executar un comportament en concret. Així doncs, s'ha creat un esdeveniment (Figura 106) on es comprova quants actors que implementen la interfície *BPI_Interaction*; una interfície creada per a que les diverses interaccions actuïn diferent; estan en contacte amb el personatge. Quan troba la primera, deixa de mirar les altres i executa la interfície de l'objecte amb el que es solapa.

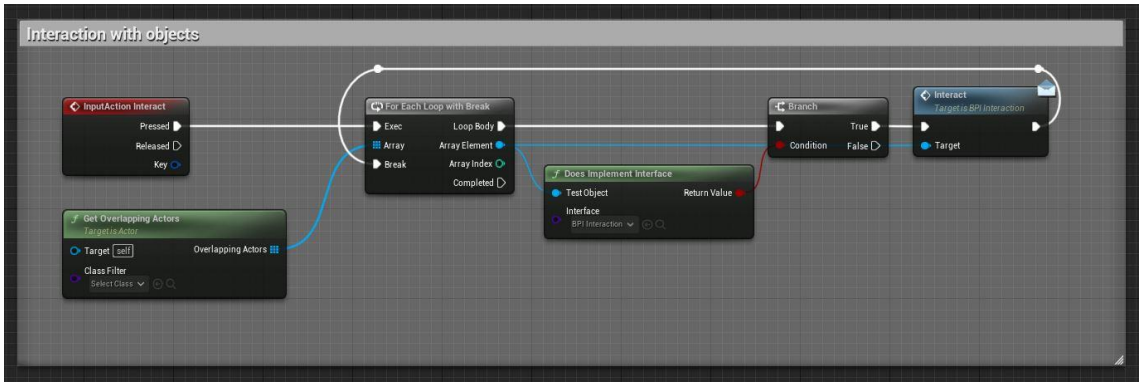

Figura 106. Interacció

6.3.6. Aconseguir arma

Un cop s'ha interactuat amb l'arma, que es troba a l'inici del joc, es crida a l'esdeveniment *WeaponAttached* (Figura 107), el qual guarda una referència a l'arma per a poder fer futures crides al *blueprint* corresponent. A més, l'arma canvia de posició i es situa a la mà del personatge. Aquesta posició s'indica amb un *socket* que s'ha creat anteriorment a la zona desitjada de l'esquelet, tal i com es pot veure a la Figura 108. Per últim, es canvia el valor de la variable *CanAttack* a *true*, ja que a partir d'aquest moment el jugador ja és capaç de lluitar.

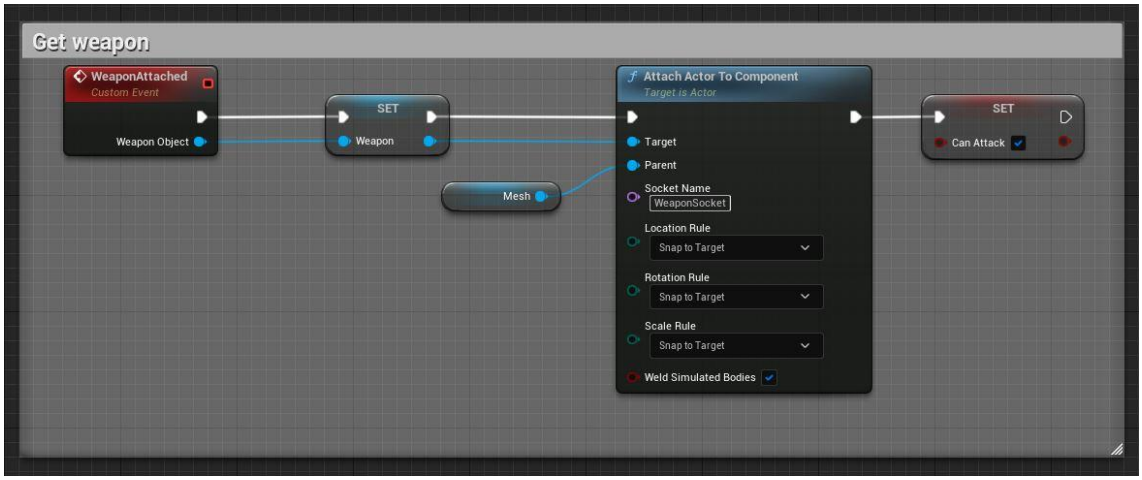

Figura 107. Agafar arma

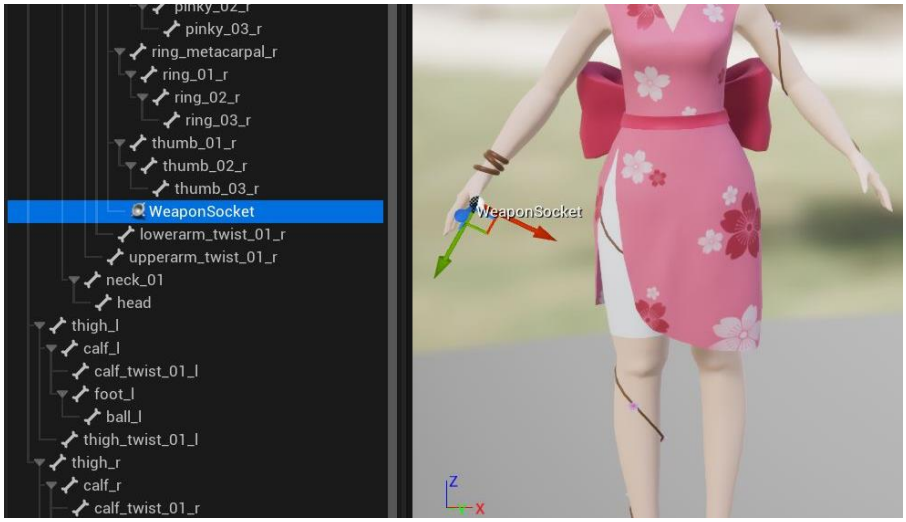

Figura 108. Socket de l'arma

6.3.7. Atacar

A continuació, com es pot veure a la Figura 109, s'explica detalladament el funcionament de la mecànica d'atacar.

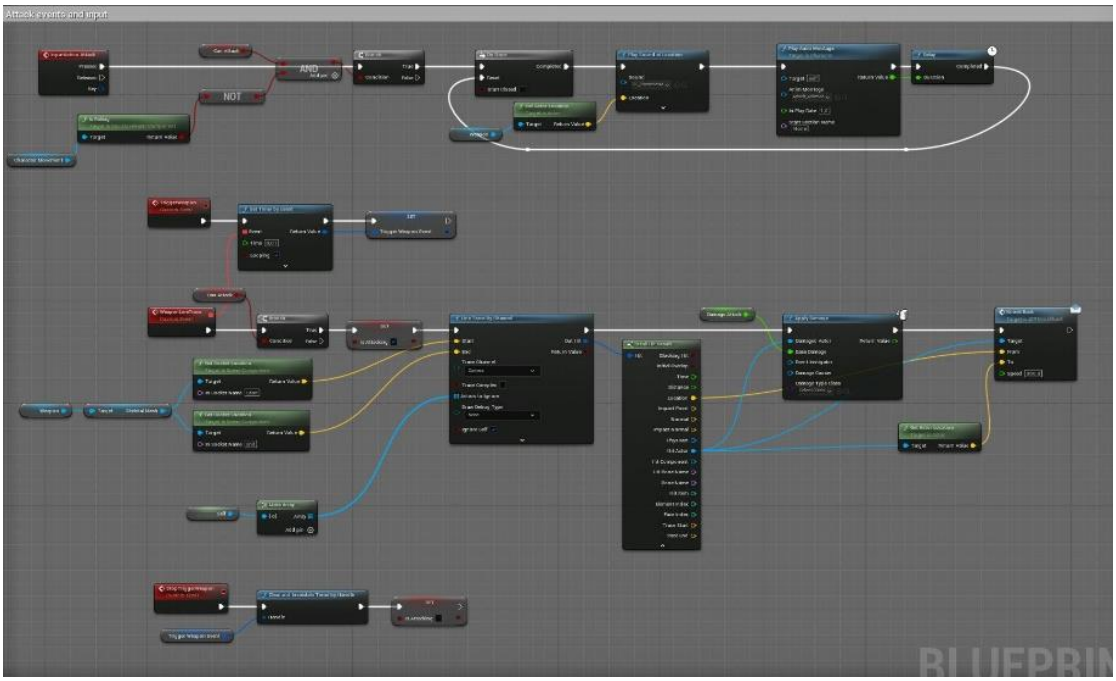

Figura 109. Visió general de l'atac

Per a atacar, el jugador ha de clicar amb el botó esquerre del ratolí o amb l'*X* del comandament. Un cop ho fa, es crida a l'esdeveniment *InputAction Attack* (Figura 110 i Figura 111), on, primer de tot, es comprova que el personatge tingui disponible l'acció d'atac i estigui a terra. Si es compleixen aquestes condicions, es reprodueix, només una vegada, un so amb la veu de la Hana i l'animació d'atac, quan acaba, es pot tornar a atacar.

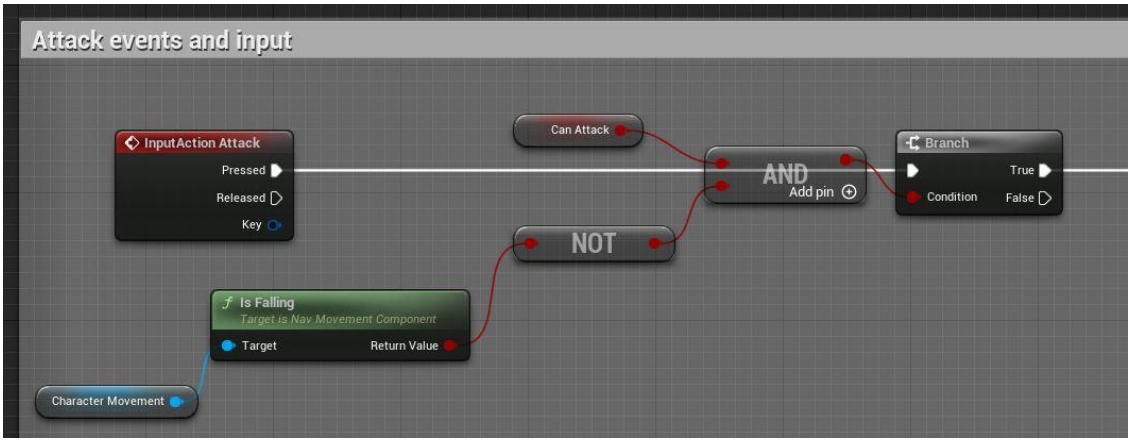

Figura 110. InputAction Attack (primera part)

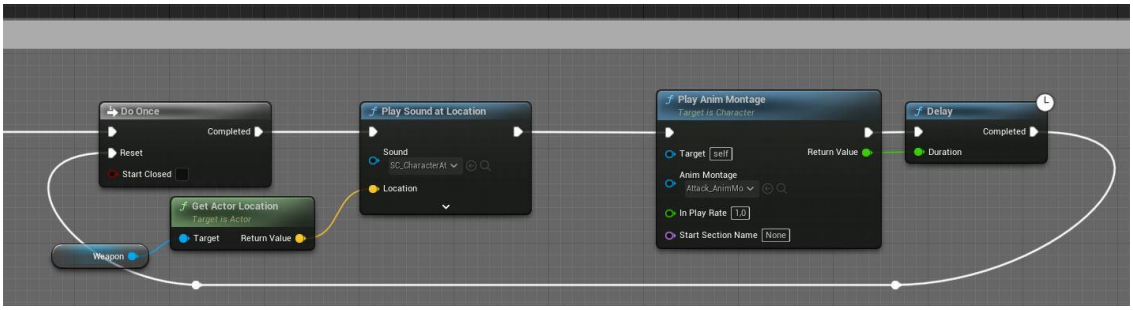

Figura 111. InputAction Attack (segona part)

Des de l'animació que s'acaba de comentar, es crida, a través d'un *notify* (Figura 112), l'esdeveniment *TriggerWeapon* del *blueprint WeaponEvent* (veure Apartat 6.3.8), el qual crea un *timer* que a la vegada executa l'esdeveniment *WeaponLineTrace* cada 0.01 segons.

El *WeaponLineTrace* (Figura 113 i Figura 114) comprova si el personatge té disponible l'atac, i si és així, ho indica al booleà *IsAttacking*. A partir dels *sockets* d'inici i fi definits a l'arma (Apartat 6.5.1) es traça una línia d'un a l'altre. Si aquesta línia col·lisiona amb algun actor, excepte el propi personatge, aquest rep mal (se li resta 1 punt de vida) i se li empeny. Aquesta última acció es duu a terme amb la interfície *BPI_Knockback*, que la implementa cada enemic (Apartat 6.4.1).

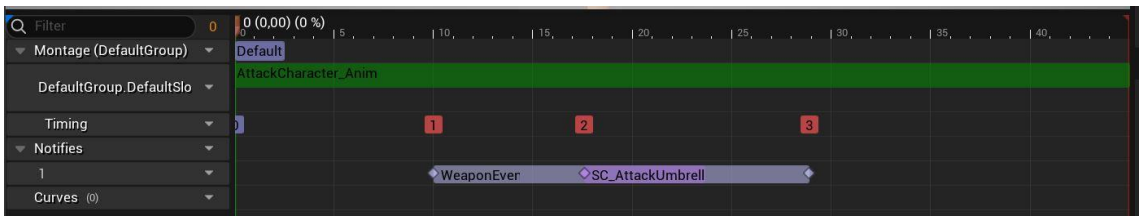

Figura 112. Línia de temps de l'animació amb el notify WeaponEvent

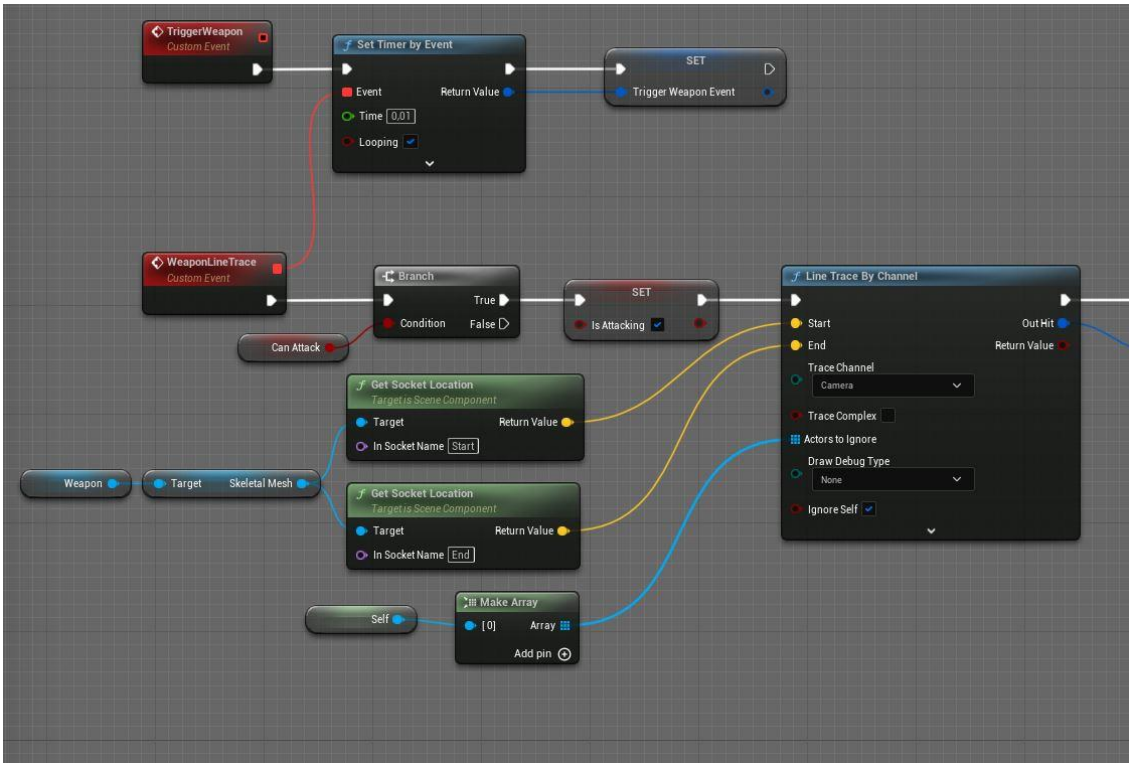

Figura 113. TriggerWeapon (primera part)

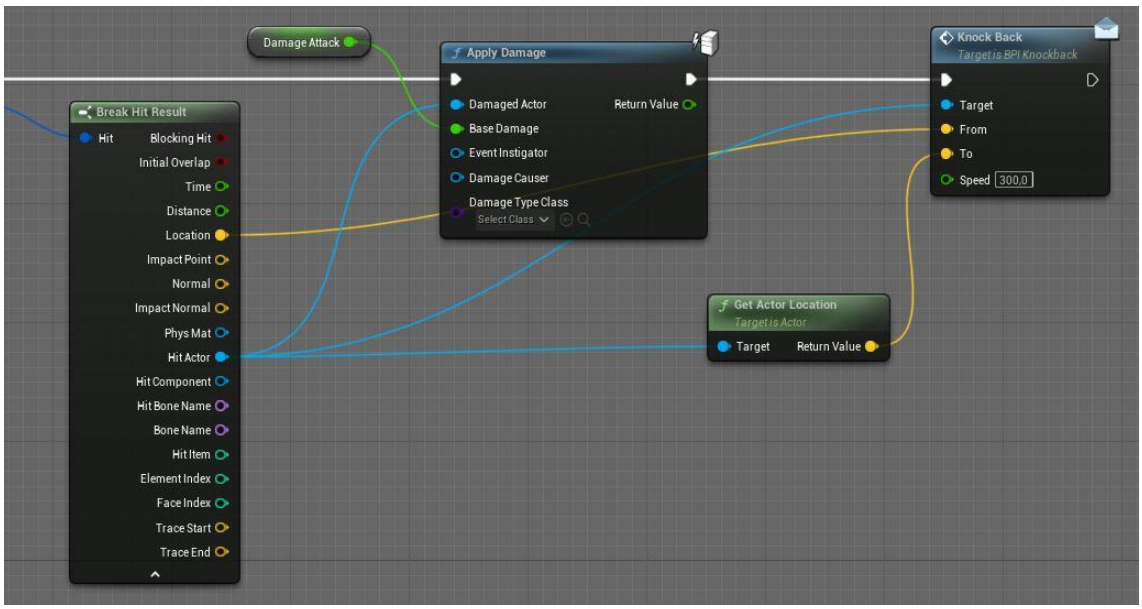

Figura 114. TriggerWeapon (segona part)

Per últim, quan el *notify* esmentat anteriorment acaba, des del *WeaponEvent* també es crida a l'esdeveniment *StopTriggerWeapon* (Figura 115), que s'encarrega de parar el *timer* iniciat a l'atacar i indicar a la variable *IsAttacking,* que ja s'ha acabat l'acció.

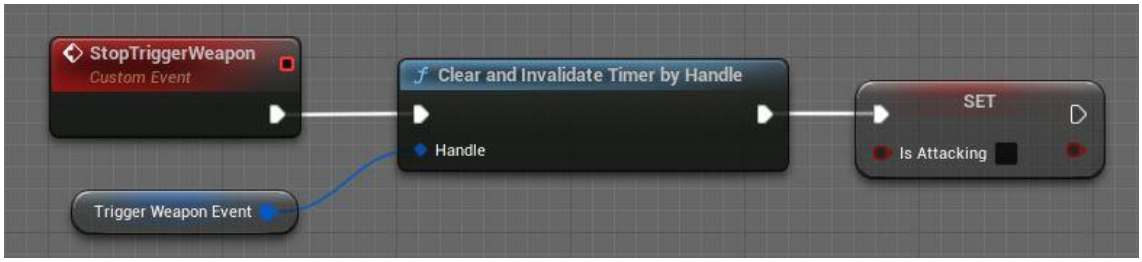

Figura 115. StopTriggerWeapon

6.3.8. WeaponEvent

El *blueprint WeaponEvent* és diferent al mostrat anteriorment, ja que aquest hereta de la classe *AnimNotifyState*, el que vol dir que s'utilitza per a cridar a diferents funcions a través de les animacions. Com es pot veure a la Figura 116, en aquest cas, s'han implementat dos de les que ja inclou la pròpia classe.

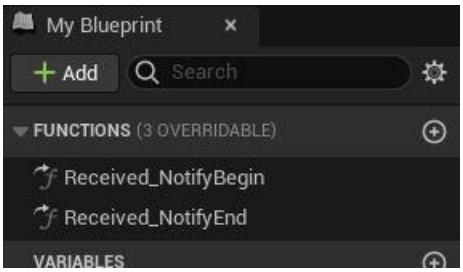

Figura 116. Funcions al WeaponEvent

Received_NotifyBegin (Figura 117) s'executa a l'inici del *notify* que s'ha adjuntat a l'animació (Figura 112), en aquest moment es crida a l'esdeveniment *TriggerWeapon* (Figura 113), vist a l'apartat anterior.

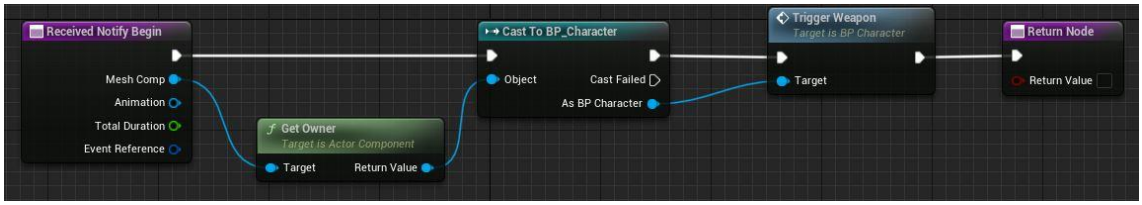

Figura 117. Received_NotifyBegin

Passa el mateix amb la funció Received_NotifyEnd (Figura 118), però en aquest cas s'executa al final del *notify* i crida a l'*StopTriggerWeapon* (Figura 115), que com s'ha comentat indica que s'ha acabat l'atac.

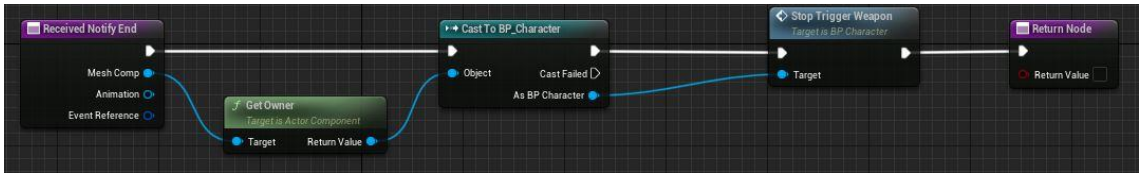

Figura 118. Received_NotifyEnd

6.3.9. Bloquejar

L'altra mecànica de lluita és el bloqueig, la qual s'activa clicant amb el botó dret del ratolí o l'*Y* del comandament. D'aquesta manera s'executa l'esdeveniment *InputAction Protect* (Figura 119).

Aquest esdeveniment, primer de tot, comprova que el personatge estigui a terra, tal com s'ha fet amb l'atac. Seguidament s'assegura que la referència de l'arma no és *null.* Per altra part, segons si el botó per al bloqueig s'ha premut o s'ha deixat anar, es fa una cosa o una altra. Si s'ha premut, la variable *CanAttack* canvia a *false* per a indicar que no pot atacar i a més, es crida a l'esdeveniment *WeaponShield,* en canvi, si el que s'ha fet ha estat alliberar el botó, la variable es posa a *true* i després es crida a l'esdeveniment *WeaponAttack.* Aquests dos esdeveniments s'expliquen més endavant a l'Apartat 6.5.1.

En aquest cas s'ha decidit que la mecànica de bloqueig s'implementi majoritàriament en la pròpia arma, cosa que no es fa amb l'atac. S'ha cregut més convenient fer-ho així, ja que el bloqueig es centra molt més en els *colliders* i animacions de la pròpia arma, quan en l'atac, tot i utilitzar el paraigües, tot el sistema de notificacions es basa en l'animació del personatge. Així doncs, aquesta manera ha facilitat el treball d'aquestes dues mecàniques.

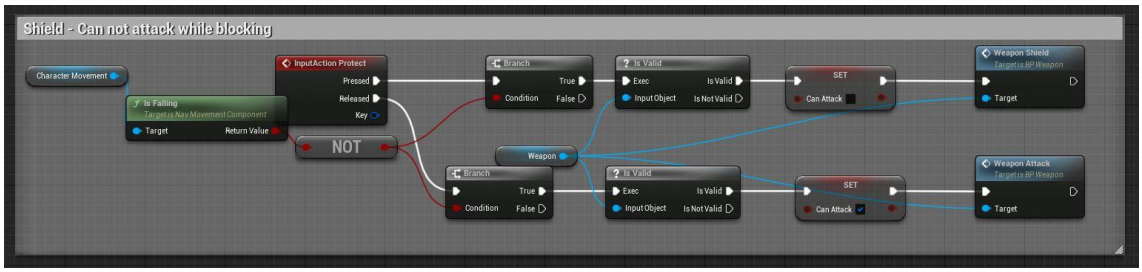

Figura 119. Bloqueig

6.3.10. Aconseguir pedra màgica

Quan el jugador és capaç de derrotar el primer grup d'enemics se li recompensa amb l'objecte necessari per a acabar el joc, la pedra màgica. Quan la pedra col·lisiona amb el personatge, es crida a l'esdeveniment *GetMagicStone* (Figura 120) que posa a *true* el booleà *HasStone* i actualitza la interfície d'usuari per a fer visible la imatge d'una pedra que indica que ja la ha aconseguit.

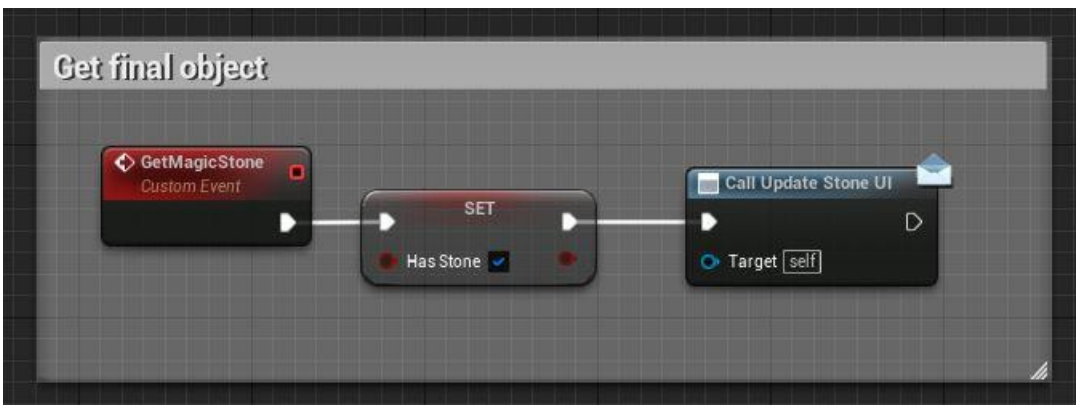

Figura 120. GetMagicStone

6.3.11. Healing

La funció *Healing* cura una determinada vida al personatge quan aquest interacciona amb el pètal o l'estàtua de la guineu. Com es pot veure a la Figura 121, es fan els càlculs pertinents per a la vida, assegurant que no se li pot afegir més de la màxima indicada a la variable *MaxLife.* A més, es crida a l'*EventDispatcher* encarregat d'actualitzar el HUD, per a restaurar els pètals (vides).

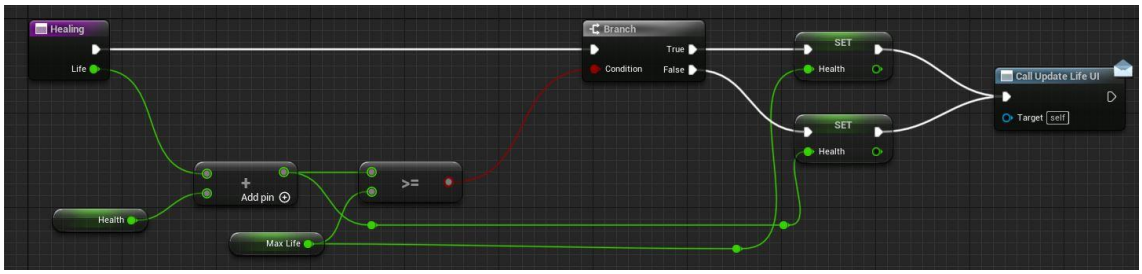

Figura 121. Healing

6.3.12. EndGame

EndGame és la funció encarregada de pausar el joc, crear i afegir el *widget FinalMenu* al *viewport,* tal i com es mostra a la Figura 122. Aquesta funció es crida un cop s'ha entregat la pedra màgica.

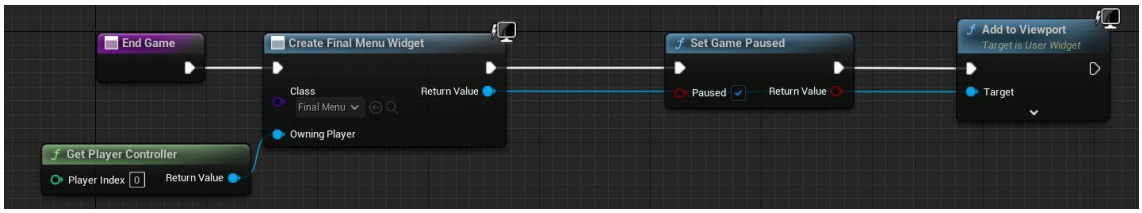

Figura 122. EndGame

6.3.13. DeadGame

Com es pot apreciar a la Figura 123, aquesta funció és pràcticament semblant a l'anterior, però, en aquest cas, es crea i afegeix el *widget DeadMenu*. Aquesta funció s'executa quan les vides del personatge arriben a 0.

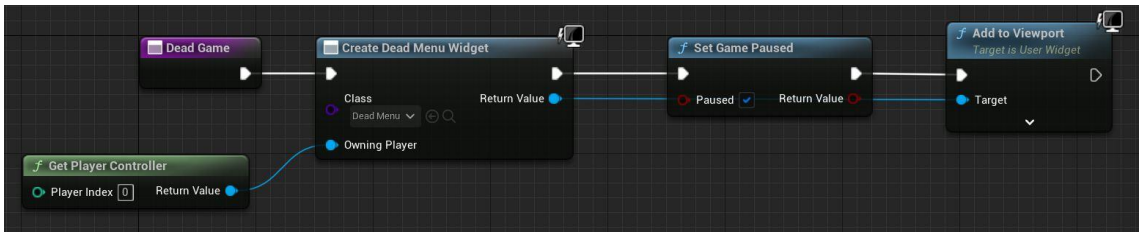

Figura 123. DeadMenu

6.3.14. HasStonePlatform

Tal i com es pot veure a la Figura 124, la funció HasStonePlatform simplement retorna el valor de la variable *HasStone*.

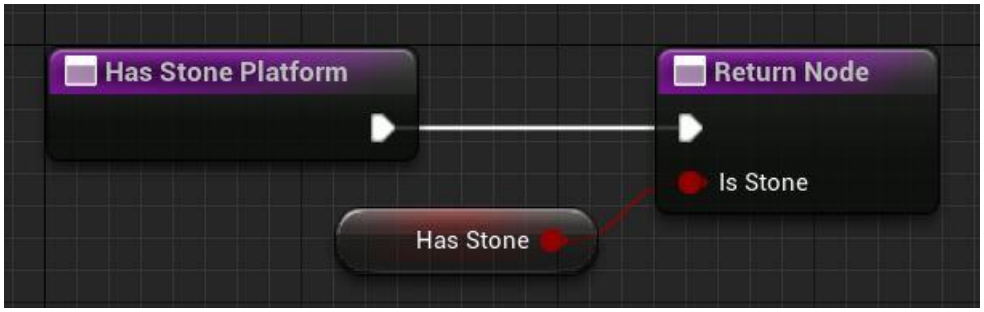

Figura 124. HasStonePlatform

6.3.15. HasMaxLife

La funció *HasMaxLife* (Figura 125) retorna un booleà que indica si la vida actual del personatge està al màxim.

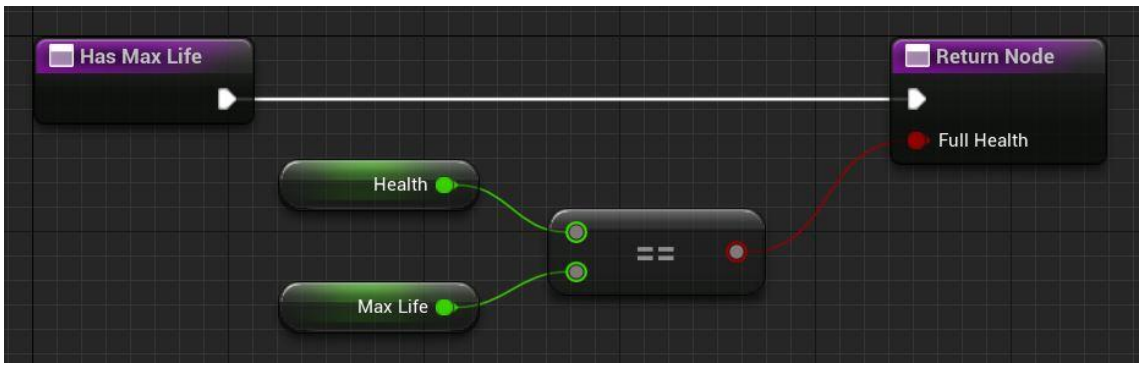

Figura 125. HasMaxLife

6.3.16. Attacking

Attacking (Figura 126) és una funció que retorna el valor de la variable booleana *IsAttacking.*

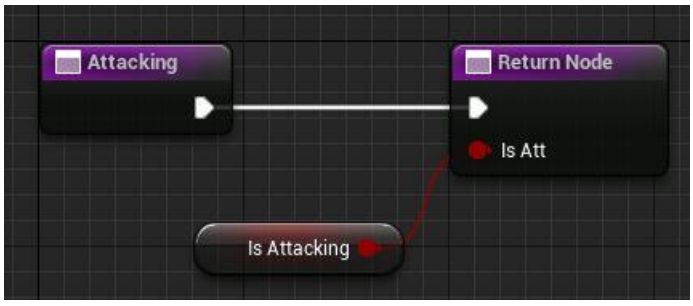

Figura 126. Attacking

6.3.17. Animacions

Com s'ha comentat en apartats anteriors, les animacions, excepte la del bloqueig, s'han extret de *Mixamo*; s'han tractat al *Blender*, ja que s'havia d'aconseguir que encaixessin correctament amb el model, i seguidament, s'han importat al motor.

Les animacions d'atac i bloqueig es criden directament des dels *blueprints*, en canvi, les de moviment (*idle,* caminar, córrer i saltar) s'han d'implementar a través d'un *Animation Blueprint*.

En el graf principal del *blueprint* (Figura 127), i en concret en l'esdeveniment *BlueprintUpdateAnimation* (Figura 128), es controla i modifica la variable booleana *IsFalling*, que indica si el personatge està saltant o no, i la variable *Speed*, que comprova la velocitat del personatge en cada instant. L'altre esdeveniment està relacionat amb els sons del personatge a l'hora de caminar, s'explica de forma detallada a l'Apartat 6.9.

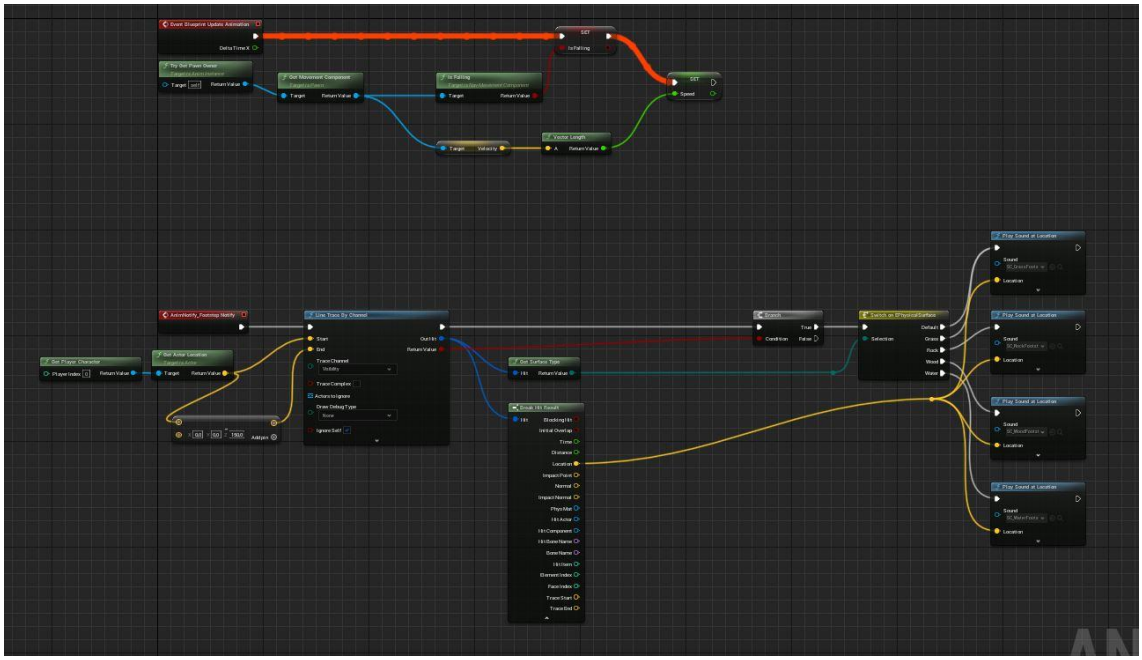

Figura 127. Visió general del graf de l'AnimationBlueprint

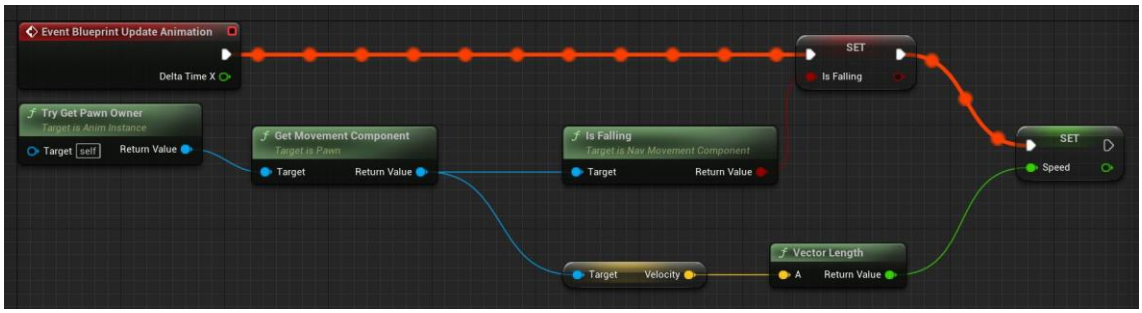

Figura 128. Esdeveniment Blueprint Update Animation

A més, aquest *blueprint* gestiona una màquina d'estats que permet canviar d'un estat a un altre mitjançant una condició. En aquest cas es compta amb dos estats el d'*idle/walk/run* i el de *jump*, tal i com es pot veure a la Figura 129.

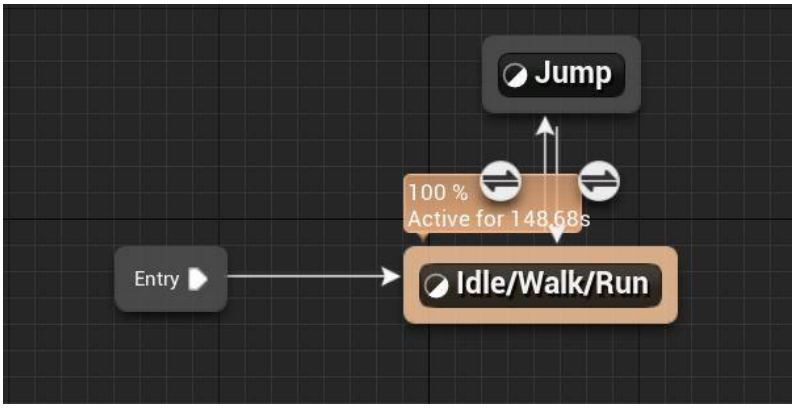

Figura 129. Màquina d'estats de les animacions

Per a la condició que determina si s'ha de transicionar d'un estat a un altre, s'utilitza la variable booleana *IsFalling*, com es mostra a la Figura 130 i Figura 131.

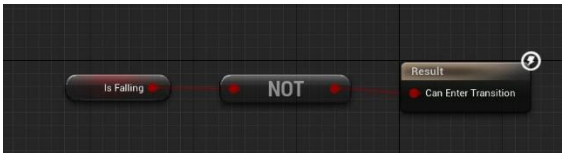

Figura 130. Transició estat jump a idle/walk/run

Figura 131. Transició estat idle/walk/run a jump

Cada estat inclou la seva pròpia implementació:

- L'estat *jump,* simplement, executa l'animació de saltar (Figura 132).

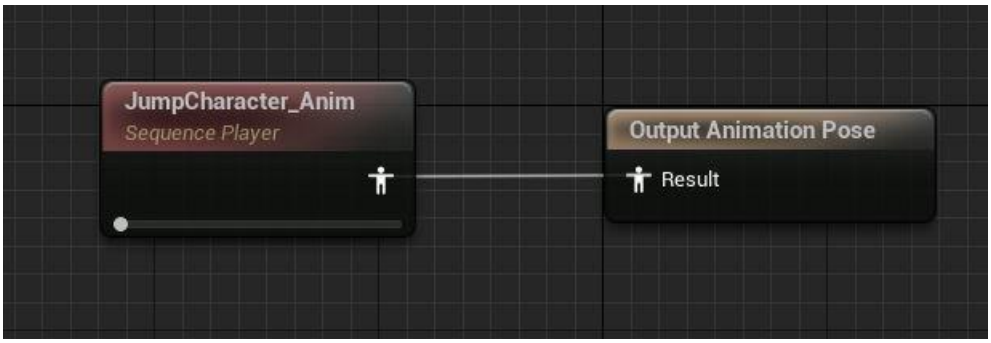

Figura 132. Estat jump

- L'estat *idle/walk/run* crida al *Blendspace* BS_TFGCharacter, al qual se li passa el valor d'*Speed*, com es pot veure a la Figura 133.

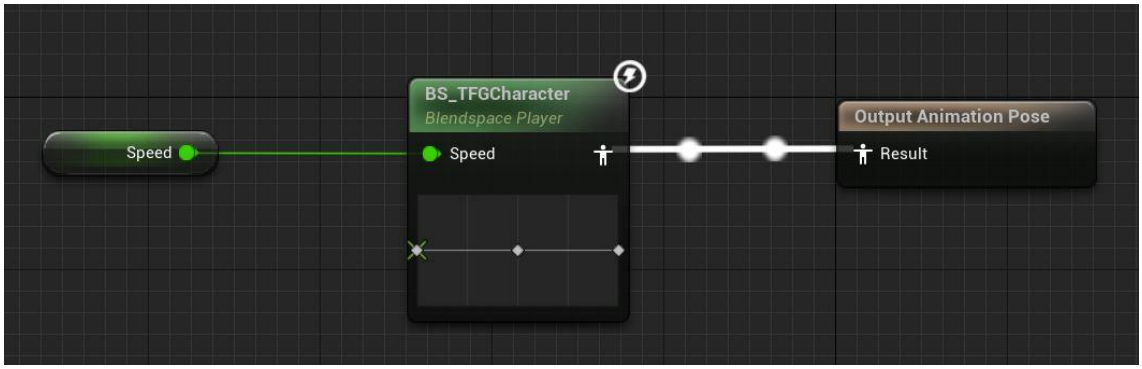

Figura 133. Estat idle/walk/run

El *blendscape* és l'encarregat de passar d'una animació a una altra segons la velocitat a la que es mou el personatge. Si no se li aplica cap velocitat, l'animació que es mostra és la d'*idle*; si la velocitat és la màxima, es mostra la de córrer; i entre mig, la de caminar, que es va adaptant a les altres segons augmenta o disminueix l'*Speed.* A la Figura 134 es pot veure com es defineixen els valors de la velocitat als paràmetres del panell de l'esquerre (*Minimum Axis Value* i *Maximum Axis Value*); també es pot apreciar al panell inferior els tres punts que indiquen les diferents animacions de moviment i com aquestes canvien en funció de la variable *Speed.*

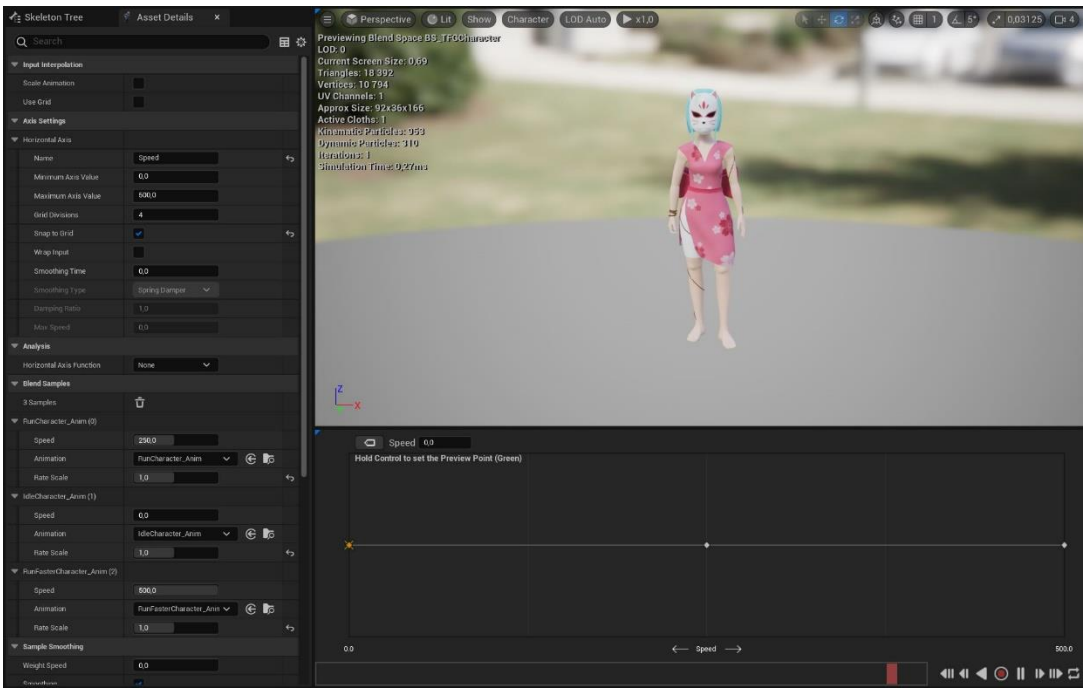

Figura 134. Blendspace BS_TFGCharacter

6.3.18. Cloth Simulation

Per a donar-li més moviment al personatge, s'ha decidit utilitzar l'eina *Cloth Simulation* (Figura 135) que incorpora l'*Unreal* i es troba a *l'asset skeletal* mesh (inclou l'esquelet, la física...). Aquesta eina permet que la roba no es quedi estàtica i es deformi de forma errònia amb les diferents animacions, així que, per aconseguir-ho, se li apliquen físiques.

Primer de tot, s'ha de crear el *clothing data,* on es guarden totes les configuracions de la roba. Aquest s'assigna al material que necessiti mobilitat, en aquest cas, s'aplica al vestit. Seguidament, activant el mode pintura, i amb l'ajuda del ratolí, es marquen les zones que es vol que tinguin física (blanc) i quines no (rosa). A més, també es pot configurar la distància del cos a la roba, la gravetat, i altres paràmetres que poden ajudar a que el moviment es vegi correctament.

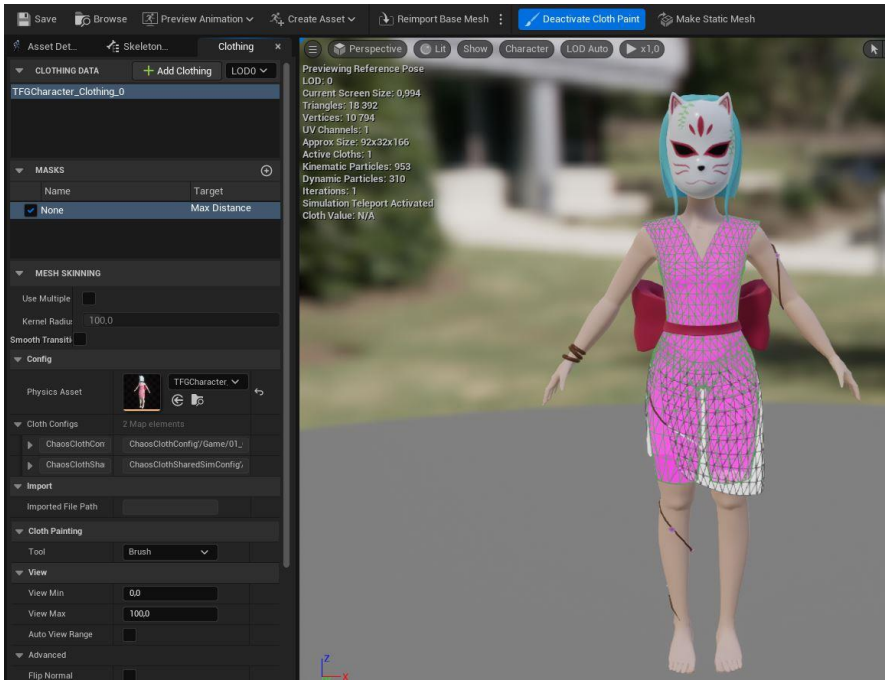

Figura 135. Editor de l'Skeletal Mesh amb el mode pintura de roba activat

Per altra part, és molt possible que segons l'animació, hi hagi zones del cos que travessen la roba. Per arreglar-ho s'ha de modificar el *physics assets* (associat a *l'skeletal mesh* anterior), el qual conté múltiples *colliders* repartits per tot el cos. Aquests *colliders* es poden moure, escalar i rotar; també s'hi pot afegir algun de nou per les zones que els altres no acaben de cobrir bé (Figura 136).

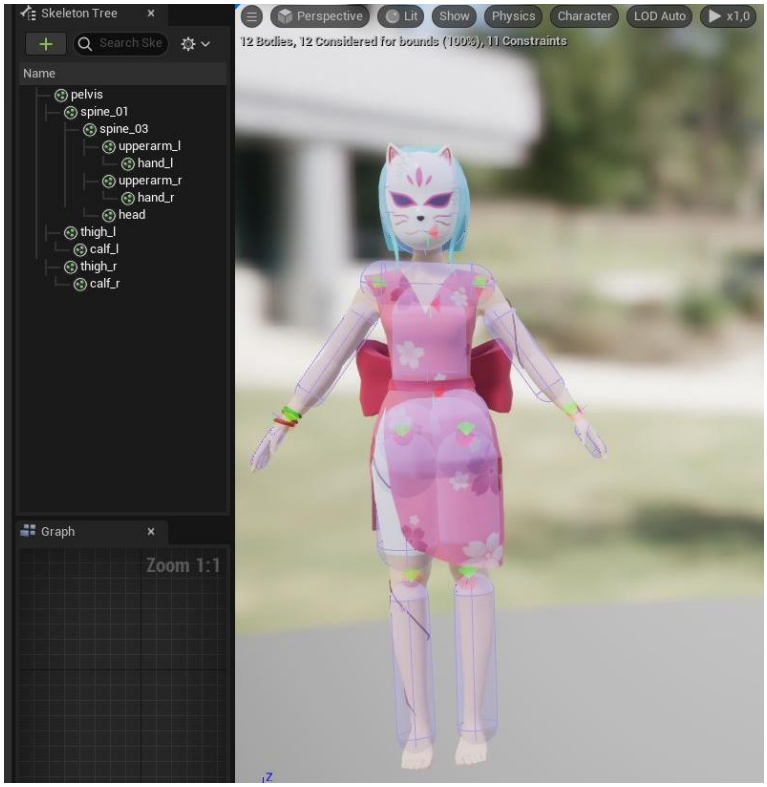

Figura 136. Physics Asset amb els colliders visibles

6.4. Enemics

6.4.1. Blueprint

Per a la implementació de l'enemic s'ha decidit utilitzar herència, és a dir, crear un enemic pare del qual hereten tres fills (*BP_Enemy_Basic, BP_Enemy_Teleport* i *BP_Enemy_Homing*), ja que la major part és comuna entre tots. El *blueprint* pare s'implementa a *BP_Enemy*. La Figura 137 mostra una visió general d'aquest.

Figura 137. Visió general de BP_Enemy

Els components són semblants als del jugador (Figura 138), però se li assigna una *AI Controller Class* (Figura 139), és a dir, una intel·ligència (Apartat 6.4.2). Per a que funcioni s'ha deshabilitat el paràmetre *Auto Possess Player*, el que indica que l'actor no és controlat pel jugador.

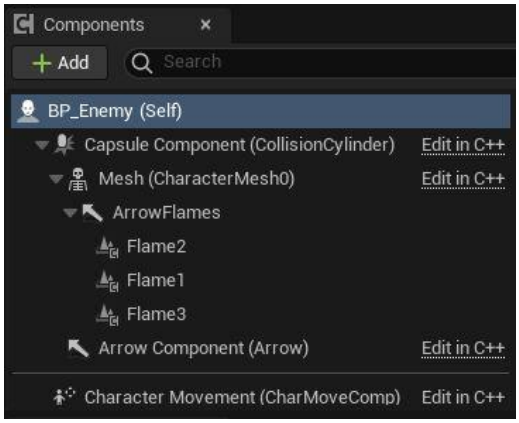

Figura 138. Components de BP_Enemy

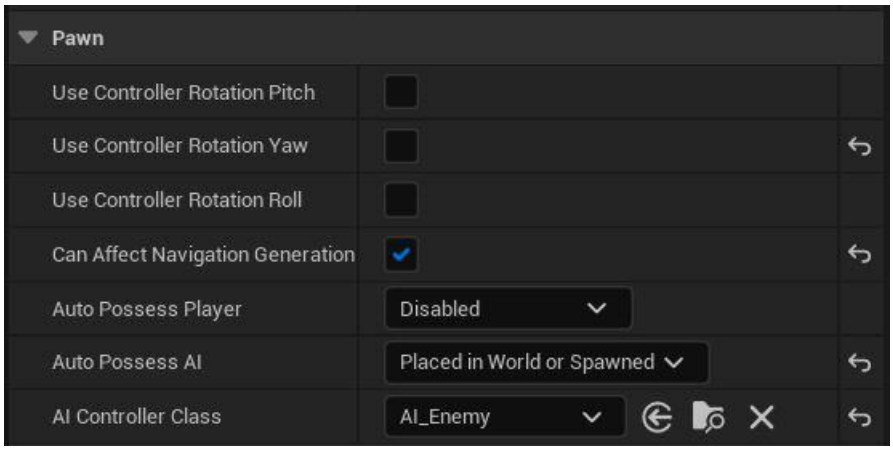

Figura 139. AI Controller

Inicialització

Per començar, a l'esdeveniment *BeginPlay* (Figura 140 i Figura 141), s'inicialitzen algunes variables, com l'*array* de flames (projectils) i una referència al personatge; es crida al mètode *SetFlamesClass,* que s'explica més endavant; guarda l'altura inicial de l'actor, i assigna la posició inicial a la variable *InitialPos* del *Blackboard*, del qual es parla en detall a l'Apartat 6.4.3.

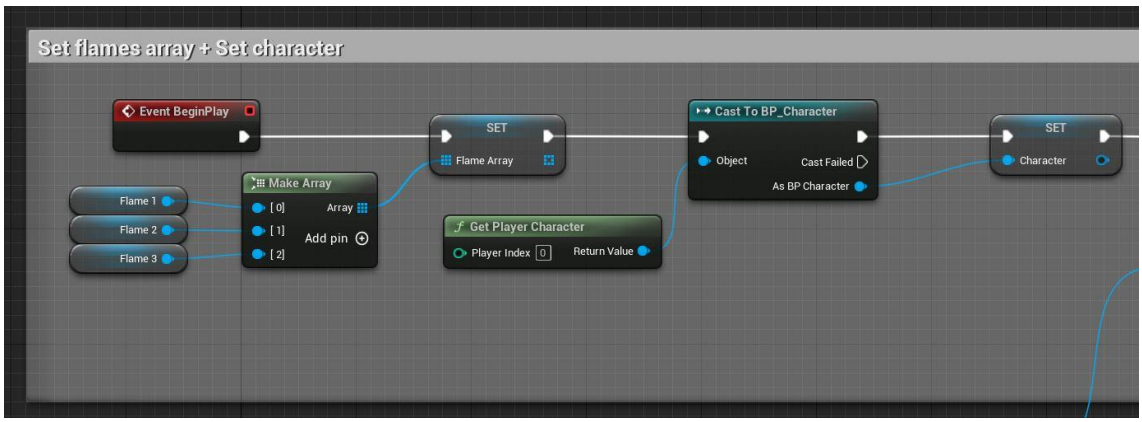

Figura 140. BeginPlay de BP_Enemy (primera part)

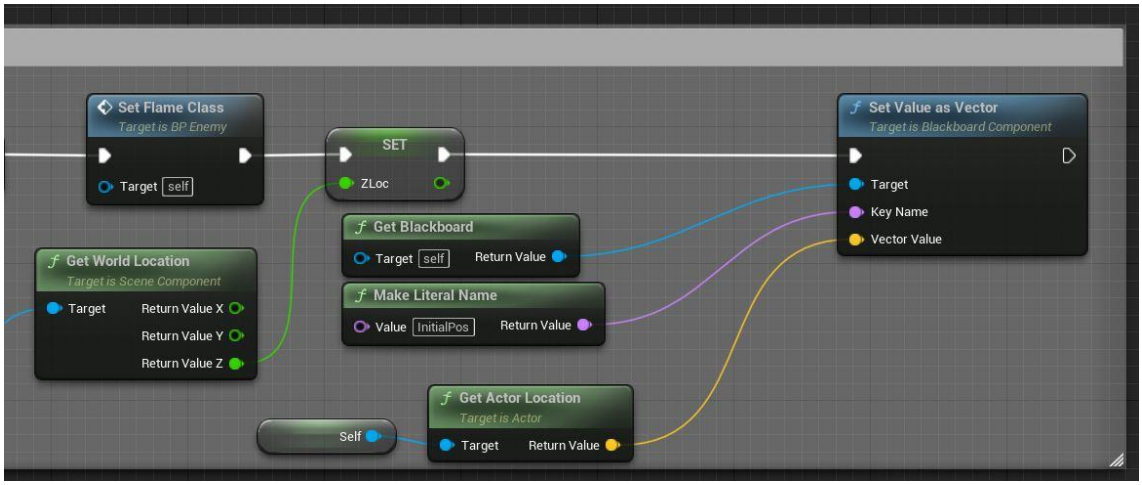

Figura 141. BeginPlay de BP_Enemy (segona part)

Event Tick

L'*Event Tick* (Figura 142, Figura 143 i Figura 144) és un mètode que s'executa cada *frame*, així que, en aquest cas, s'encarrega del moviment de l'enemic. Primer de tot, se li aplica una rotació al component *ArrowFlames*, el que fa que es vegi reflectit en les flames perquè aquestes són les seves filles. A continuació, s'assigna la posició en l'eix Z, ja que, per la decisió de fer que l'actor volés, el moviment en aquest eix ha donat molts problemes. Per acabar, en cas que l'enemic estigui atacant, aquest rota sobre ell mateix per a poder mirar al jugador.

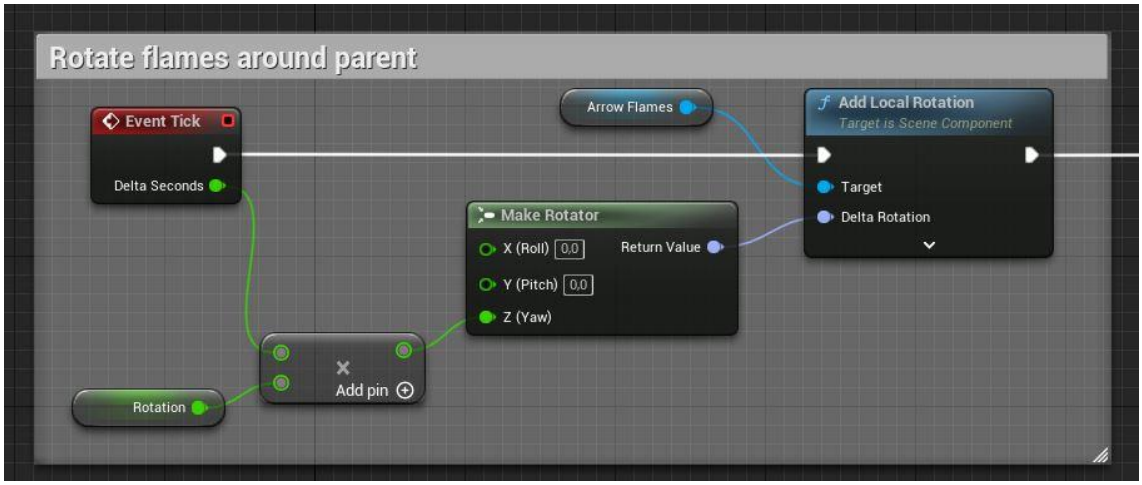

Figura 142. EventTick de BP_Enemy (primera part)

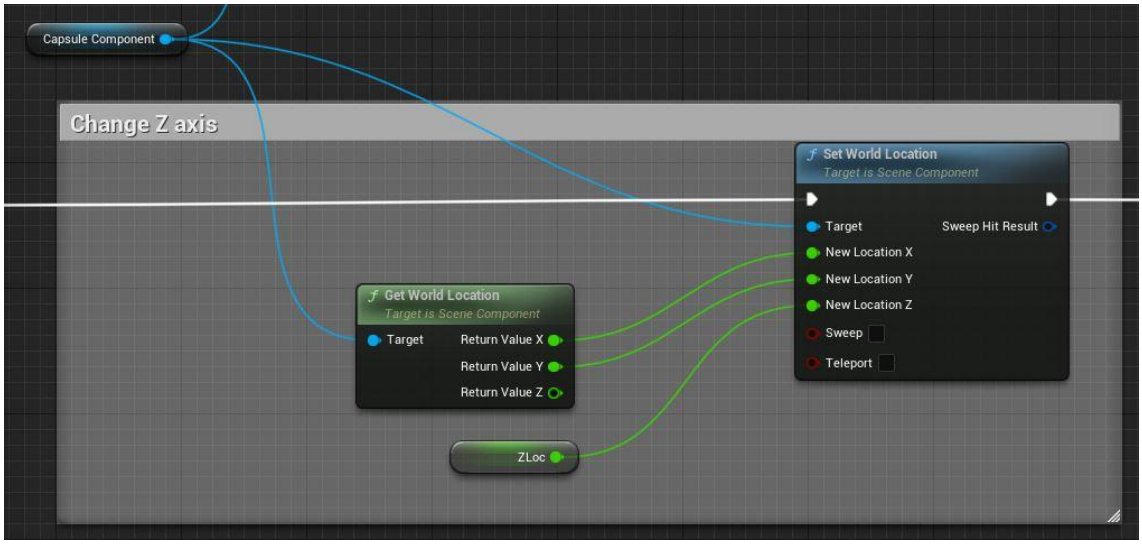

Figura 143. EventTick de BP_Enemy (segona part)

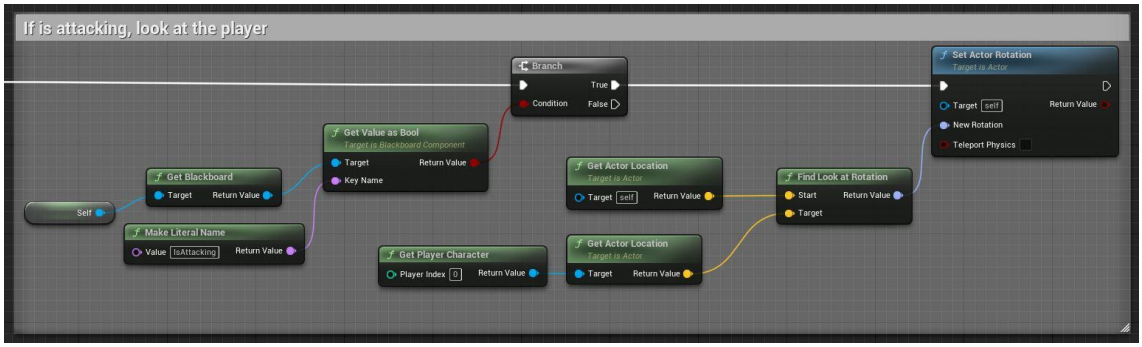

Figura 144. EventTick de BP_Enemy (tercera part)

SetFlameClass

L'esdeveniment *SetFlameClass* (Figura 145) canvia el valor de la variable *FlameType*, la qual indica quina flama s'ha d'instanciar.

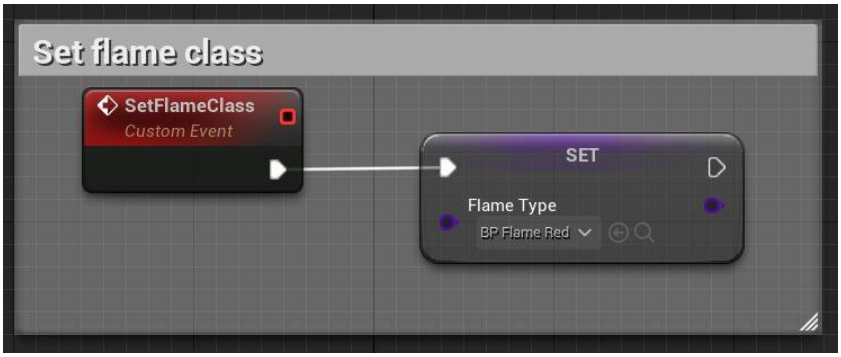

Figura 145. SetFlameClass

AnyDamage

El mètode *AnyDamage* (Figura 146 i Figura 147) s'executa quan el jugador ataca i l'arma col·lisiona amb l'enemic. Quan això ocorre, es reprodueix un so en el lloc, es resta 1 vida a la total de l'enemic. Si la vida no és 0, s'espera 1 segon a poder tornar a rebre mal; en

canvi, si sí que ho és, s'indica a la variable *isDead* del *Blackboard*, es crida a la funció *ChangeEnemiesAlive* del grup d'enemics (Apartat 6.4.6) i finalment, es destrueix l'actor.

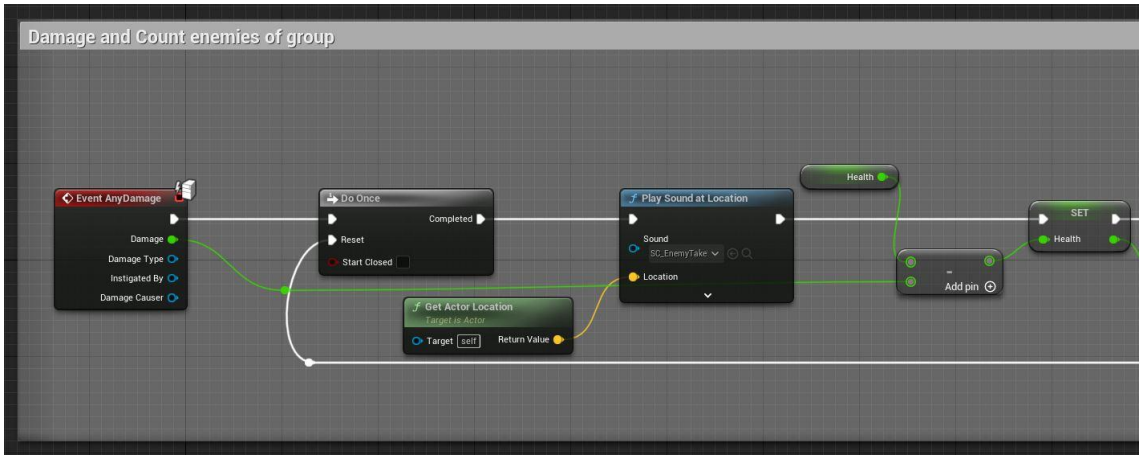

Figura 146. AnyDamage de BP_Enemy (primera part)

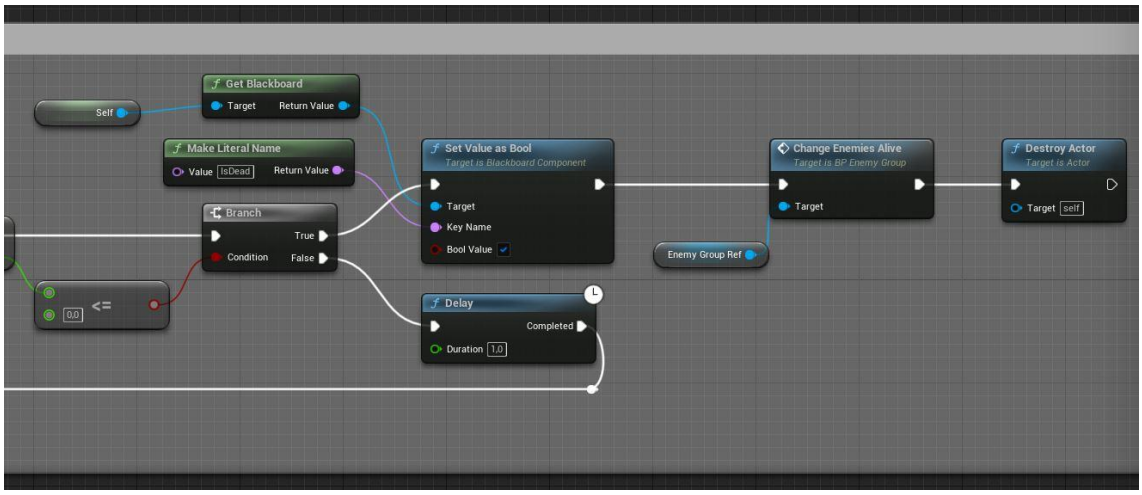

Figura 147. AnyDamage de BP_Enemy (segona part)

FireFlames

L'esdeveniment *FireFlames* (Figura 148, Figura 149 i Figura 150) és l'encarregat d'instanciar els projectils quan l'enemic ha d'atacar. Primer de tot, utilitzant la variable *FlameType* assignada anteriorment, s'instancia un actor d'aquesta classe a la posició d'una de les flames. A més, se li aplica la rotació pertinent per a que apunti en direcció al component *HomingProjectile* del jugador, el qual indica el destí. Seguidament, reprodueix un so i oculta el component que representa la flama en qüestió. A continuació, modifica la variable *CurrentFlameShot*, que fa la funció de comptador, per a mantenir un control del número de flames disparades; si ja les ha utilitzat totes, al cap de 2.5 segons es reinicien.

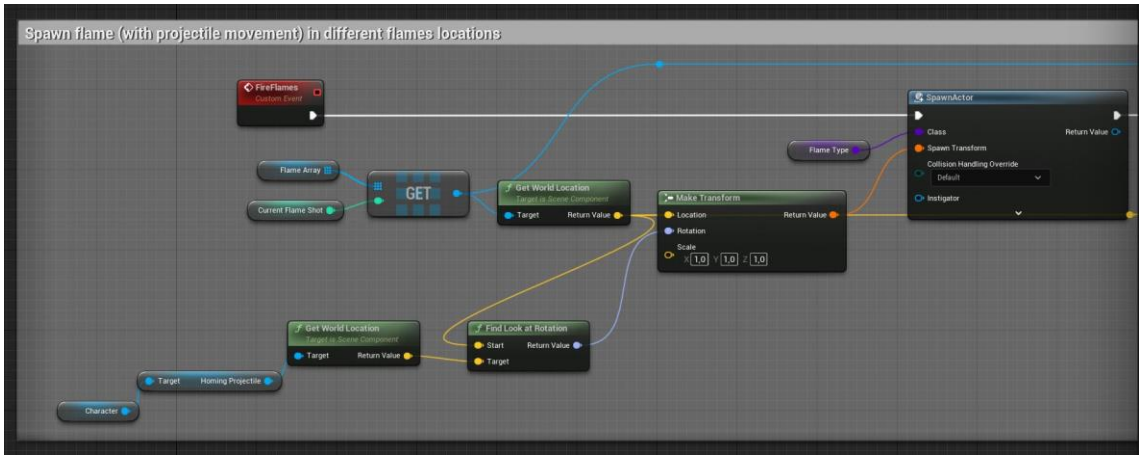

Figura 148. FireFlames (primera part)

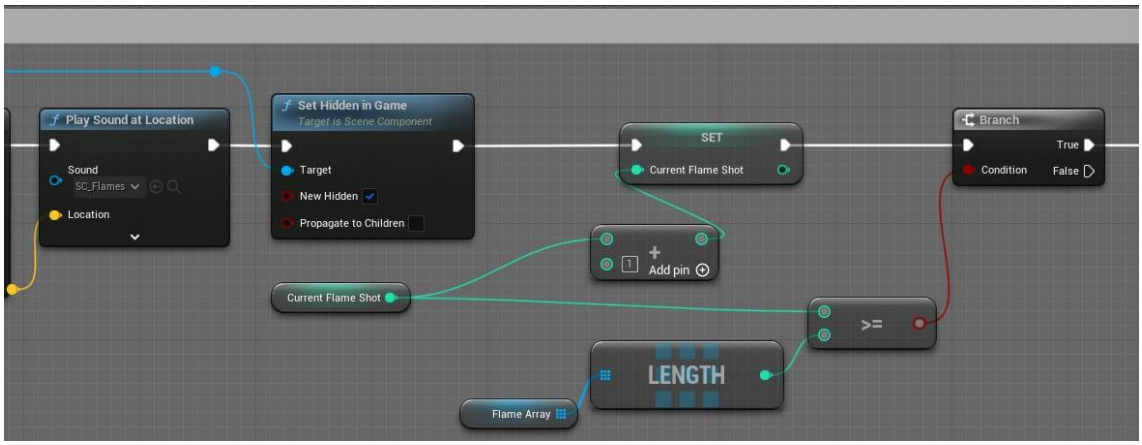

Figura 149. FireFlames (segona part)

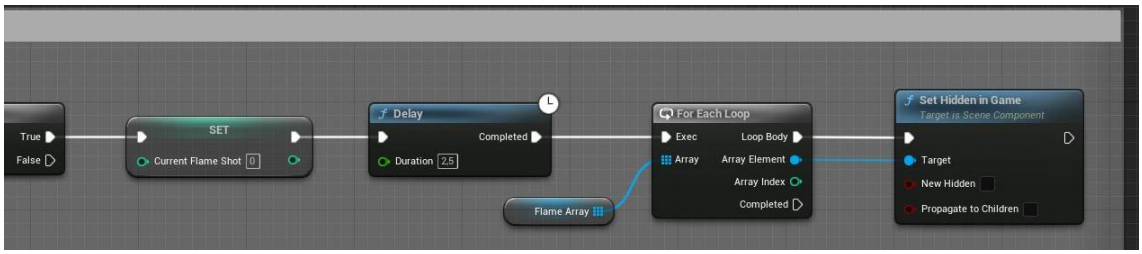

Figura 150. FireFlames (tercera part)

Knockback

Quan el jugador ataca o bloqueja a l'enemic, es crida a l'esdeveniment *Knockback* (Figura 151 i Figura 152) de la interfície *BPI_Knockback.* Aquest esdeveniment és l'encarregat d'aplicar un impuls a l'actor en una direcció i amb una velocitat en concret. Un cop aplicat es torna a canviar el tipus de moviment del personatge a *Flying*, ja que a l'executar l'impuls aquest canvia.

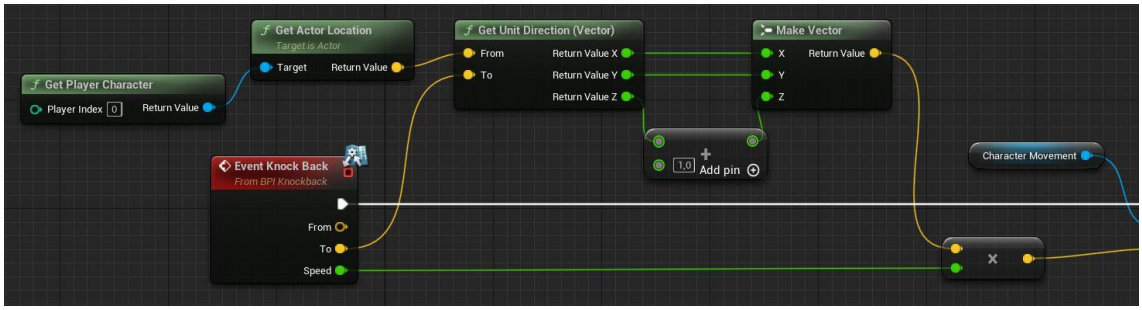

Figura 151. KnockBack (primera part)

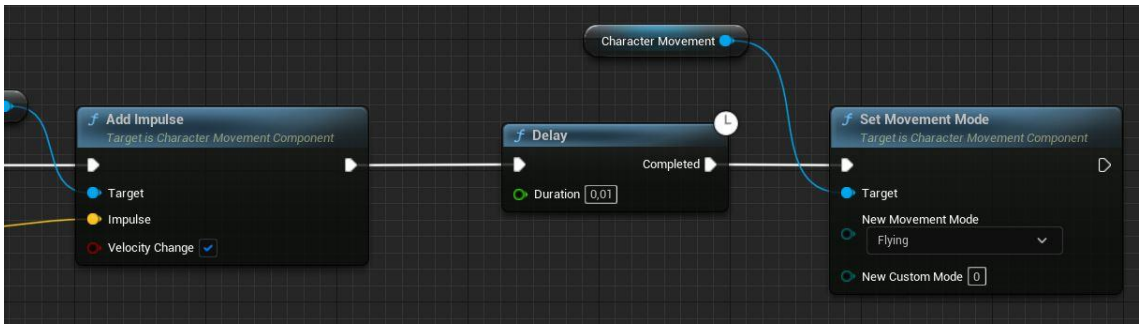

Figura 152. Knockback (segona part)

RestartFlames

A la Figura 153 es pot veure l'esdeveniment *RestartFlames*. Aquest mètode reinicia les flames, torna a mostrar-les totes i posa la variable *CurrentFlameShot* a 0, per a que al pròxim atac comenci de nou.

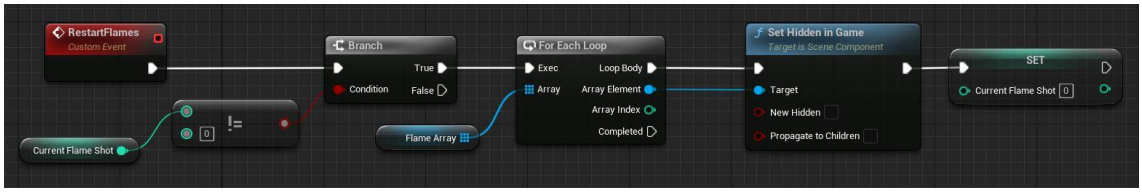

Figura 153. RestartFlames

Enemy_Basic

La implementació del fill *BP_Enemy_Basic* (Figura 154) és exactament igual que el pare, tot i així, s'ha creat aquest fill per si en un futur s'hagués de fer alguna modificació.

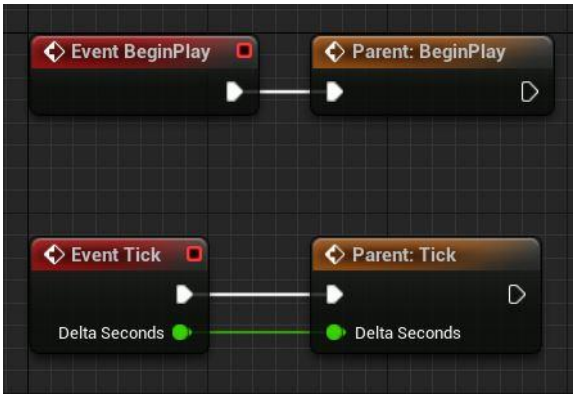

Figura 154. BP_Enemy_Basic

Enemy_Teleport

Pel fill *BP_Enemy_Teleport,* els únics canvis són el tipus de flama, com es pot veure a la Figura 155; l'*AI Controller Class*, que es canvia per un altre (Figura 156), i el valor de la variable *Health,* que augmenta a 3.

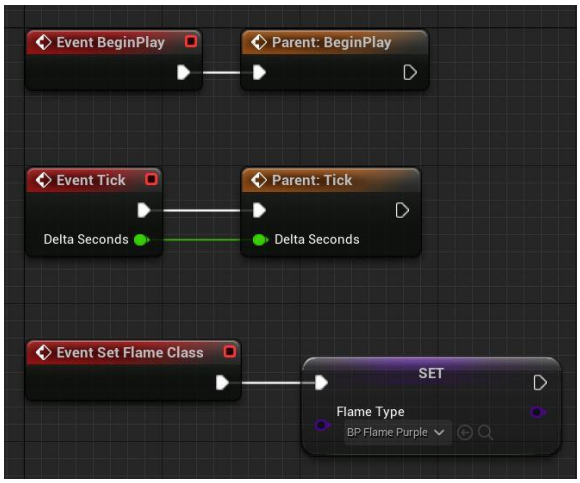

Figura 155. BP_Enemy_Teleport

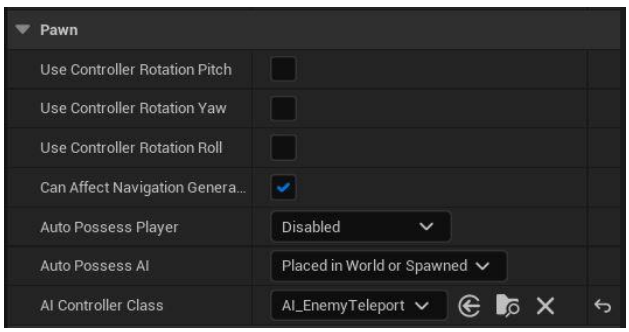

Figura 156. AIControllerClass de BP_Enemy_Teleport

Enemy_Homing

En el cas del fill *BP_Enemy_Homing* (Figura 157), passa el mateix que en l'apartat anterior. Aquest enemic modifica el tipus de flama per la *BP_FlameHoming*, s'assigna un *AIControllerClass* diferent (Figura 158) i se li assigna un valor de 4 a la variable *Health*.

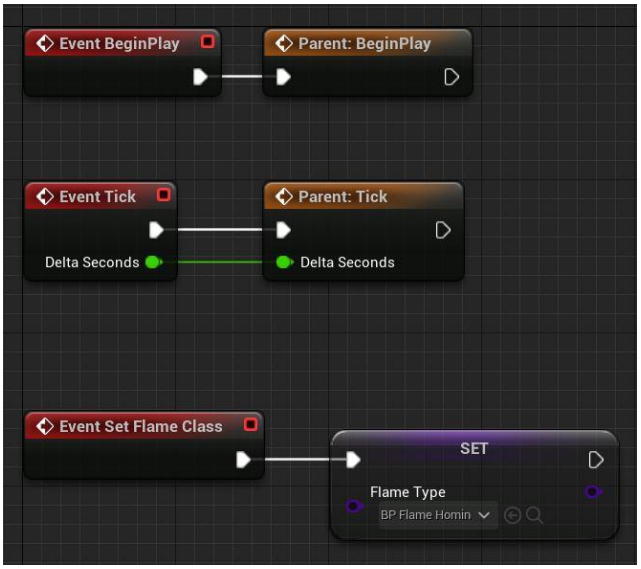

Figura 157. BP_Enemy_Homing

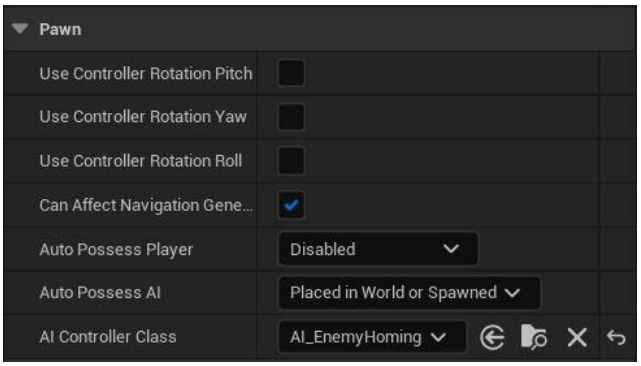

Figura 158. AIControllerClass de BP_Enemy_Homing

6.4.2. AI Controller

L'*AI Controller* és l'encarregat d'associar el comportament a l'enemic. Cadascun té el seu propi *AIController*; el *BP_Enemy_Basic* té l'*AI_Enemy*, el *BP_Enemy_Teleport* té l'*AI_EnemyTeleport* i el *BP_Enemy_Homing* té l'*AI_EnemyHoming*; tal i com es pot veure en la Figura 159, Figura 160 i Figura 161. En la inicialització es crida al node *RunBehaviorTree*, que inicia el comportament. A més, s'ha afegit un component *AIPerception*, que inclou l'esdeveniment *OnTargetPerceptionUpdated*, el qual crida a la funció *HandleSightSense*.

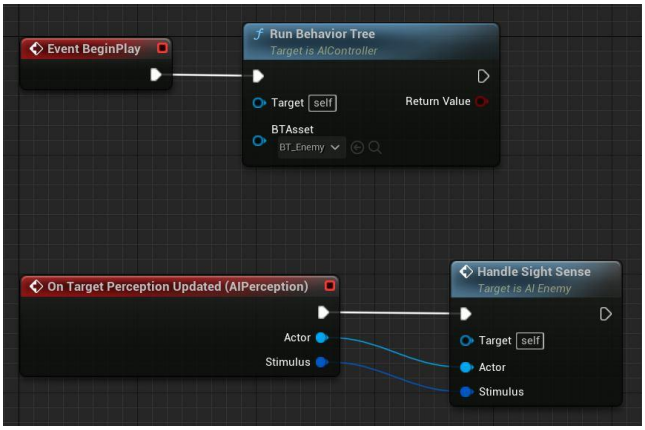

Figura 159. AI_Enemy

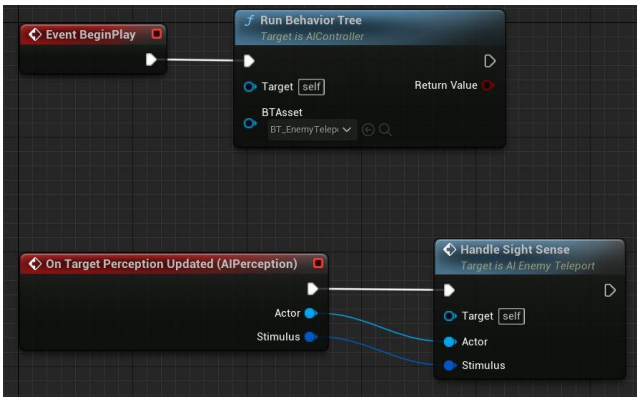

Figura 160. AI_EnemyTeleport

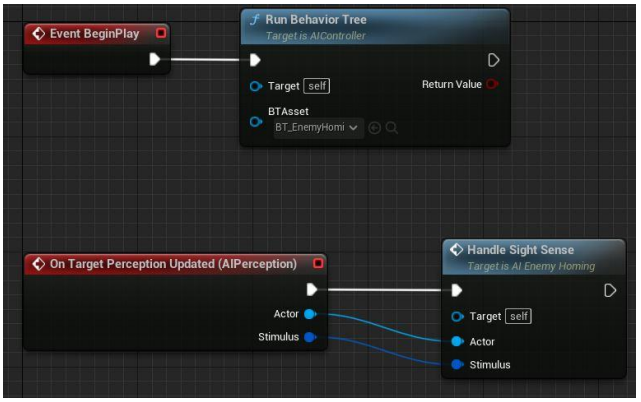

Figura 161. AI_EnemyHoming

HandleSightSense

La funció *HandleSightSense* (Figura 162 i Figura 163) s'implementa igual en tots els enemics. Aquesta funció s'executa quan un objecte entra a l'àrea de visió de l'enemic. Si l'objecte és el personatge, es guarda una referència i se l'assigna a la variable *TargetActor* del *Blackboard*.

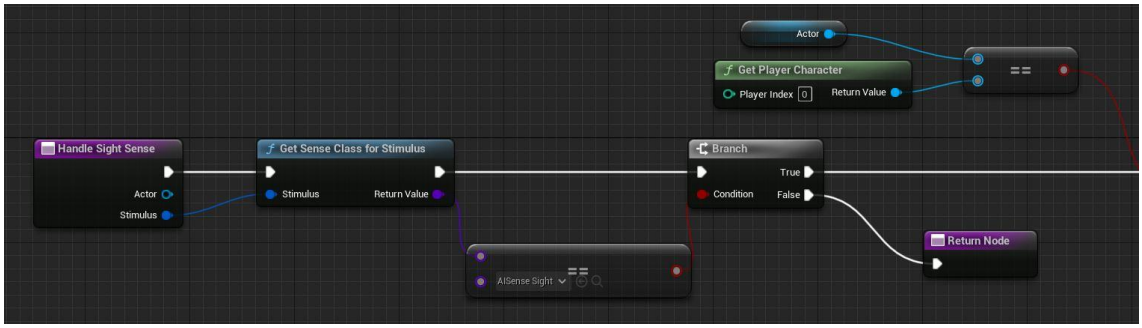

Figura 162. HandleSightSense (primera part)

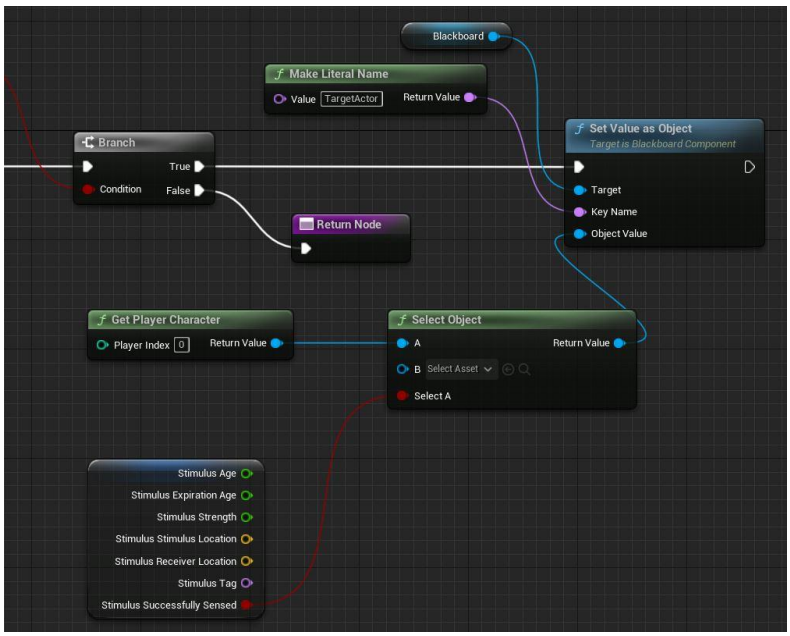

Figura 163. HandleSightSense (segona part)

6.4.3. Behavior Tree

El *Behavior Tree* (BT) és un *blueprint* que s'utilitza per crear la intel·ligència artificial dels personatges que no es poden controlar, com en aquest cas, els enemics. Els BT són arbres que defineixen el comportament i accions d'aquets personatges, i com diu el nom, tenen forma d'arbre. A més, tenen associat un *Blackboard* (BB), un altre *asset* que conté la definició de les variables, anomenades *keys*, que s'utilitzen al BT (Figura 164) per a controlar el comportament i dirigir-lo cap a certes branques amb els respectius nodes.

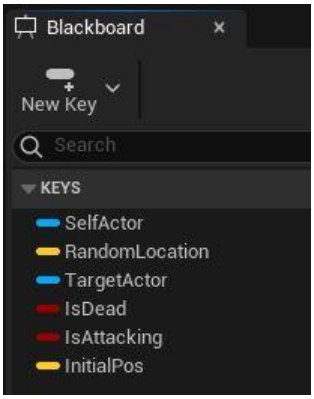

Figura 164. Keys del BlackBoard

Els nodes més importants són els següents:

- Root
  El *Root* és el primer node, el qual no es pot alterar. D'aquest se'n poden penjar altres, com un *Selector*, *Sequence* o *SimpleParallel.*
- Selector
  El *Selector* és un node que executa els seus fills d'esquerra a dreta, però no passa al següent fins que l'anterior falla.
- Sequence
  El *Sequence* és semblant al *Selector*, però quan tota la branca del fill esquerre acaba, passa al següent, no necessita que cap avorti.
- SimpleParallel
  El *SimpleParallel*, és un node que pot executar dos accions a l'hora, una principal i l'altre que s'executa en segon pla. En aquest cas, cap dels enemics l'utilitza.
- Task

  Les *Tasks* són nodes que executen una acció. Existeixen algunes predeterminades, com el *MoveTo,* que mou al personatge cap a un objectiu; el *Wait* que executa un *delay*; el *PlaySound* que reprodueix un so... A més, es poden crear i implementar-ne de noves. En aquests nodes es poden relacionar les variables de la *task* amb les del *Blackboard* i assignar-les valor, tal i com es mostra a la Figura 165.

Figura 165. Variables Task i BB

A tots els nodes, excepte el *Root* i les *Tasks*, se'ls pot afegir un *Decorator*. El *Decorator* és una condició que permet controlar si s'executa el node en qüestió o no. A la Figura 166 es mostra un exemple en el que es pot veure com aquest node no s'executaria fins que la variable *TargetActor* que es troba al *Blackboard* tingui valor. A més, en cas que

s'hagi entrat i s'estiguin executant els fills d'aquest node, si el resultat de la condició deixa de ser *true*, totes les branques i els respectius nodes des d'aquest, avorten (Figura 167).

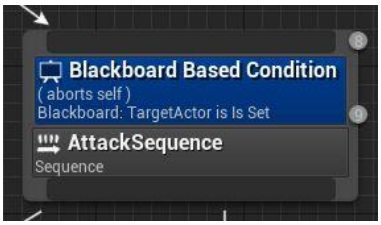

Figura 166. Decorator

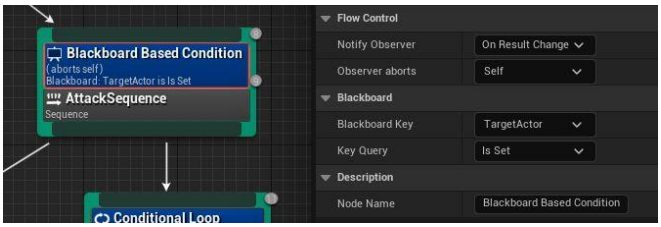

Figura 167. Avort

Enemy_Basic

El BT de l'*Enemy_Basic* (Figura 168), primer de tot, comprova si ha mort, i si no és així, mira si la variable *TargetActor* té un valor assignat. Si no té un valor assignat, es crida a la *task ResetAttack*, que reinicia les flames; és útil per quan s'avorta la branca de la dreta; després, mentre el *TargetActor* segueixi sense valor, es busca una posició aleatòria dins d'un rang, el personatge es mou fins aquesta i espera 3 segons per tornar a fer aquestes tres accions en bucle.

En cas que a la variable *TargetActor* se li assigni un valor, la branca esquerra s'avorta, i es comença a executar els nodes de la dreta. Primer, es mou a l'enemic fins a una certa distància del jugador. Seguidament, s'espera 1 segon fins a començar a atacar; això s'executa tres vegades amb una pausa d'1.5 segons entre atac i atac. Per acabar, s'executa un *delay* de 2.5 segons i es reprodueix un so. Aquest últim *delay* s'utilitza per donar més temps de reacció al jugador.

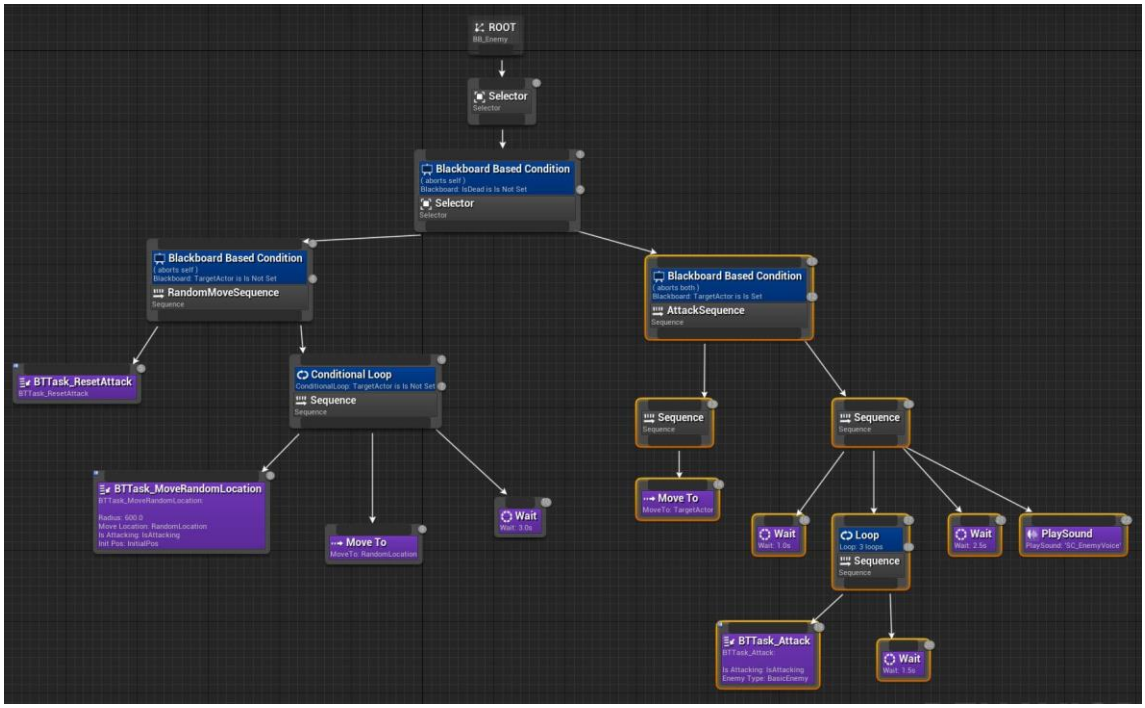

Figura 168. BT_Enemy

Enemy_Teleport

L'*Enemy_Teleport* té associat el BT que es mostra a la Figura 169. Aquest *Behavior Tree* és semblant a l'anterior, però, en aquest cas, la primera gran branca no conté el node *ResetAttack*, així que si torna a atacar, continua amb l'últim projectil pendent. Per altra part, la segona branca (on es comprova si el *TargetActor* té valor) espera dos segons per atacar, però abans de fer-ho es transporta a un punt proper del jugador, espera 1 segon i ataca amb un projectil. Per últim, s'executa un *delay* 3.5 segons. Quan acaba es torna a repetir la seqüència des del teletransport en bucle fins que s'avorti.

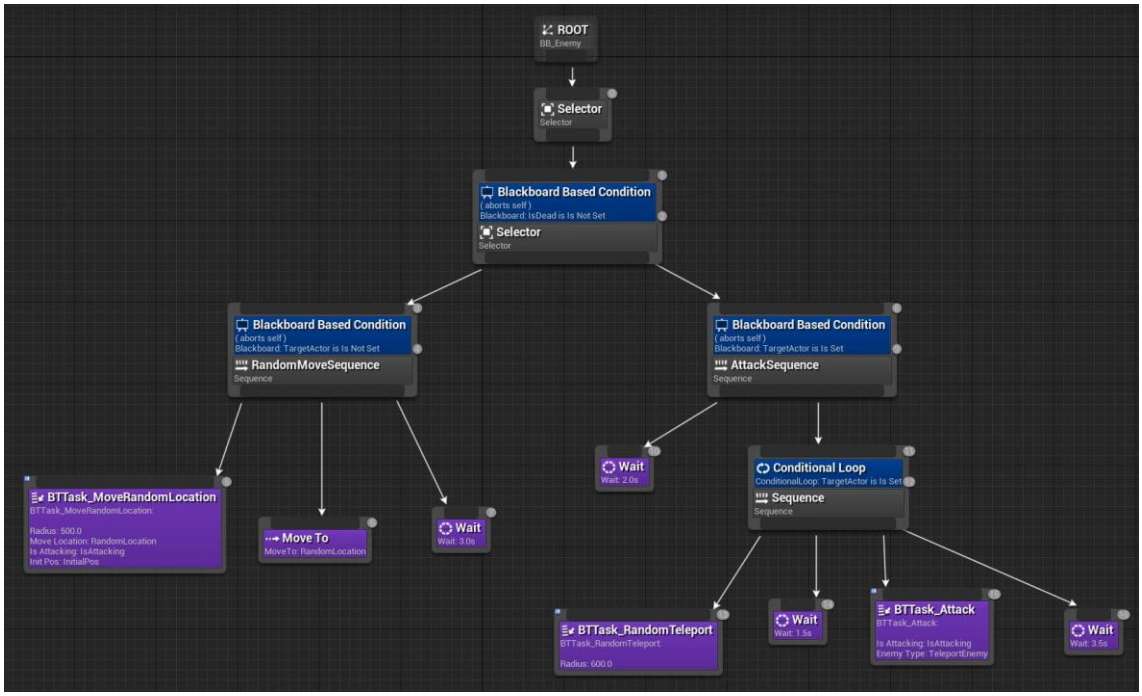

Figura 169. BT_EnemyTeleport

Enemy_Homing

El BT de l'*Enemy_Homing* (Figura 170) és el mateix que el de l'*Enemy_Basic*, excepte els nodes *Wait* que s'han augmentat per a facilitar el combat al jugador, ja que, encara que aquest enemic tingui el mateix comportament que el primer, té més vida i és més difícil esquivar els seus projectils.

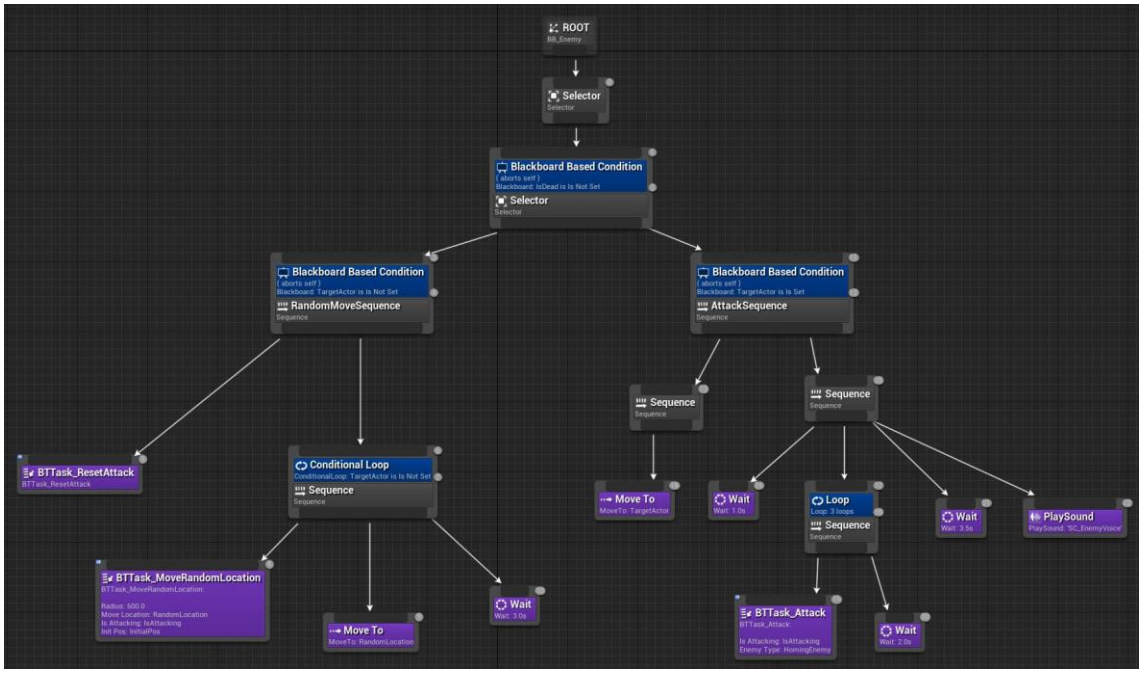

Figura 170. BT_EnemyHoming

Task ResetAttack

La *Task ResetAttack* (Figura 171) crida a l'esdeveniment *RestartFlames* de l'enemic, el qual reinicia el número de projectils.

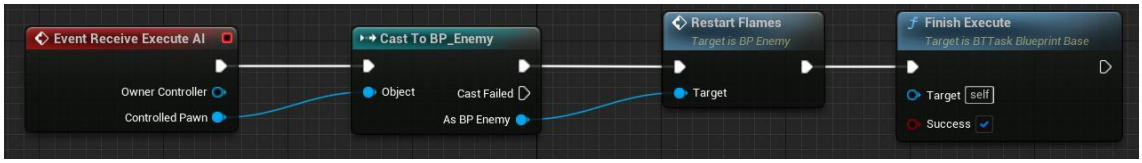

Figura 171. Task ResetAttack

Task MoveRandomLocation

MoveRandomLocation (Figura 172) és la tasca encarregada de trobar una posició aleatòria en la que l'enemic es pugui moure dins d'un cert radi des de la seva posició inicial, això es fa gràcies ala funció *GetRandomReachablePointInRadius*. A més es canvia el valor de la variable *IsAttacking* a *false* per a indicar que el fantasma no veu al jugador.

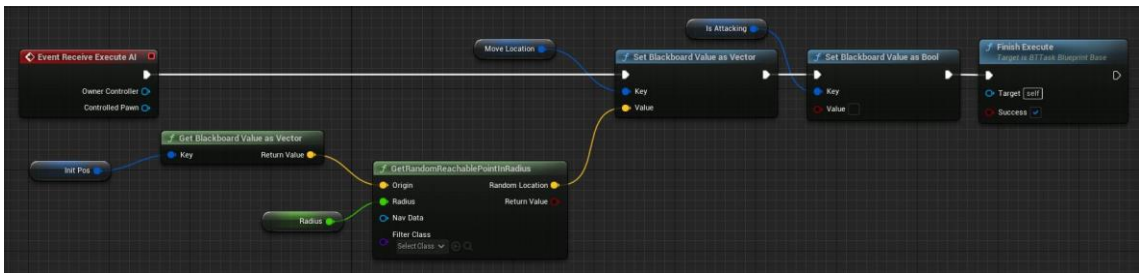

Figura 172. Task MoveRandomLocation

Task Attack

Task Attack (Figura 173) és l'encarregada d'indicar que l'enemic està atacant i cridar a l'esdeveniment *FireFlames*.

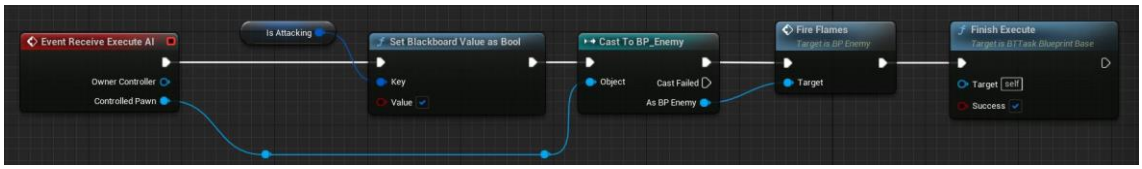

Figura 173. Task Attack

Task RandomTeleport

RandomTeleport (Figura 174) és semblant a *MoveRandomLocation*, però en aquest cas la posició aleatòria a retornar és al voltant del jugador. A més, just en aquest instant, es reprodueix un so.

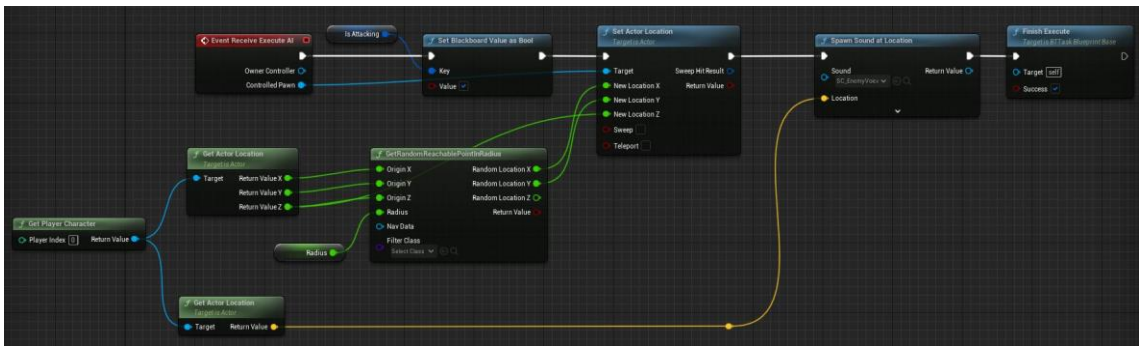

Figura 174. Task RandomTeleport

6.4.4. Flames

Les flames són els projectils que disparen els enemics cap al jugador. Cadascun dels enemics en té tres, i els recarrega segons els llença.

Al igual que passa amb els enemics, també s'ha creat un *blueprint* pare amb tres fills, un per cada fantasma. Aquest *blueprint* s'anomena *BP_Flame*, a la Figura 175 es pot veure una visió general.

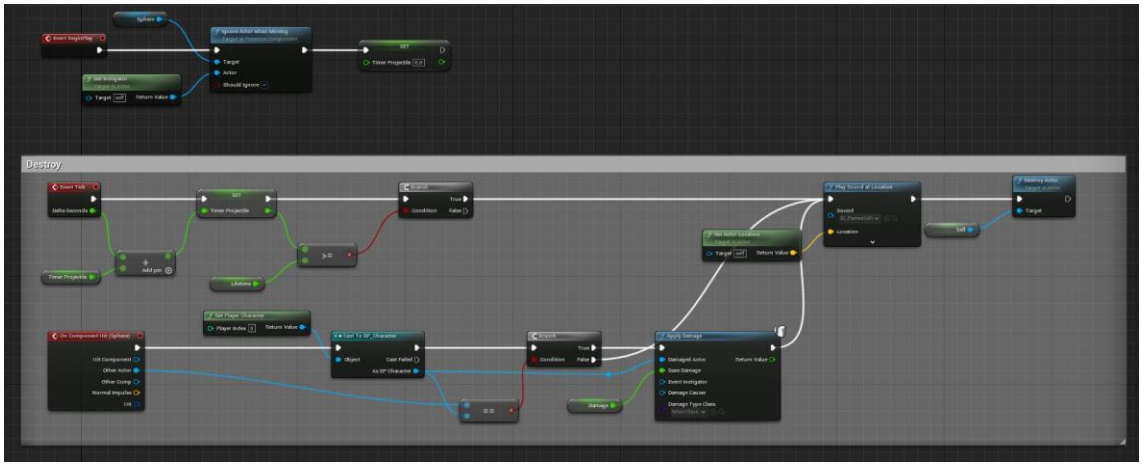

Figura 175. Visió global de BP_Flame

Com aquesta flama és un projectil, s'ha hagut d'afegir el component *ProjectileMovement* (Figura 176), que li aplica moviment i velocitat a l'actor.

Figura 176. Components BP_Flame

Inicialització

A l'esdeveniment *BeginPlay* (Figura 177) s'indica que el *collider Sphere,* que representa a la flama, no xoqui contra l'*instigator*, és a dir, l'actor que l'ha instanciat, en aquest cas, l'enemic. A més s'inicialitza la variable *TimerProjectile* a 0.

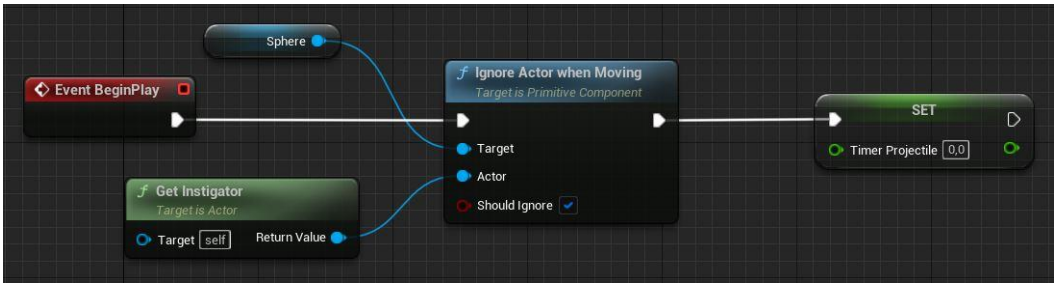

Figura 177. BeginPlay de BP_Flame

Event Tick

L'*Event Tick* (Figura 178 i Figura 179) controla el temps des de que el projectil s'ha disparat. Quan supera els 4 segons, es reprodueix un so d'apagat i es destrueix la flama.

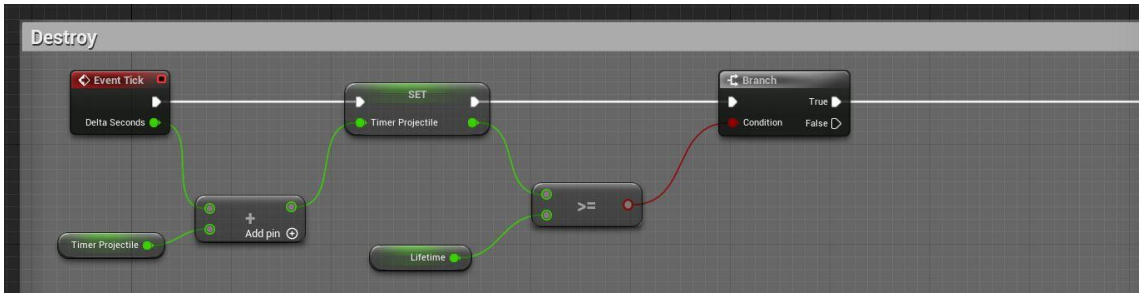

Figura 178. EventTick de BP_Flame (primera part)

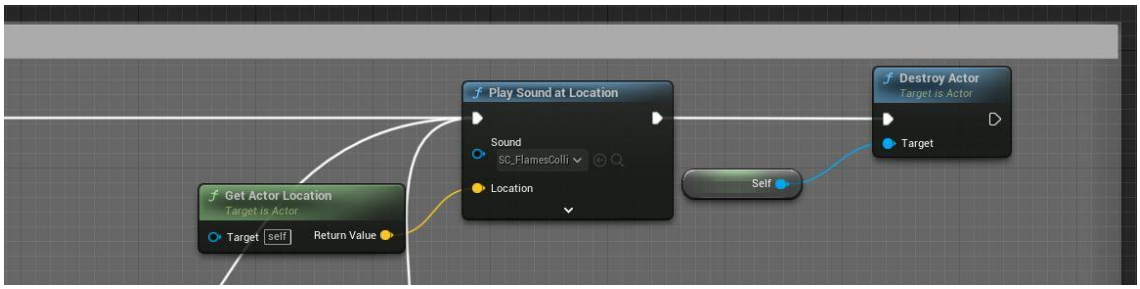

Figura 179. EventTick de BP_Flame (segona part)

Col·lisions

Quan el *collider Sphere* col·lisiona amb un actor, s'inicia l'esdeveniment *OnComponentHit* (Figura 180 i Figura 181). Aquest mètode comprova que s'hagi col·lisionat amb el jugador, si és així, es crida a la funció *ApplyDamage* per a que se li resti 1 vida. A continuació, es reprodueix el mateix so que l'apartat anterior i es destrueix la flama, encara que això ocorre tant si l'actor amb el que s'ha xocat ha estat el jugador o no.

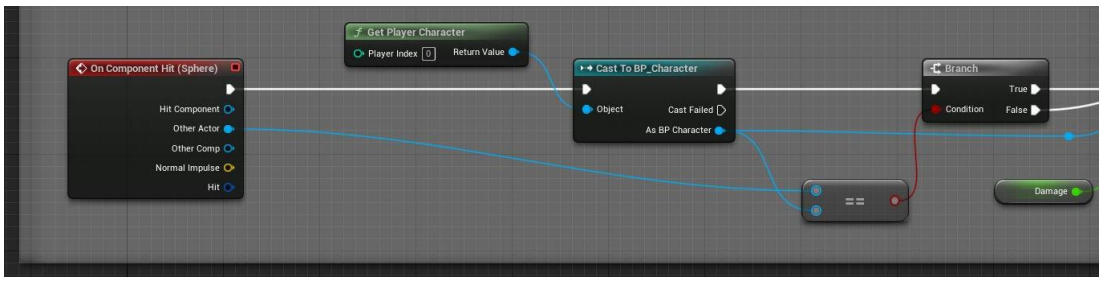

Figura 180. OnComponentHit de BP_Flame (primera part)

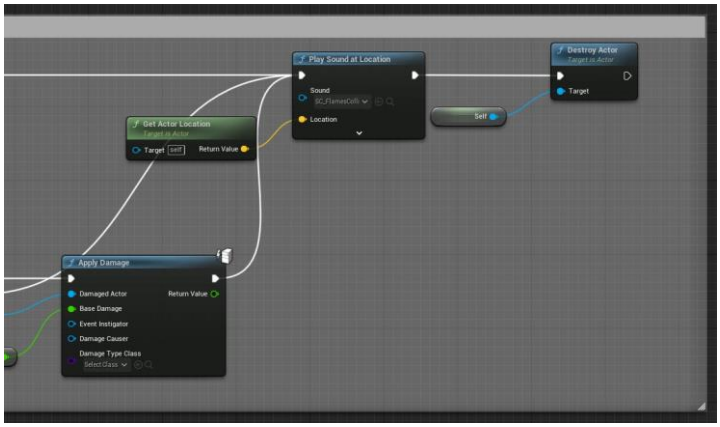

Figura 181. OnComponentHit de BP_Flame (segona part)

FlameRed

La flama vermella està implementada a *BP_FlameRed* (Figura 182). Aquest actor no presenta cap diferència respecte el pare.

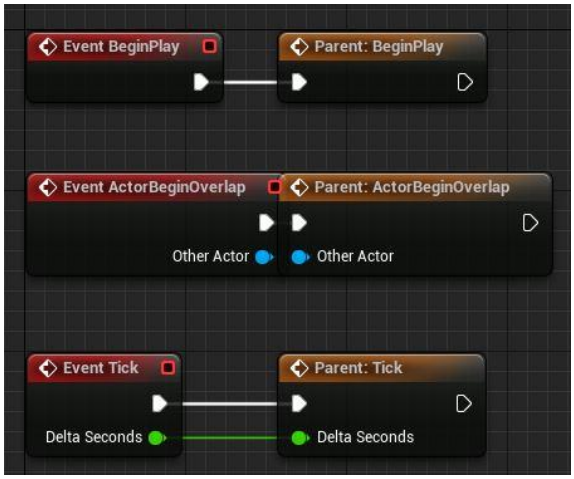

Figura 182. BP_FlameRed

FlamePurple

La flama lila, que correspon al segon grup d'enemics, està implementada a *BP_FlamePurple* (Figura 183). Aquest actor tampoc presenta cap diferència respecte el pare en la implementació, però sí que s'han canviat les partícules (Figura 184), així que el canvi és només visual.

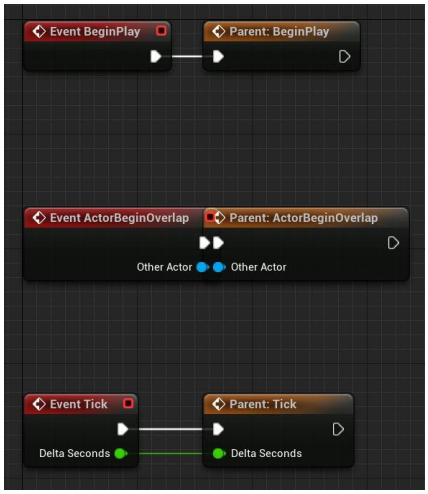

Figura 183. BP_PurpleFlame

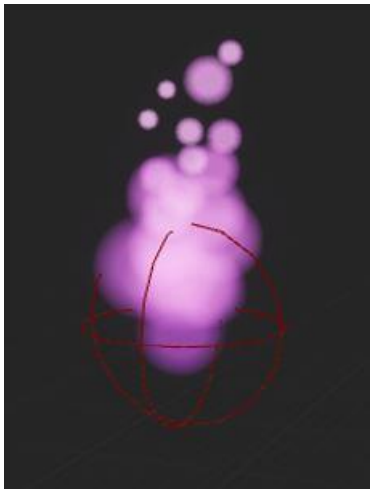

Figura 184. Flama lila

FlameHoming

La flama blava que persegueix al jugador s'ha implementat al *blueprint BP_FlameHoming* (Figura 185). En aquest cas, com es pot veure, l'esdeveniment *BeginPlay* sí ha estat sobreescrit pel fill.

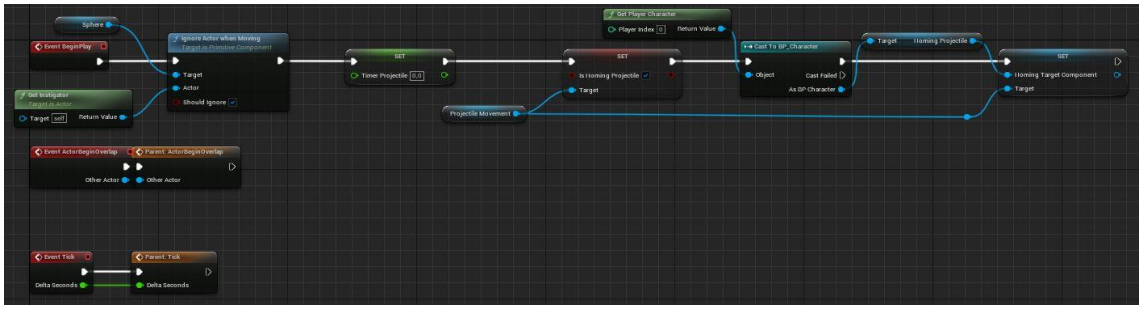

Figura 185. BP_FlameHoming

Inicialització

El primer que fa l'esdeveniment *BeginPlay* (Figura 186 i Figura 187) és ignorar l'actor que l'ha instanciat; inicialitza la variable *TimerProjectile* a 0; li indica al component *ProjectileMovement* que és un *Homing Projectil*, és a dir, que és un projectil que persegueix a l'objectiu, que en aquest cas, és un component del personatge anomenat també *Homing Projectile*, creat per a aquesta flama.

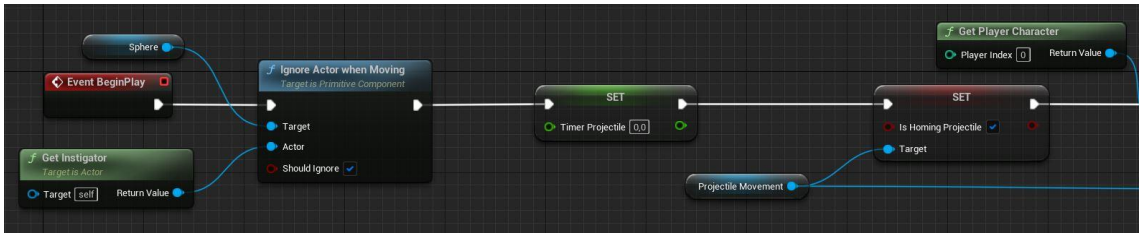

Figura 186. BeginPlay de BP_FlameHoming (primera part)

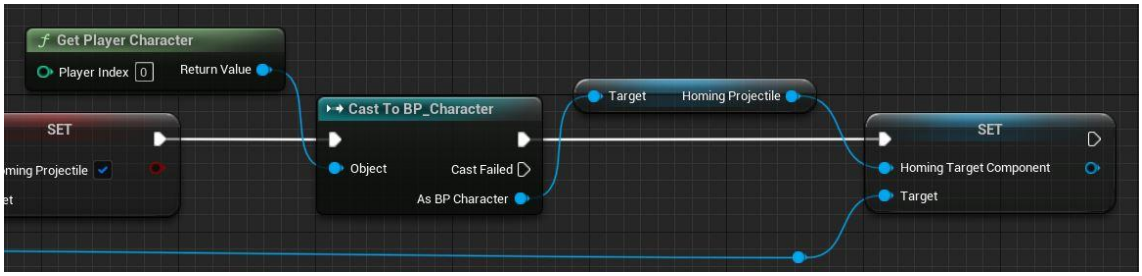

Figura 187. BeginPlay de BP_FlameHoming (segona part)

6.4.5. NavMesh

Un altre element molt important per a aconseguir que els enemics es moguin pel terreny sense problemes, és el *NavMesh*. El *NavMesh*, és un actor del propi *Unreal* que permet calcular un camí que indica quines zones de l'escenari estan disponibles i són accessibles per a que, en aquest cas, els enemics es puguin moure per elles. Per a utilitzar aquest actor, s'ha d'afegir l'objecte anomenat *NavMeshBoundsVolume* a l'escena i s'ha d'escalar tant com sigui necessari per a que abasti tot el mapa. Per a comprovar quines zones s'han calcular com accessibles, es pot prémer la tecla *P* del teclat, així es mostren en verd les zones disponibles per l'enemic, tal i com es pot veure a la Figura 188.

Figura 188. Àrea del NavMesh

6.4.6. Grup d'enemics

Per a facilitar la instanciació dels diferents enemics i les recompenses, s'ha implementat el *blueprint BP_EnemyGroup.* Aquest actor es col·loca a la zona del mapa on es volen instanciar un o més enemics. A més se li indica de quin tipus d'enemic es tracta, quants es volen crear i la recompensa des del panell de detalls a l'escena. Tot això es troba a les variables, majoritàriament públiques, del *blueprint* (Figura 189).

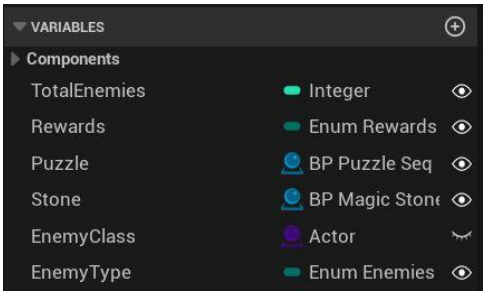

Figura 189. Variables de BP_EnemyGroup

Construction Script

Abans de que es comencin a executar els esdeveniments del graf principal, s'inicia el *Construction Script* (Figura 190), on es guarda la classe de l'enemic que es vol instanciar segons el tipus escollit.

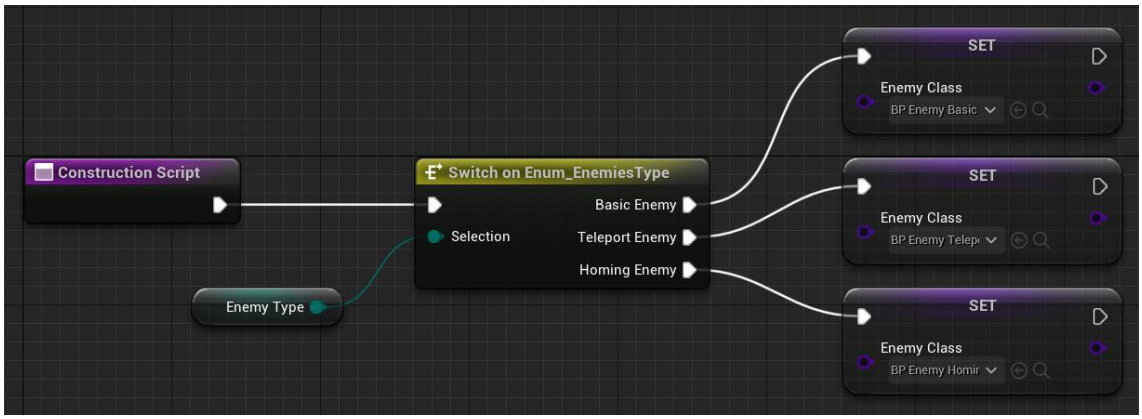

Figura 190. Construction Script de BP_EnemyGroup

Inicialització

A l'esdeveniment *BeginPlay* (Figura 191 i Figura 192), el primer que es fa és instanciar un enemic de la classe indicada a la variable *EnemyClass* utilitzant el node *SpawnActor*; això es fa tants cops com el valor de *TotalEnemies* indiqui. A continuació, segons el tipus d'enemic escollit al desplegable i guardat a la variable *EnemyType*, es crea una referència al propi *blueprint* que es guarda a l'*EnemyGroupRef* de l'enemic instanciat.

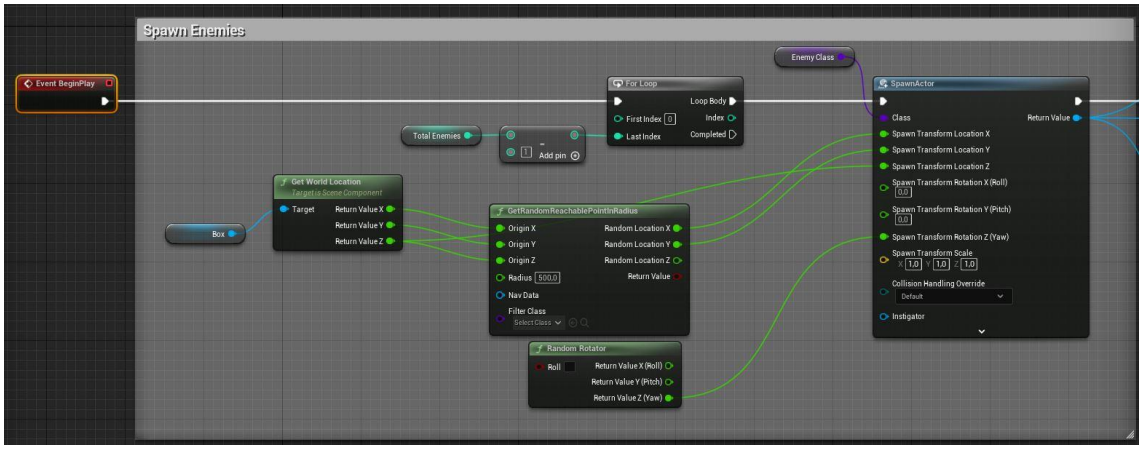

Figura 191. BeginPlay de BP_EnemyGroup (primera part)

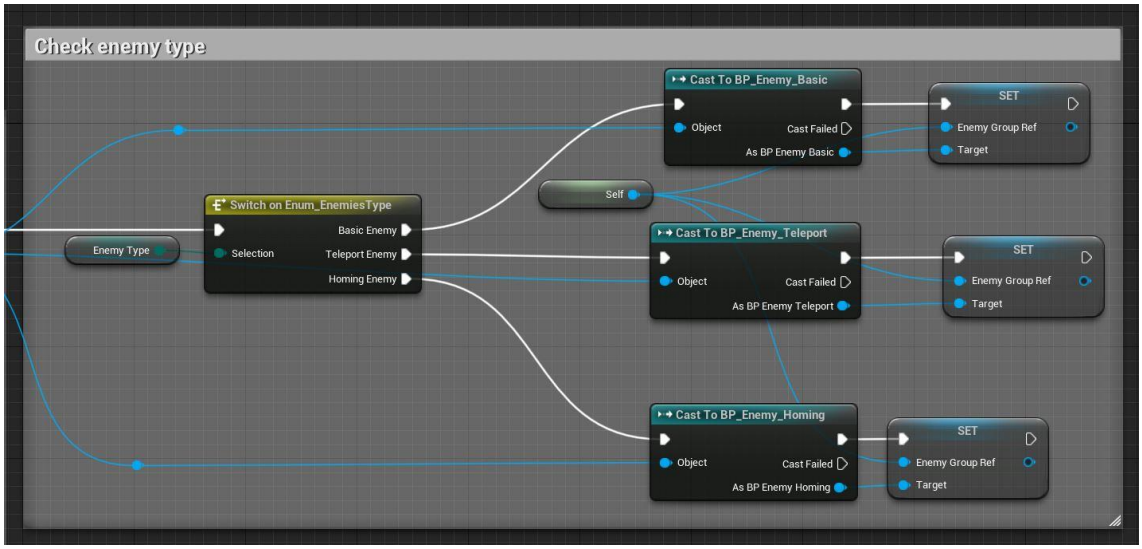

Figura 192. BeginPlay de BP_EnemyGroup (segona part)

ChangeEnemiesAlive

La funció *ChangeEnemiesAlive* (Figura 193) s'executa des de *BP_Enemy* cada cop que un enemic mor. Aquesta funció comprova quants enemics del grup creat queden vius i, si s'han eliminat tots, es crida a la funció *GiveRewards.*

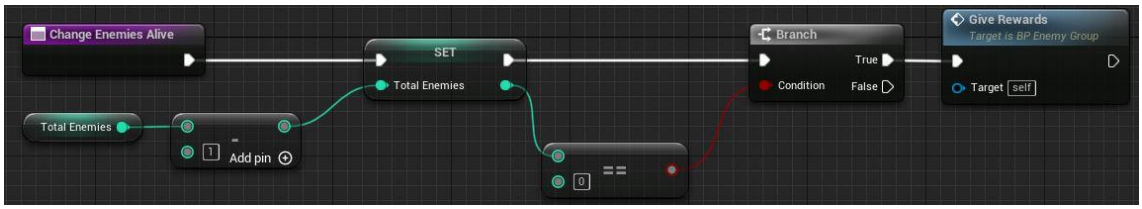

Figura 193. Funció ChangeEnemiesAlive

GiveRewards

La funció *GiveRewards* (Figura 194) s'encarrega de, segons la recompensa escollida des del panell de detalls i guardada a la variable *Rewards*, cridar a les funcions oportunes o crear els actors que toquen. Si la recompensa és la pedra màgica, es crida a l'*Activate Stone*; si és el pètal, s'instancia un actor, i si s'ha eliminat l'últim grup d'enemics, es desbloqueja el puzle amb la funció *Unlock.*

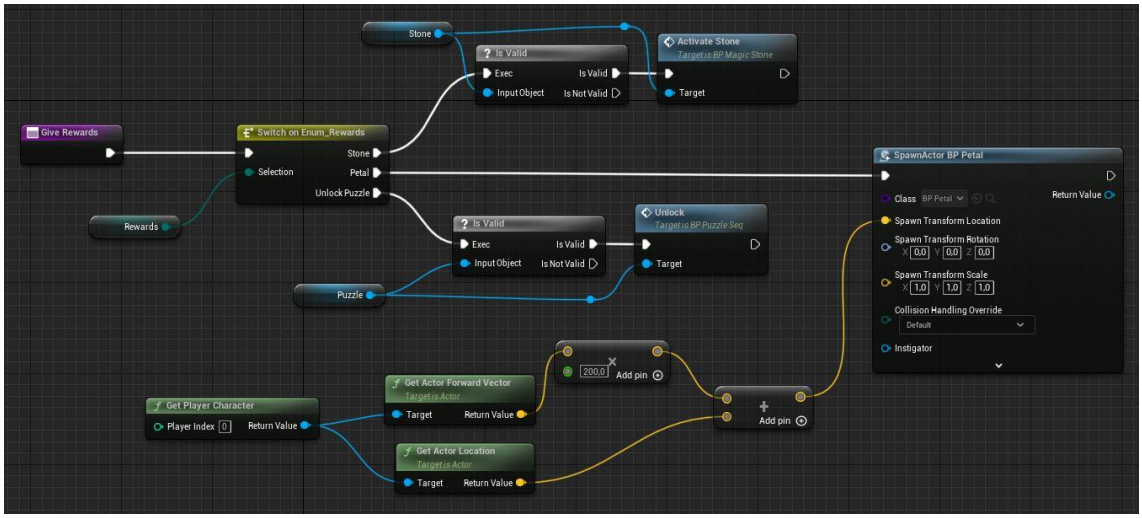

Figura 194. Funció GiveRewards

Variables Enum

Per a poder seleccionar el tipus d'enemics i recompenses al panell de detalls del *blueprint*, s'han creat les variables de tipus Enum. Aquestes variables necessiten d'un arxiu *Enumeration* (Figura 195 i Figura 196) que simplement defineix la llista de possibilitats.

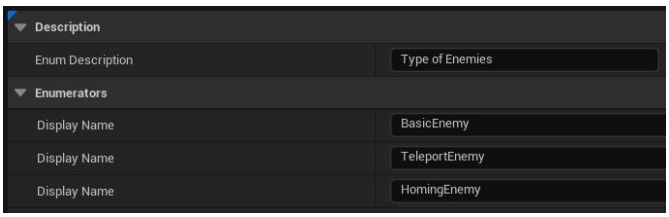

Figura 195. Enumeration Type of Enemies

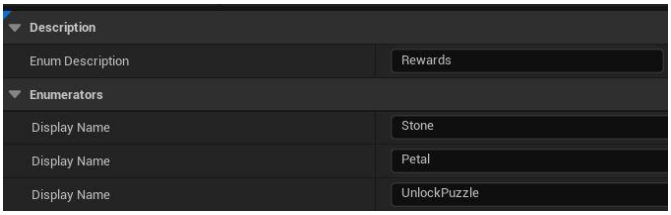

Figura 196. Enumeration Rewards

6.5. Objectes

6.5.1. Arma

El *blueprint BP_Weapon,* que es mostra a la Figura 197, conté la implementació de l'arma, la qual és necessària per a poder avançar a noves zones del joc i eliminar als enemics que es troben pel mapa.

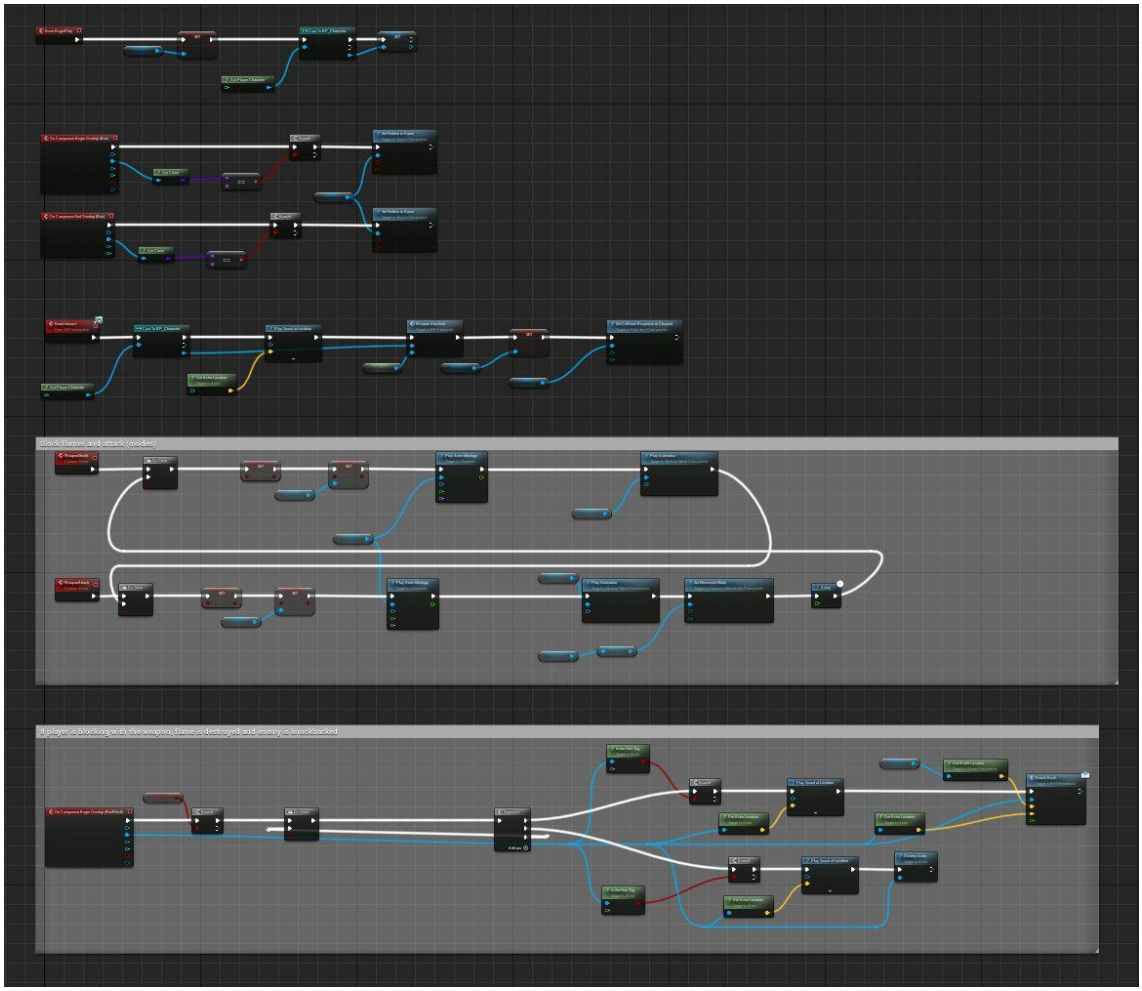

Figura 197. Visió general del graf de BP_Weapon

Components

Per a assegurar el correcte funcionament de la interacció, atac i bloqueig, l'arma disposa, com es pot veure a la Figura 198, de dos *colliders* amb una funcionalitat diferent per a cadascun i un component *widget* que indica al jugador que és possible interaccionar amb l'objecte. Aquest últim s'explica amb més detall a l'Apartat 6.6.2.

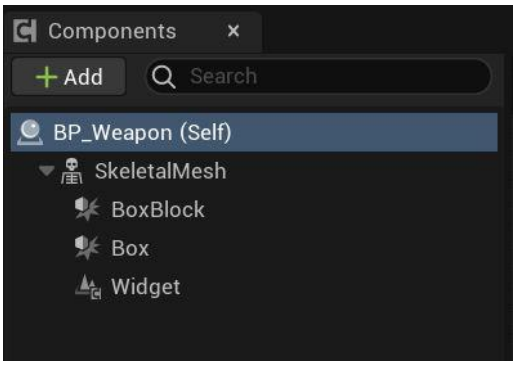

Figura 198. Components de l'arma

Sockets

A més, un detall important de l'*skeletal mesh* al que s'ha fet referència a l'Apartat 6.3.7 és el dels *sockets*. Com es pot veure a la Figura 199, s'han creat dos *sockets*, un a cada punta de l'arma. Amb aquests s'ha pogut traçar la línia amb la qual es controla si s'ha impactat amb l'enemic o no.

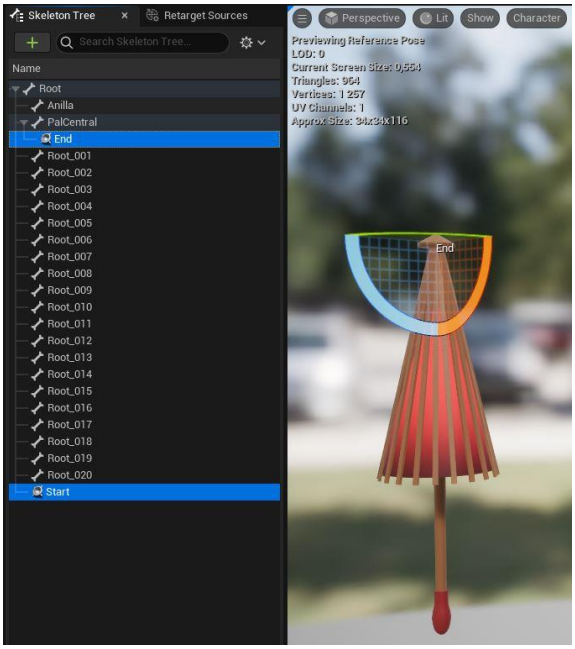

Figura 199. Sockets de l'arma

Inicialització

A l'esdeveniment *BeginPlay* (Figura 200) se li indica al *collider BoxBlock* que no generi esdeveniments al col·lisionar amb cap element, ja que aquest només s'activa quan el jugador acciona el bloqueig, i es guarda una referència del *BP_Character* (el personatge).

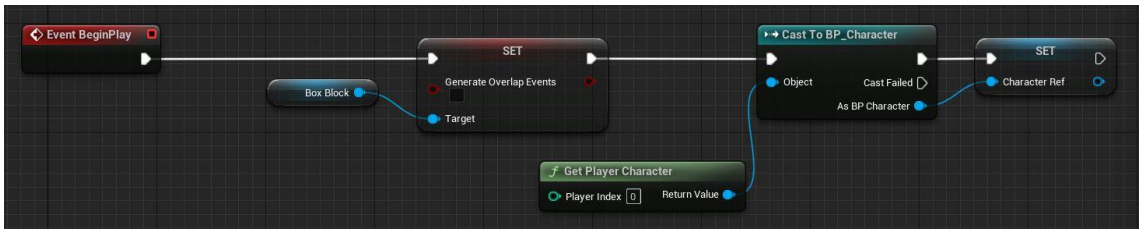

Figura 200. Esdeveniment BeginPlay de de BP_Weapon

Col·lisions

Per a que el *widget*, que inclou un missatge indicant el botó que el jugador ha de prémer, es mostri quan el jugador s'apropa, es criden als esdeveniments *OnComponentBeginOverlap i OnComponentEndOverlap* (Figura 201). Aquests esdeveniments s'executen quan un altre actor entra o surt de l'espai que ocupa el *collider*. En aquest cas, si l'actor és el personatge, es mostra o oculta el missatge d'interacció, segons si aquest entra o surt del *Box*.

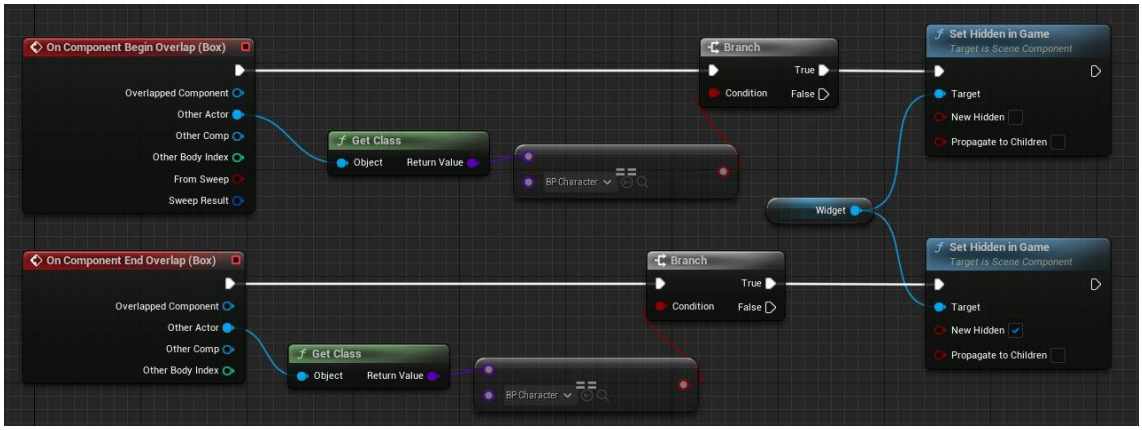

Figura 201. Overlaps de BP_Weapon

Interacció

Quan el jugador interacciona amb l'arma, s'executa la interfície *BPI_Interaction* (Figura 202 i Figura 203) que s'ha comentat en apartats anteriors. L'esdeveniment *Interact* de la interfície *BPI_Interaction* s'ha implementat per a que reprodueixi un so de recompensa a l'hora d'agafar l'arma per primer cop. També es crida a l'esdeveniment *WeaponAttached* comentat en l'Apartat 6.3.6, el qual adjunta el paraigües a la mà del personatge. Per acabar, es modifiquen les col·lisions; es desactiva la generació d'esdeveniment del *collider Box,* ja que no es torna a utilitzar, i a les configuracions de l'*skeletal mesh* se li indica que ignori als actors *Pawn*, com el personatge.

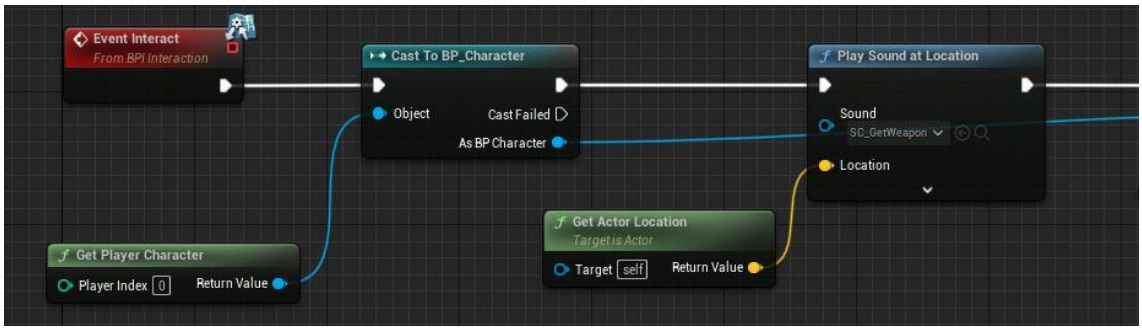

Figura 202. Esdeveniment Interact (primera part)

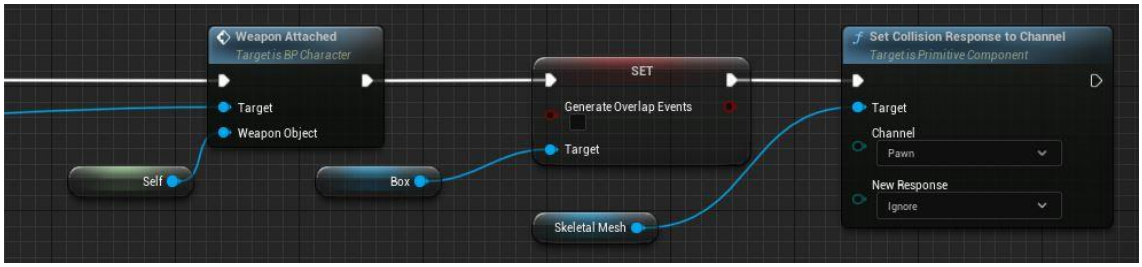

Figura 203. Esdeveniment Interact (segona part)

Bloquejar

Per a poder controlar si l'arma està bloquejant o llesta per a atacar, s'han definit els dos esdeveniments que es mostren en la Figura 204, els quals s'han nombrat anteriorment a la implementació del personatge. S'ha decidit que un depengui de l'altre, d'aquesta

manera les diferents animacions no es solapen, pel que no creen moviments i comportaments bruscos.

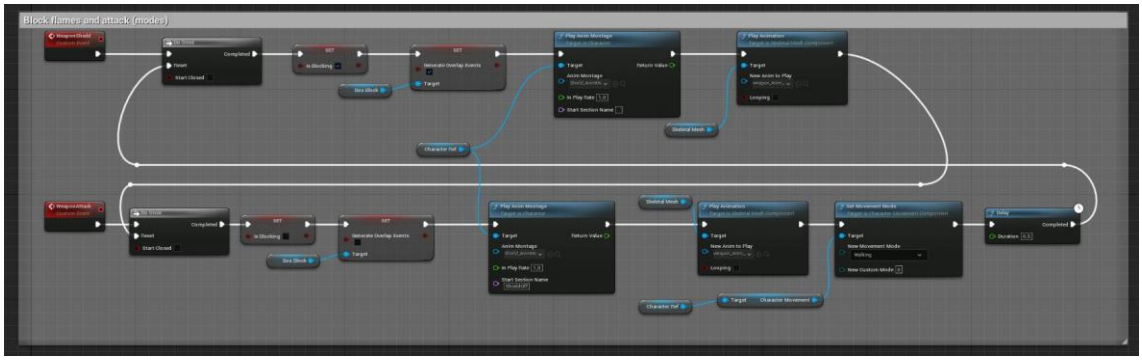

Figura 204. Visió general dels esdeveniments que controlen el bloqueig i atac

WeaponShield (Figura 205 i Figura 206) s'executa quan s'acciona el bloqueig. Aquest esdeveniment canvia el valor de la variable *IsBlocking* a *true* (inicialment es troba a *false*), activa el *collider BoxBlock,* inicia l'animació del personatge i seguidament la del paraigües, ja que aquest es troba tancat quan està en repòs. Un cop acaben les animacions, es permet que l'arma es pugui tancar i atacar.

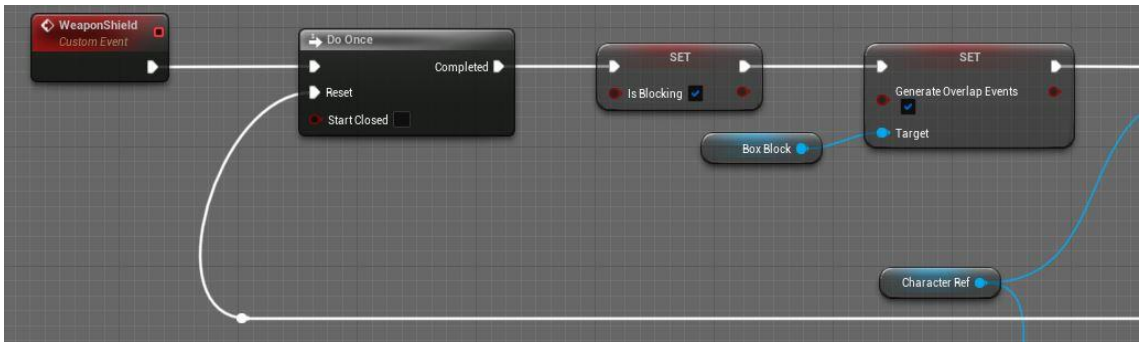

Figura 205. Esdeveniment WeaponShield (primera part)

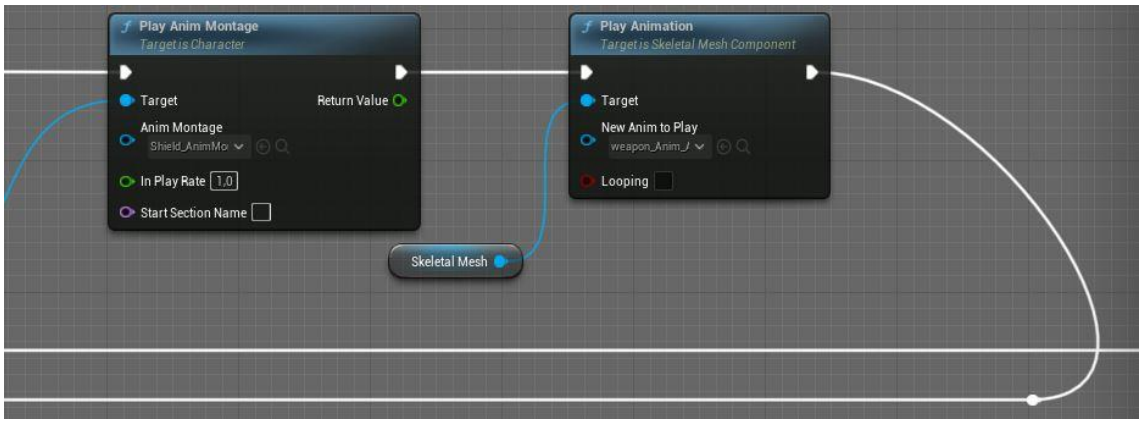

Figura 206. Esdeveniment WeaponShield (segona part)

Per altra part, l'esdeveniment *WeaponAttack* (Figura 207 i Figura 208) fa el contrari a l'anterior, l'*IsBlocking* canvia a *false* i el *collider BoxBlock* deixa de col·lisionar. D'aquesta manera s'indica que el personatge està llest per a tornar a atacar. Seguidament s'acaba de reproduir la segona part de l'animació de bloqueig del personatge (s'havia quedat

pausada mentre el bloqueig està actiu). Això s'indica amb el paràmetre *Start Section Name,* que permet iniciar l'animació en un punt indicat, com es pot veure a la Figura 209; també es mostra l'animació de l'arma plegant-se. A més es torna a indicar el mode de moviment del personatge, ja que al bloquejar es para. Per últim, s'aplica un petit *delay* per a que, si el jugador vol tornar a bloquejar, no ho faci immediatament, just després es fa un *reset* del primer esdeveniment per a que sigui possible tornar-lo a executar.

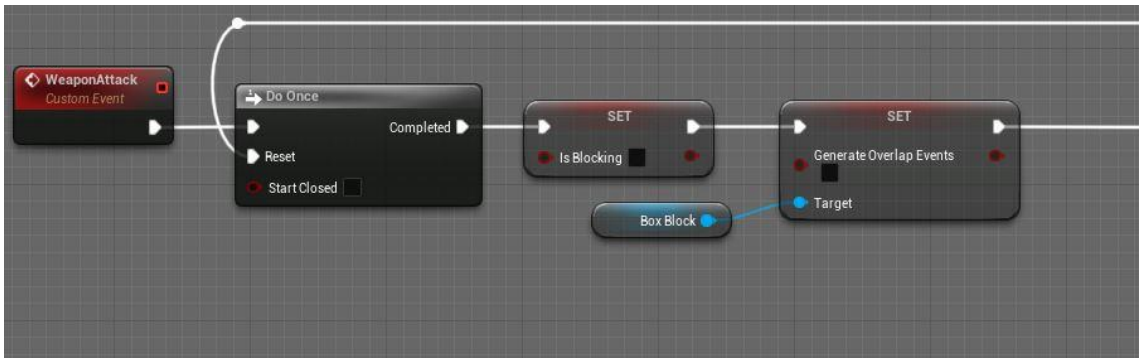

Figura 207. Esdeveniment WeaponAttack (primera part)

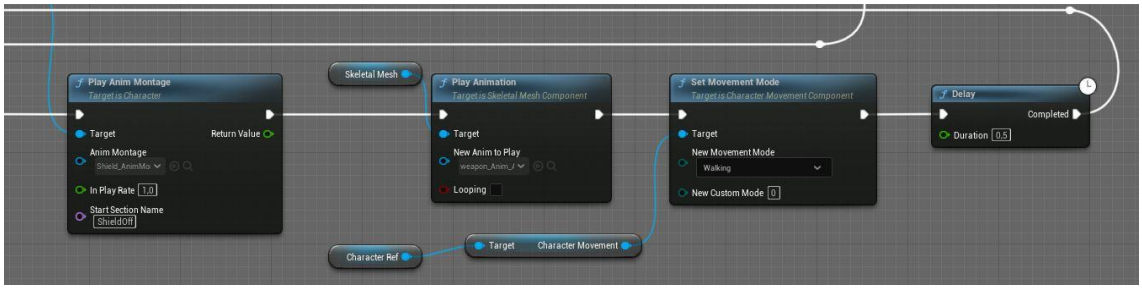

Figura 208. Esdeveniment WeaponAttack (segona part)

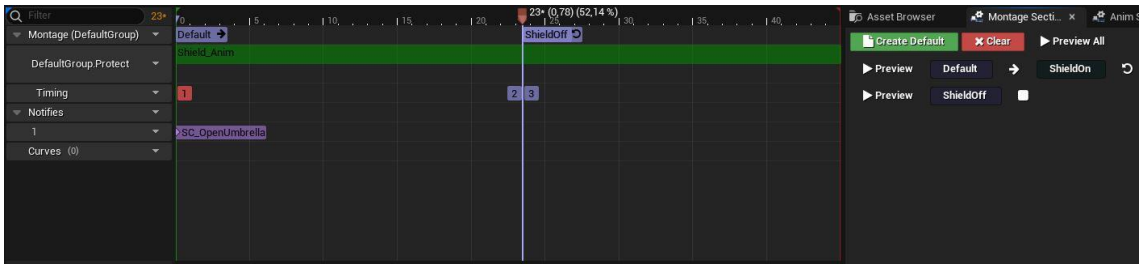

Figura 209. Línia de temps de l'animació de bloqueig del personatge

Mentre el personatge està bloquejant i el paraigües es manté obert, en cas que algun actor es solapi amb el *collider BoxBlock,* s'executa l'esdeveniment *OnComponentBeginOverlap* del component esmentat, tal i com es pot veure a la Figura 210 i Figura 211.

Si l'actor que xoca amb el *collider* té un *tag* anomenat *Enemy*, es crida a l'esdeveniment *Knockback* de la interfície *BPI_Knockback*, la qual s'utilitza per empènyer a l'enemic, tot i així, s'explica amb més profunditat a l'Apartat 6.4.1. Per altra part, si el *tag* és *Flame*, vol dir que l'actor és un projectil, així que es reprodueix un so i es destrueix l'actor.

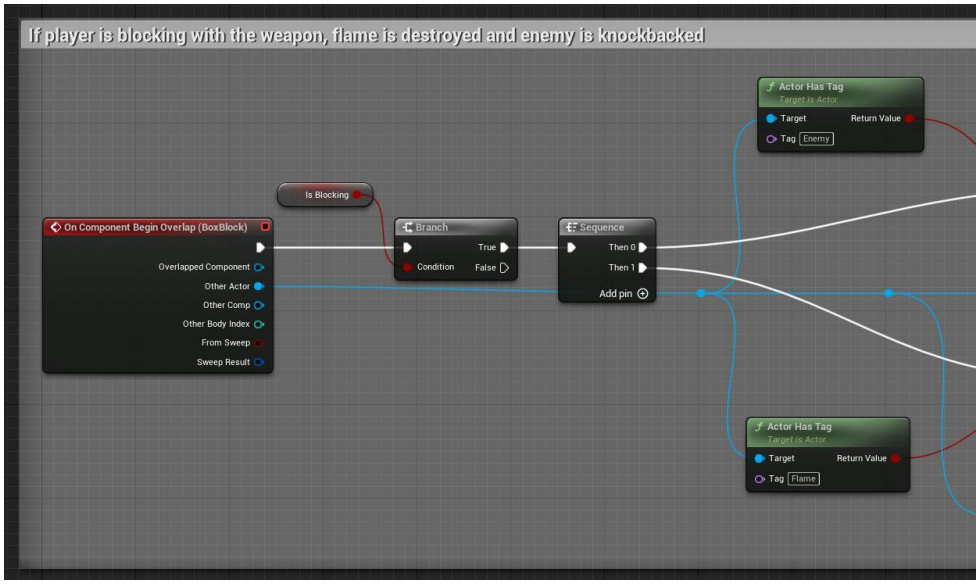

Figura 210. Bloqueig actor (primera part)

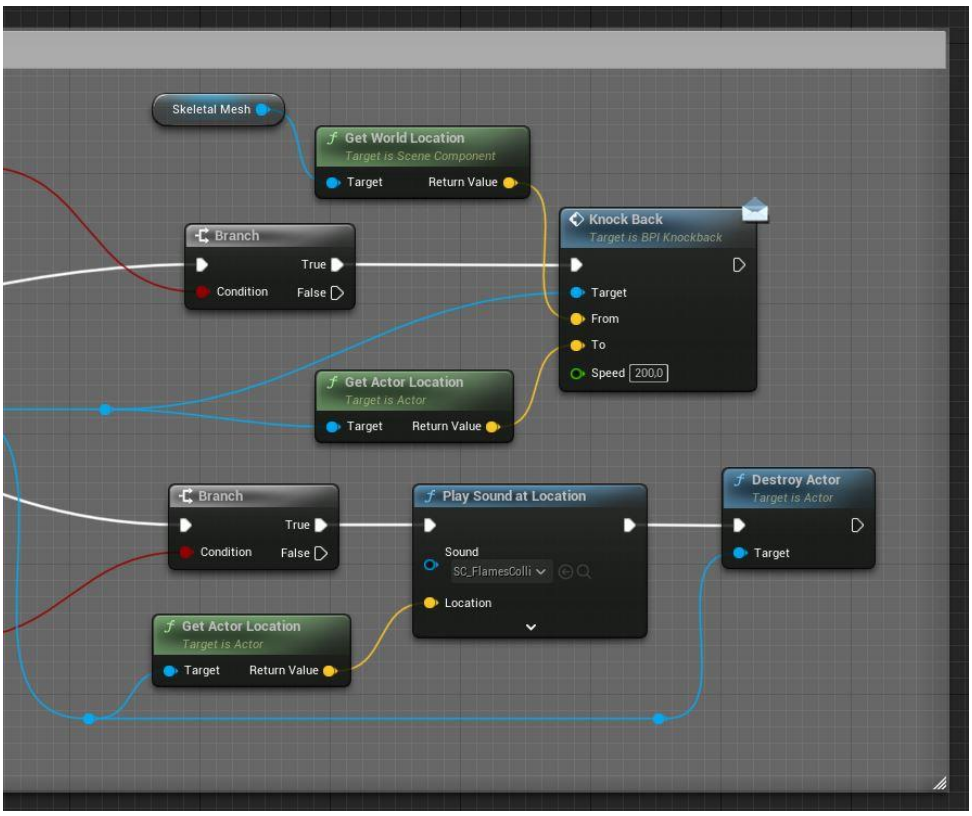

Figura 211. Bloqueig actor (segona part)

6.5.2. Esperits ajudants

La implementació dels petits esperits que es troben al llarg del mapa i ajuden al jugador, es troba tant al *blueprint BP_ForestSpirits* com al *BP_ForestSpiritControls*.

Esperit amb controls

Al principi del joc es poden trobar dos esperits a prop de l'arma, els quals expliquen alguns dels controls bàsics. En la Figura 212 es pot veure una visió global dels diferents esdeveniments.

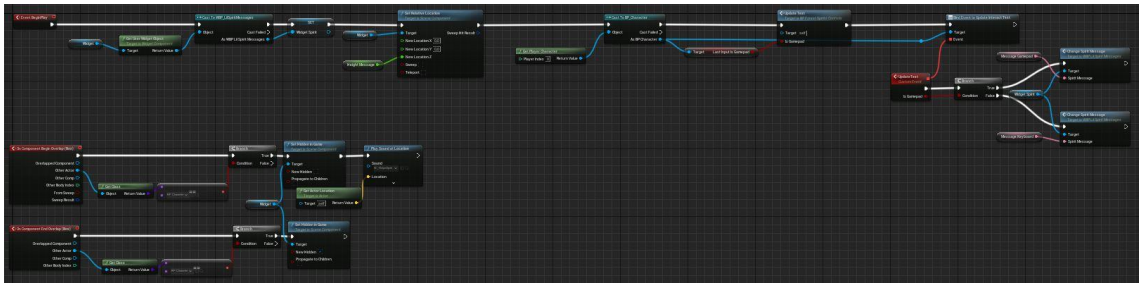

Figura 212. Visió global de BP_ForestSpiritControls

Primer de tot, a l'esdeveniment *BeginPlay* (Figura 213 i Figura 214) es guarda el *widget;* afegit com a component; en una variable, això facilita l'accés. A aquest widget se li assigna l'alçada desitjada, d'aquesta manera, segons les dimensions del missatge, es pot evitar que tapi el model de l'actor. A continuació, es crida a l'esdeveniment UpdateText, que es troba definit al final del graf. Aquest esdeveniment canvia el text del missatge, segons si s'està utilitzant teclat o comandament, gràcies al mètode *ChangeSpiritMessage* del *widget* (Apartat 6.6.2). A més, si durant la partida es canvia el dispositiu d'entrada, el node *Bind* executa, tantes vegades com es canviï el dispositiu, l'esdeveniment que s'acaba de mencionar. Cal dir que, per a facilitar la modificació de l'alçada i el contingut dels missatges, les variables *HeightMessage, MessageGamepad* i *MessageKeyboard* són públiques i es poden canviar des de l'editor.

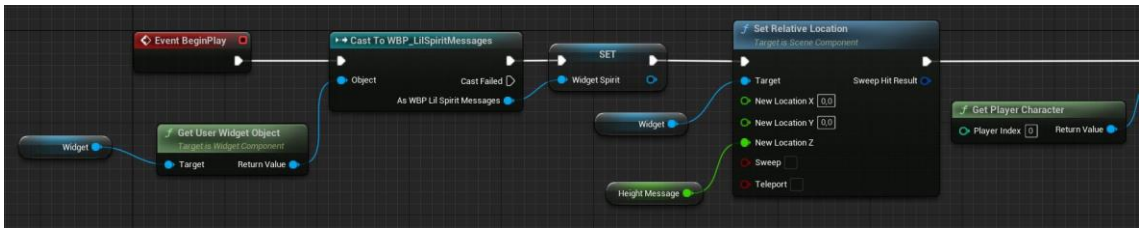

Figura 213. BeginPlay de BP_ForestSpiritControls (primera part)

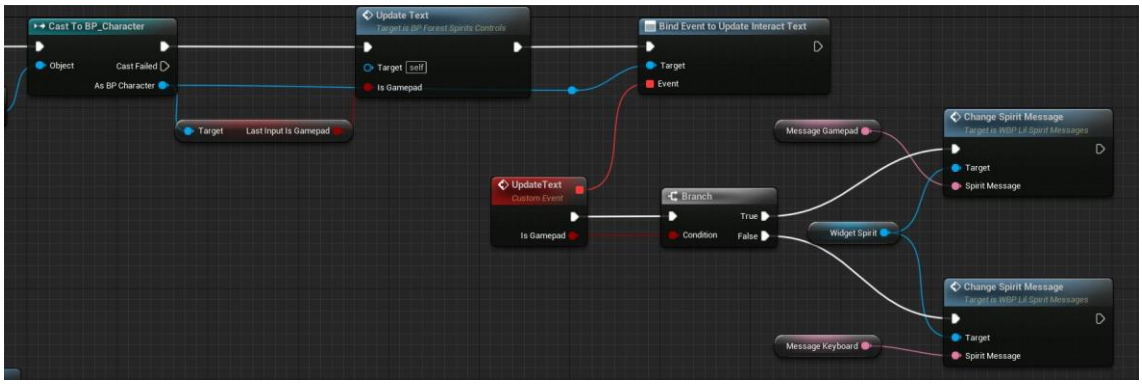

Figura 214. BeginPlay de BP_ForestSpiritControls (segona part)

Per altra part, com es pot veure en la Figura 215, si el personatge s'apropa a algun dels esperits, en concret al *collider Box*, aquest s'encarrega de fer visible el *widget* i de reproduir un so a través de l'esdeveniment *OnComponentBeginOverlap*. En canvi, si s'allunya, el missatge desapareix a conseqüència d'haver-se executat *OnComponentEndOverlap*.

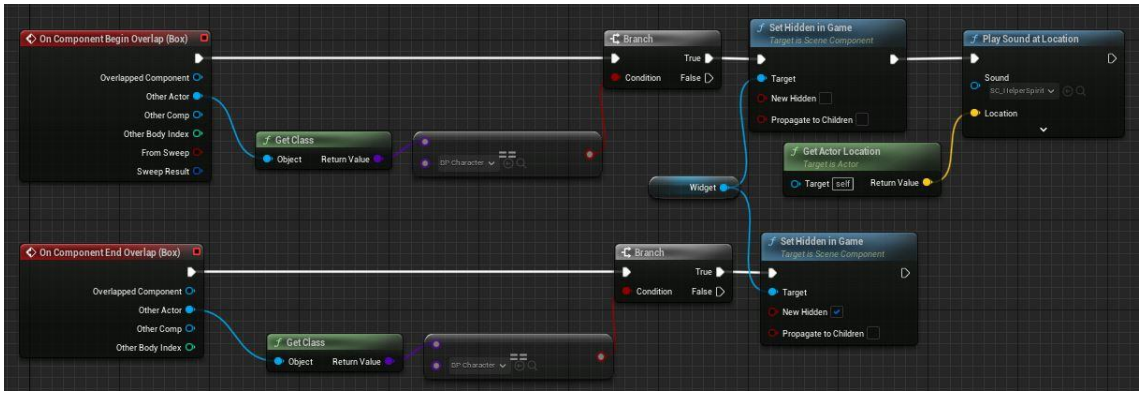

Figura 215. Overlaps de BP_ForestSpiritControls

Esperit amb pistes

En el cas del *blueprint BP_ForestSpirit* (Figura 216), la implementació és pràcticament igual, a diferència de que les pistes que es mostren amb aquests actors no estan pensades per a que incloguin controls, així que només s'ha d'introduir un tipus de missatge.

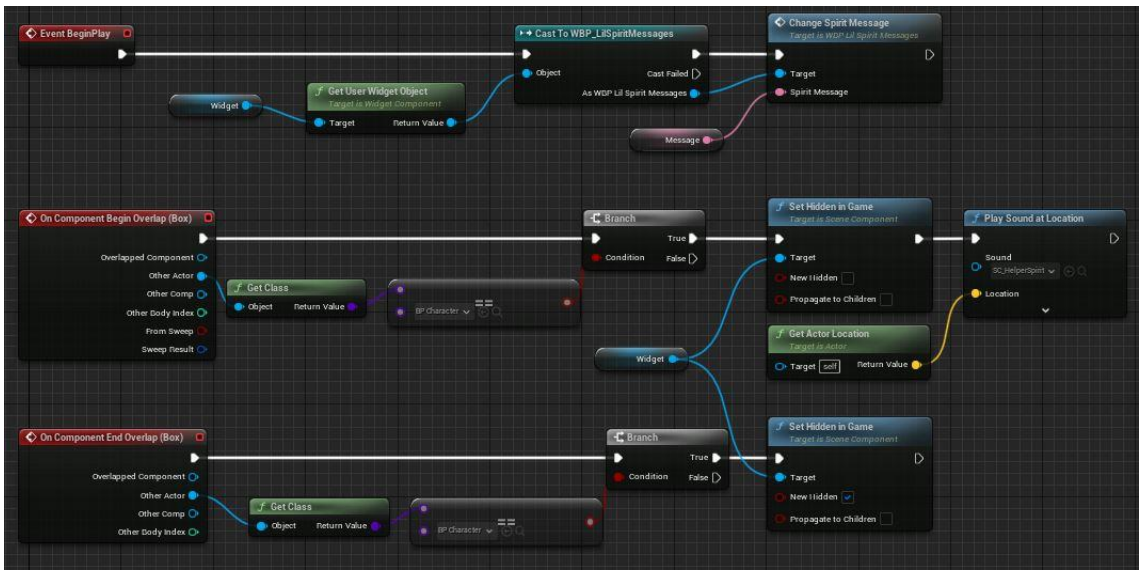

Figura 216. Visió general de BP_ForestSpirit

Al *BeginPlay* (Figura 217), semblant a com s'ha vist a l'apartat anterior, es guarda el *widget* a mostrar i es crida al mètode *ChangeSpiritMessage* per a assignar el missatge definit des de l'editor.

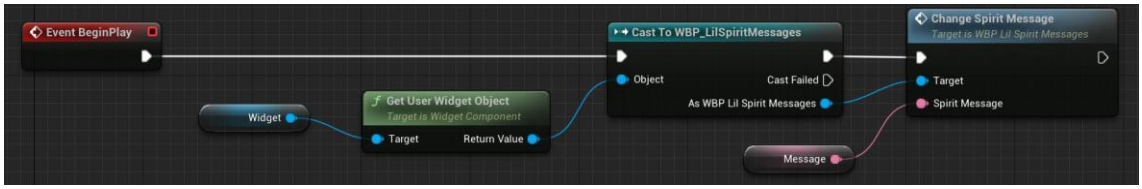

Figura 217. BeginPlay de BP_ForestSpirit

Per acabar, a l'igual que l'anterior *blueprint* i tal com es veu en la Figura 218, si el jugador s'apropa a l'esperit, es reprodueix un so i es mostra el *widget*, i si s'allunya, es torna a ocultar.

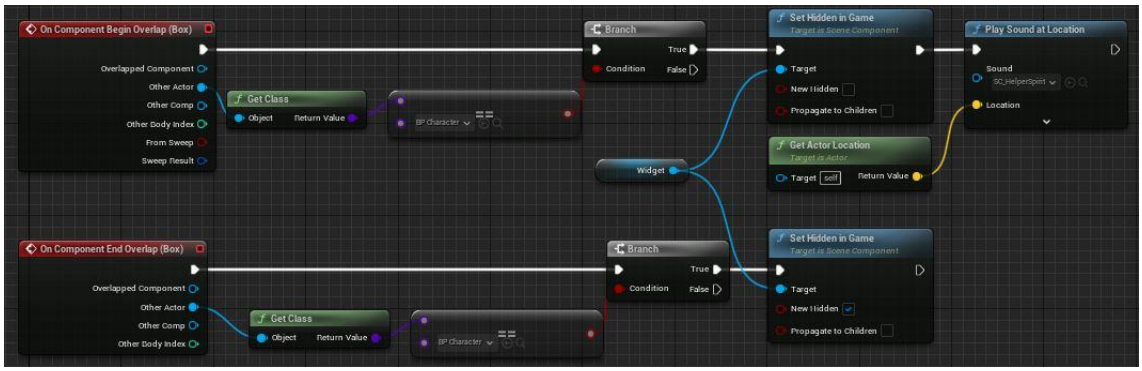

Figura 218. Overlaps de BP_ForestSpirit

6.5.3. Arc Torii

L'arc Torii és l'entrada a la segona zona del mapa. La implementació es troba al *blueprint BP_Torii* (Figura 219). A continuació, es pot veure una visió global dels esdeveniments que composen aquest *blueprint*.

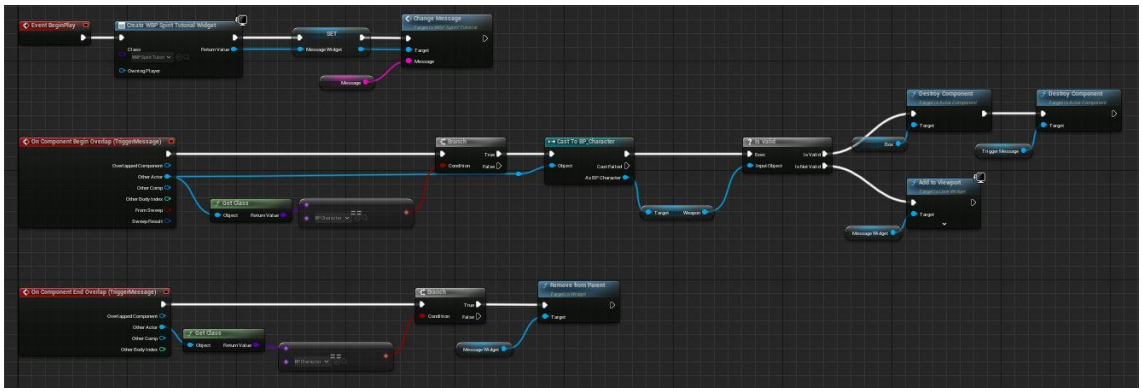

Figura 219. Visió global de BP_Torii

Inicialització

L'esdeveniment *BeginPlay* (Figura 220), el qual s'executa primer, crea el *widget WBP_SpiritTutorial* (Apartat 6.6.2) i li assigna un valor, que s'utilitza com a missatge, a través de la crida a la funció *ChangeMessage* del *widget*.

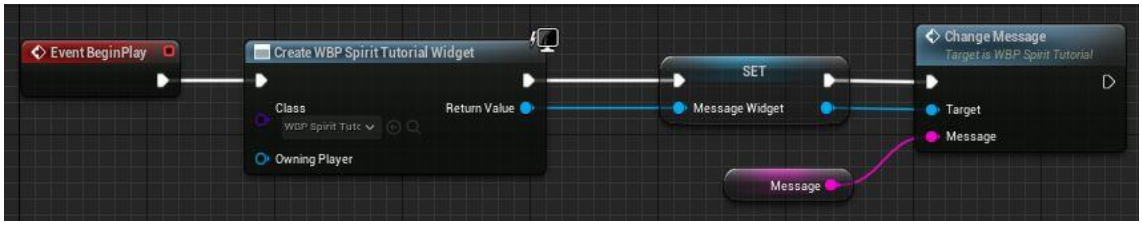

Figura 220. BeginPlay de BP_Torii

Col·lisions

Com es pot veure a la Figura 221 i a la Figura 222, quan el personatge es solapa amb el *collider TriggerMessage*, s'executa l'esdeveniment *OnComponentBeginOverlap*. Primer de tot, es comprova si ja té l'arma. Si no la té, s'afegeix el *widget* al *viewport*, en canvi, si la té, es destrueix el *collider* que bloqueja el pas i el que crida a l'esdeveniment actual.

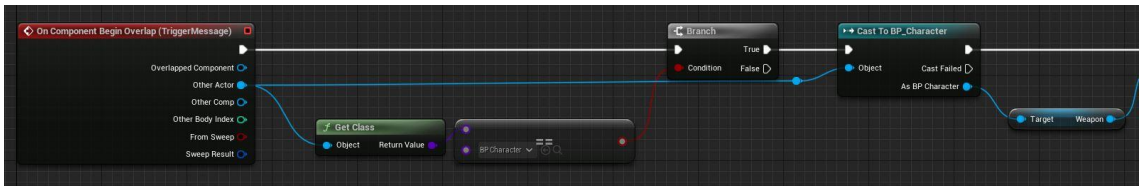

Figura 221. OnComponentBeginOverlap de BP_Torii (primera part)

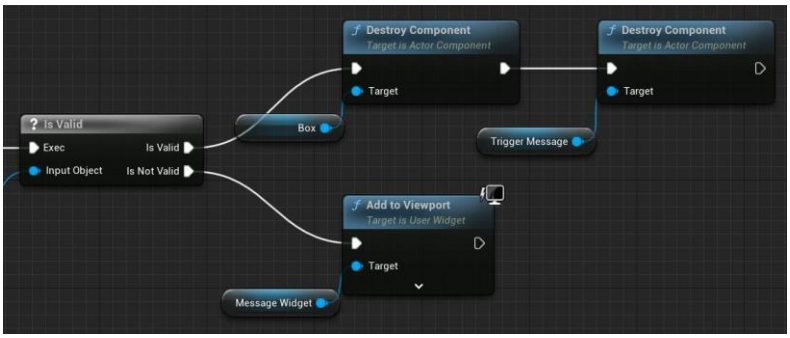

Figura 222. OnComponentBeginOverlap de BP_Torii (segona part)

Pel que fa a l'esdeveniment *OnComponentEndOverlap* (Figura 223), quan el personatge encara no disposa de l'arma i surt de l'àrea del *TriggerMessage*, el missatge s'esborra.

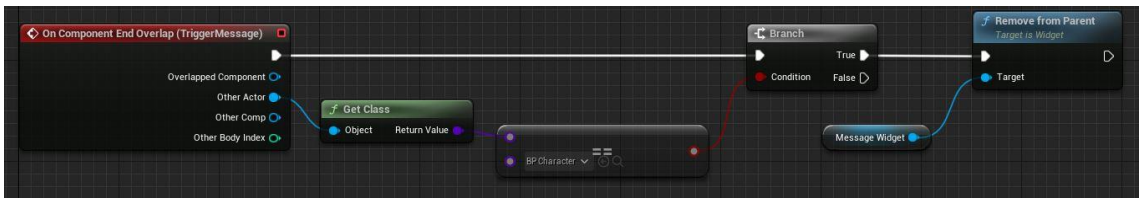

Figura 223. OnComponentEndOverlap de BP_Torii

6.5.4. Pedra màgica

La pedra màgica està implementada a *BP_MagicStone*. Com aquest és un dels objectes més importants del joc, s'ha volgut destacar visualment, així que, com es veu a la Figura 224, se li han afegit unes partícules (*NS_SparklesBlue*) al voltant i una esfera (*Shiny*) que representa una aura lluminosa al seu voltant.

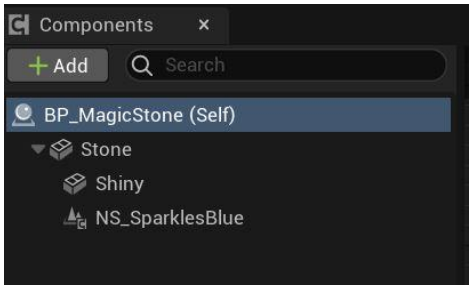

Figura 224. Components de BP_MagicStone

Per altra part, a la Figura 225 es mostra la visió global d'aquest *blueprint*.

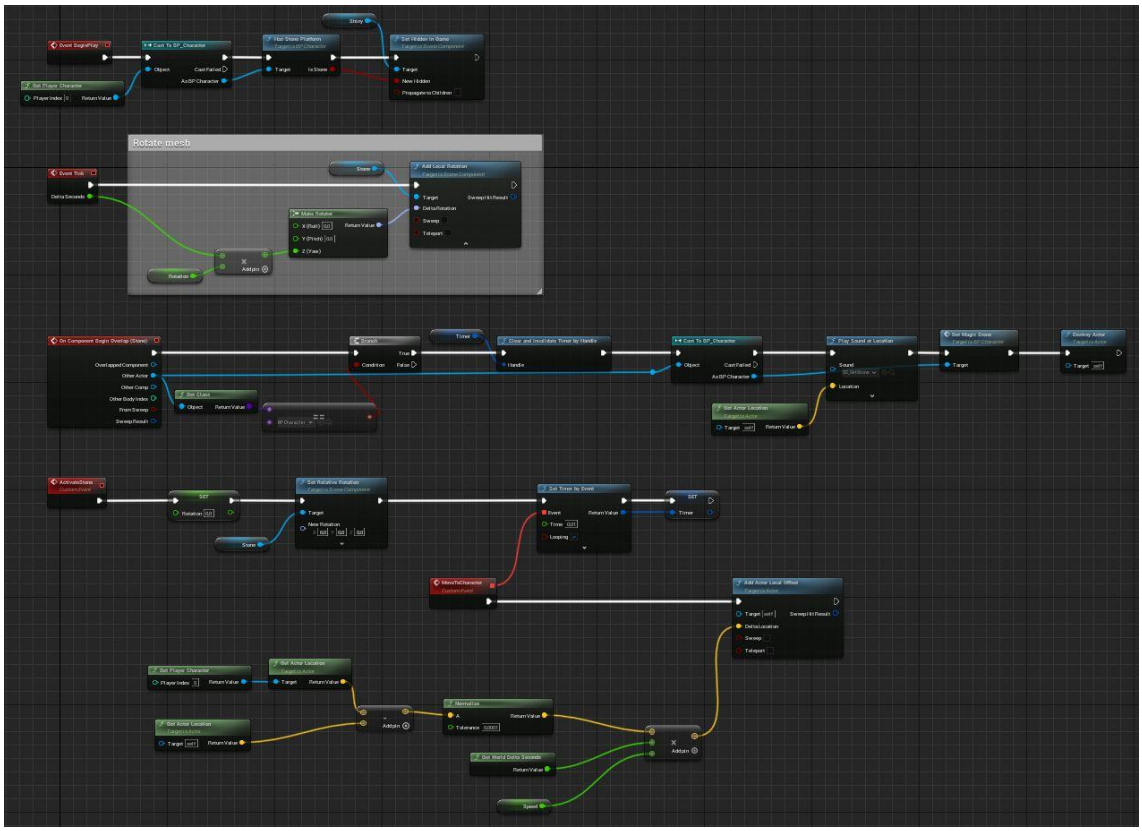

Figura 225. Visió global de BP_MagicStone

Inicialització

A l'esdeveniment *BeginPlay* (Figura 226) es comprova si el jugador ja té la pedra, si és així, no és necessari mostrar l'esfera brillant que està al seu voltant, aquesta només s'havia pensat que es mostrés abans d'obtenir-la.

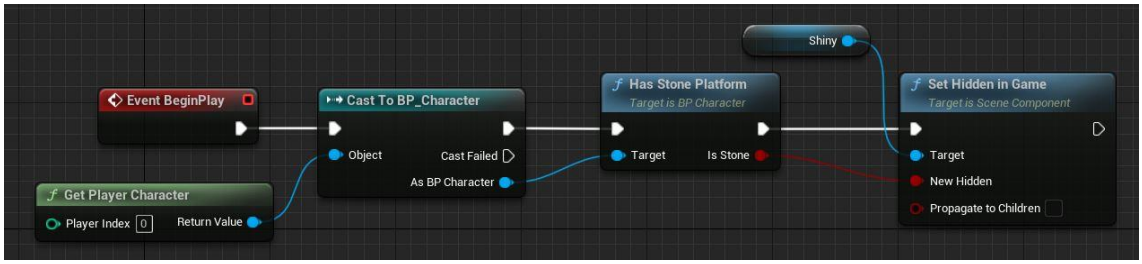

Figura 226. BeginPlay de BP_MagicStone

Moviment

L'esdeveniment *Tick* s'executa a cada *frame*, i del que s'encarrega en aquesta ocasió és de rotar l'objecte, tal i com es pot veure a la Figura 227. No s'ha cregut necessari crear una animació o *timeline* per a aquest moviment tant senzill.

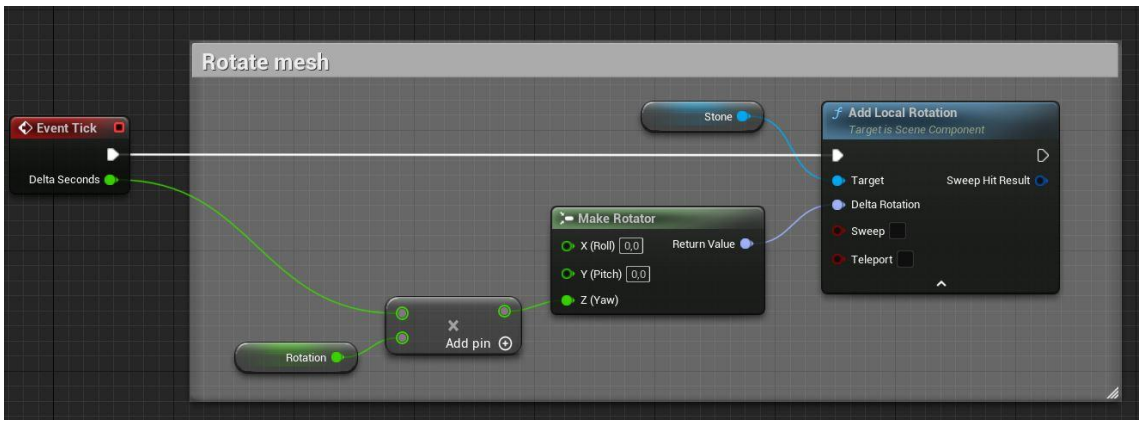

Figura 227. EventTick de BP_MagicStone

Activate Stone

En la Figura 228 es mostra l'esdeveniment *ActivateStone*, el qual s'inicia una vegada s'ha eliminat el primer grup d'enemics. Aquest mètode atura la rotació de l'actor i, a continuació, executa cada 0.01 segons, l'esdeveniment *MoveToCharacter*. Aquest últim s'encarrega de moure l'objecte des de la posició inicial fins a la del personatge.

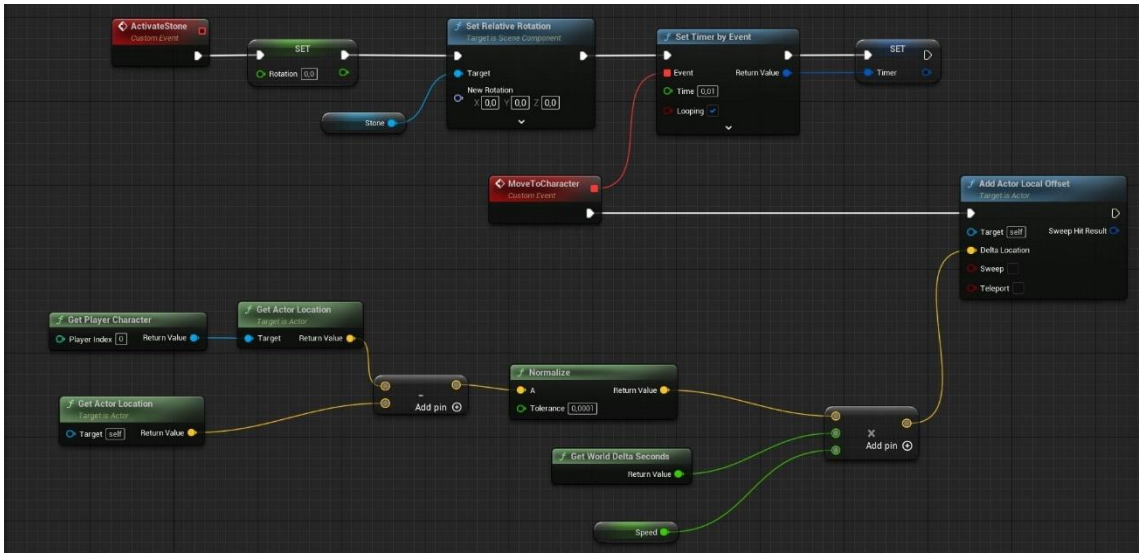

Figura 228. Esdeveniment ActivateStone

Col·lisions

Tal i com es mostra a la Figura 229 i a la Figura 230, quan la pedra col·lisiona amb el jugador, el *timer* creat en l'esdeveniment anterior s'elimina. Seguidament, es reprodueix un so; es crida el mètode *GetMagicStone* (explicat a l'Apartat 6.3.10), el qual modifica una variable per a indicar que el jugador posseeix l'objecte, i, finalment, es destrueix.

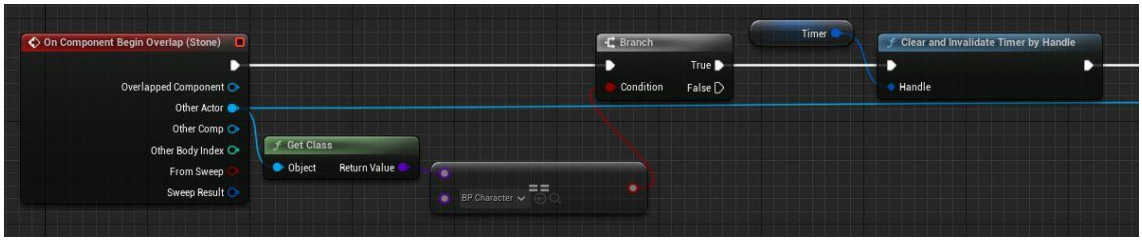

Figura 229. OnComponentBeginOverlap (primera part)

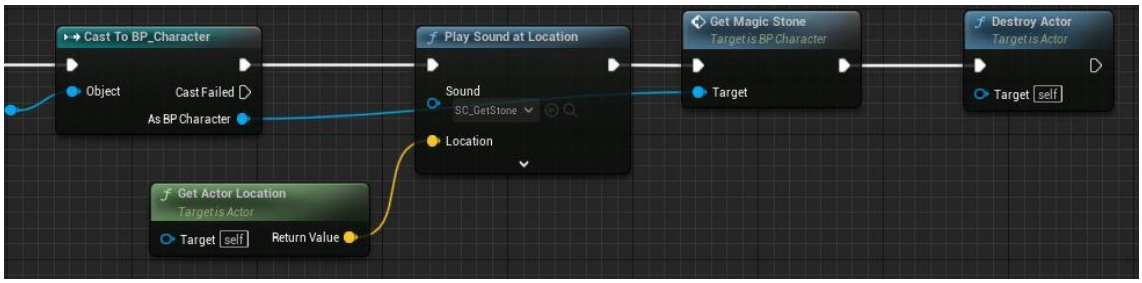

Figura 230. OnComponentBeginOverlap (segona part)

6.5.5. Estàtua curativa

L'estàtua curativa que permet al jugador recuperar la vida per complet, s'ha implementat a *BP_HealerFox*. En la Figura 231 es mostren tots els esdeveniments del *blueprint* en qüestió.

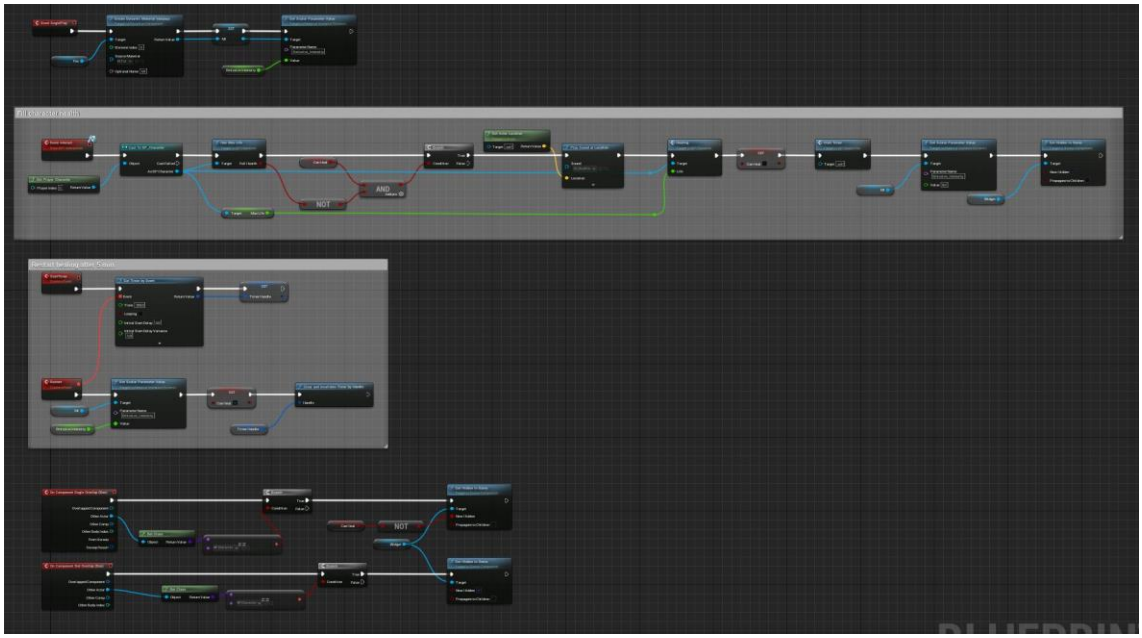

Figura 231. Visió global del BP_HealerFox

Inicialització

L'esdeveniment *BeginPlay* (Figura 232) s'encarrega de crear una instància d'un material i canviar-li el valor a un dels seus paràmetres, en aquest cas, l'emissiu, que aporta lluminositat a les esquerdes de l'objecte. Que aquest valor sigui major a 0, indica que la curació de l'estàtua està disponible.

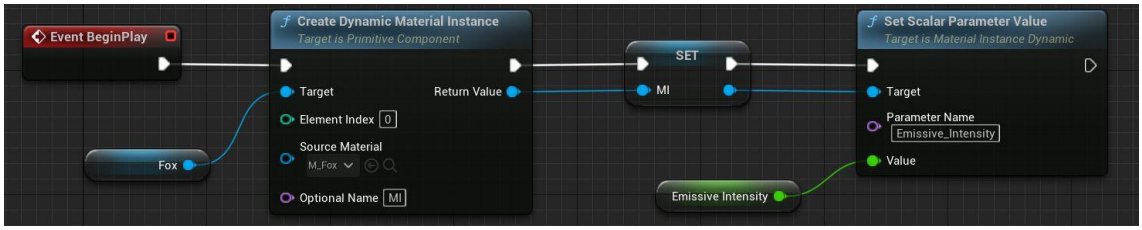

Figura 232. BeginPlay de BP_HealerFox

Col·lisions

Com ja s'ha vist en altres objectes i es mostra en la Figura 233, si el jugador s'apropa al *collider* d'aquesta guineu i és possible recuperar vides, se li indica que, prement un botó o tecla en concret, pot interaccionar amb l'actor. En canvi, quan surt del seu rang, el missatge desapareix.

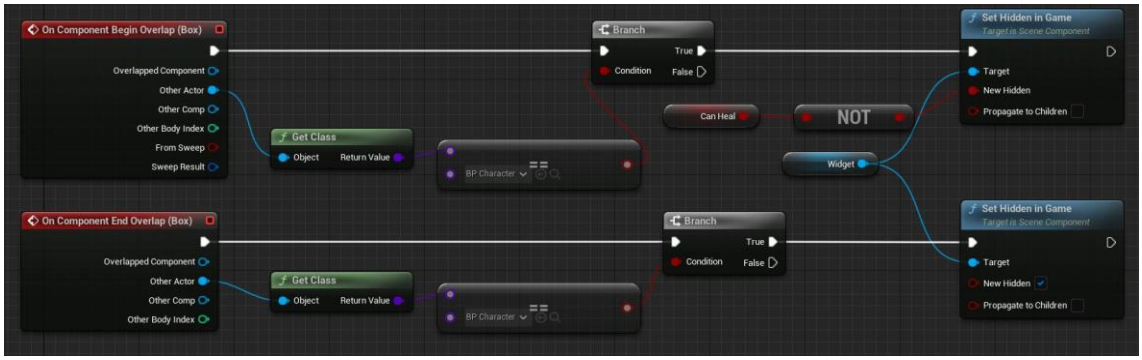

Figura 233. Overlaps de BP_HealerFox

Interacció

En aquest *blueprint* també s'utilitza l'esdeveniment *Interact del BPI_Interaction* (Figura 234, Figura 235 i Figura 236). En aquesta ocasió, quan el jugador interacciona amb l'objecte, primer de tot, es comprova que la vida no estigui al màxim i que la curació estigui disponible. Si totes dues condicions es compleixen, es reprodueix un so de curació; es crida a l'esdeveniment *Healing* (Apartat 6.3.11) indicant que es vol curar el màxim de vida passant-li la variable *MaxLife*; com ja s'ha curat, es canvia la variable *canHeal* a *false*; es torna a canviar el valor del paràmetre emissiu a 0; s'amaga el *widget*, i es crida al mètode *StartTimer* del propi *blueprint* que torna a habilitar la curació al cap d'un temps.

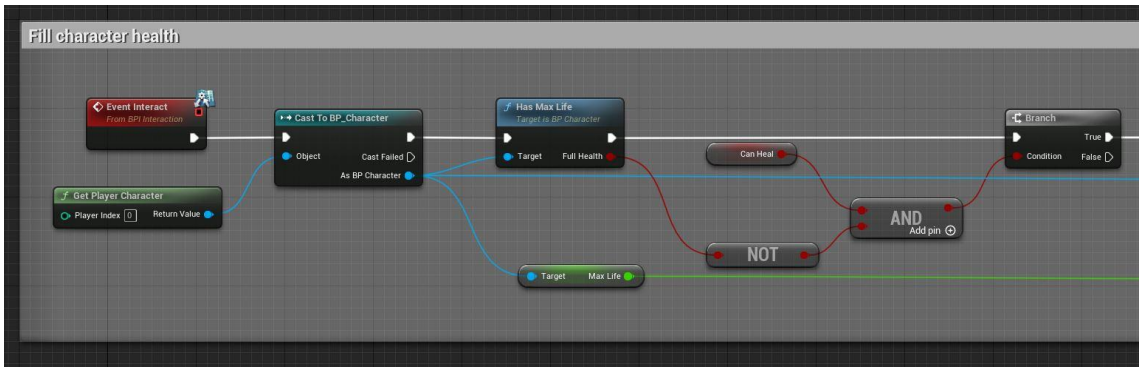

Figura 234. Interact de BP_HealerFox (primera part)

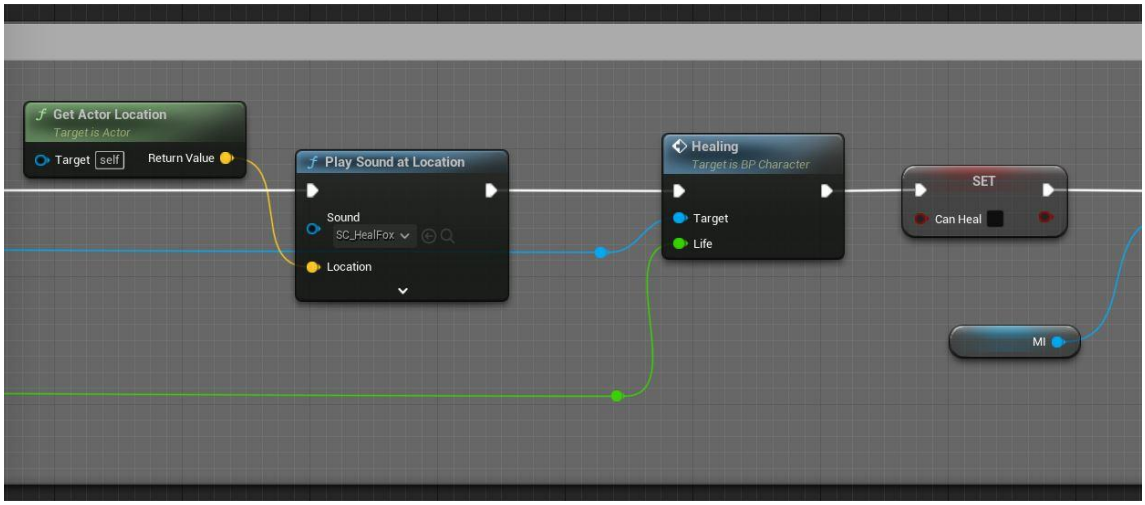

Figura 235. Interact de BP_HealerFox (segona part)

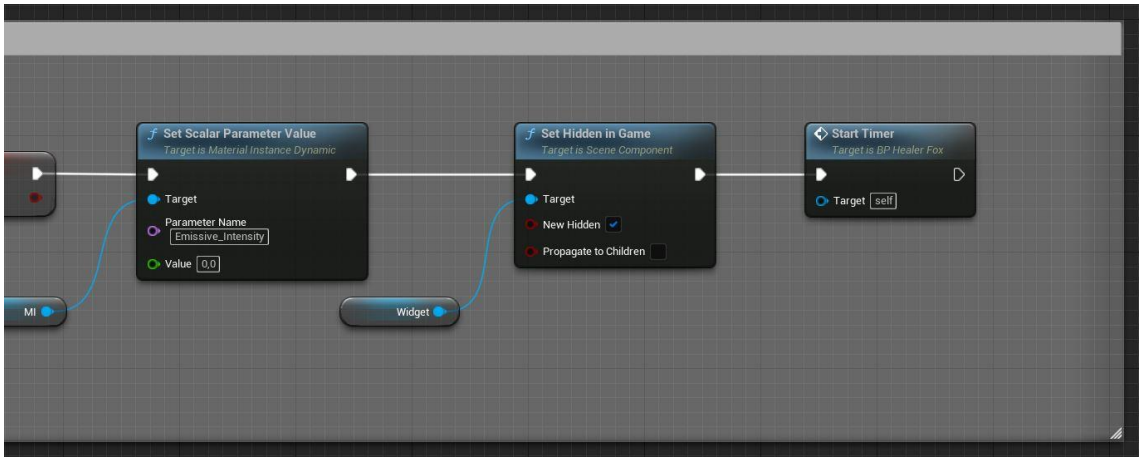

Figura 236. Interact de BP_HealerFox (tercera part)

Reinici

Com es pot veure en la Figura 237 i com s'acaba de comentar, l'esdeveniment *StartTimer* inicia un *timer* que executa el mètode *Restart* al cap de 5 minuts. Aquest mètode torna a posar el material emissiu a un valor superior a 0 i la variable *CanHeal* a *true*, a més, s'elimina el *timer*. Després d'haver reiniciat aquestes variables, el jugador pot tornar a curar-se.

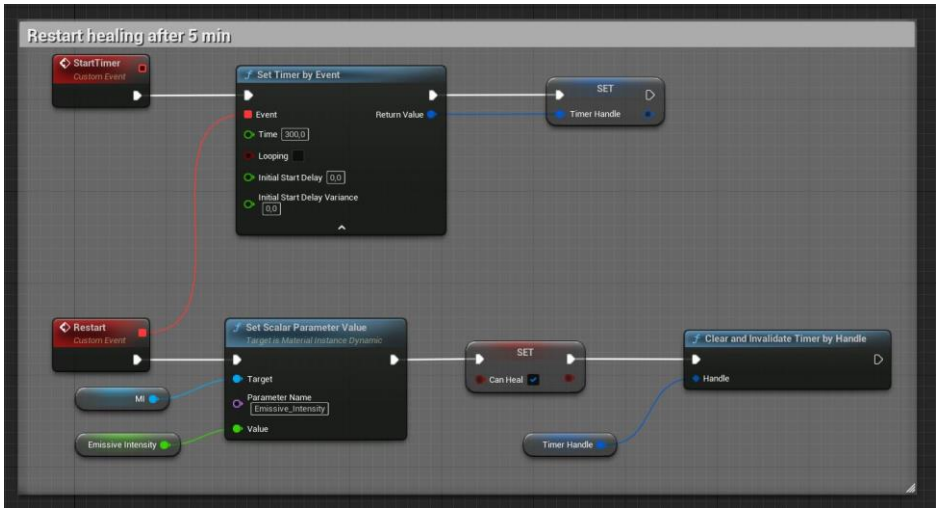

Figura 237. Restart de BP_HealerFox

6.5.6. Pètal

El pètal ha estat implementat al *blueprint BP_Petal* (Figura 238). Aquest actor apareix quan s'elimina el segon grup d'enemics i cura una vida al jugador, a més, igual que amb la pedra màgica, se li han afegit partícules per embellir l'objecte.

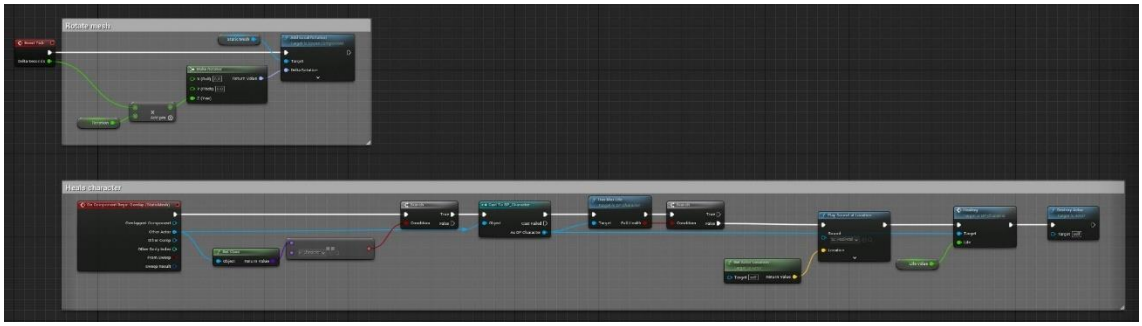

Figura 238. Visió global de BP_Petal

Moviment

L'*Event Tick* (Figura 239) aplica, a cada *frame*, una petita rotació a l'eix Z, d'aquesta manera el pètal es veu en moviment, a l'espera de que el jugador l'agafi.

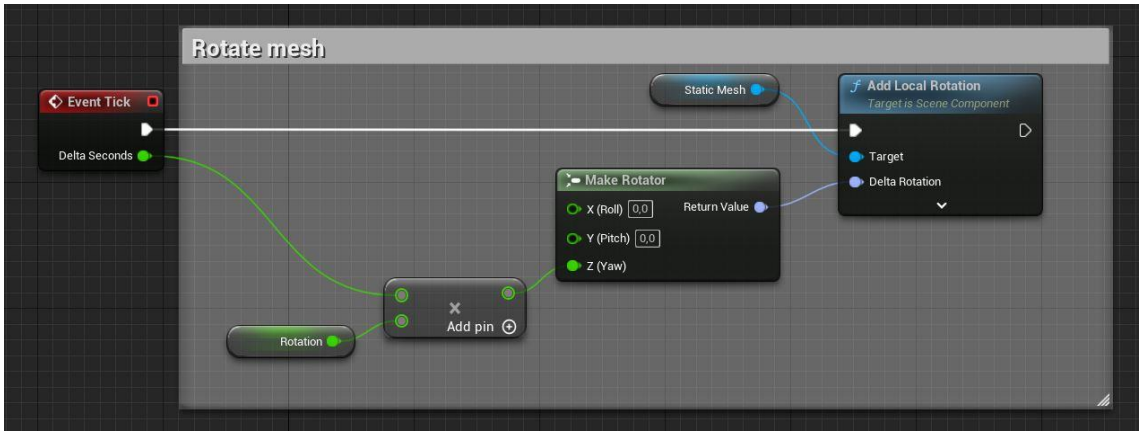

Figura 239. Event Tick de BP_Petal

Col·lisions

Quan el jugador passa pel damunt de l'actor, es crida a l'esdeveniment *OnComponentBeginOverlap* (Figura 240 i Figura 241). Aquest mètode comprova si el personatge té tota la vida o no. En cas que no, es reprodueix un so; es crida a l'esdeveniment *Healing* (Apartat 6.3.11) passant-li la variable *LifeValue*, que en aquesta ocasió, el valor és 1, i es destrueix.

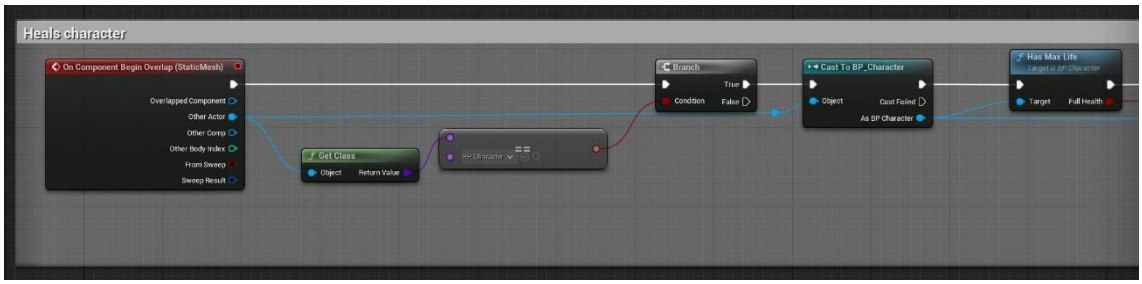

Figura 240. BeginOverlap de BP_Petal (primera part)

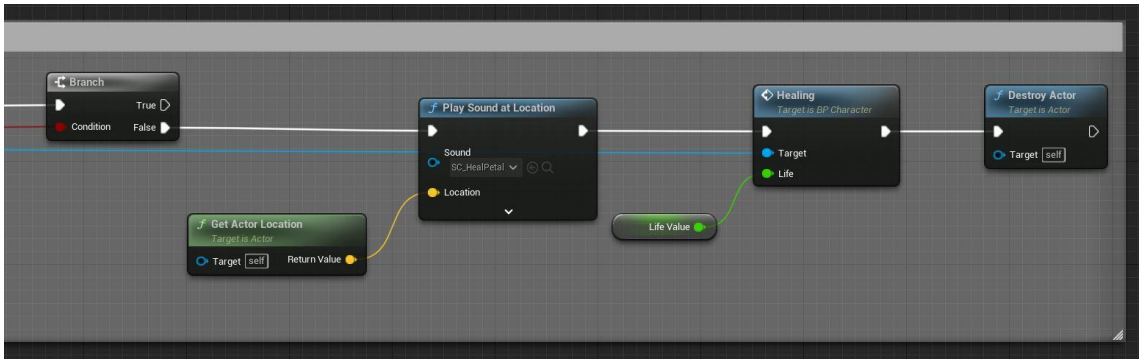

Figura 241. BeginOverlap de BP_Petal (segona part)

6.5.7. Puzle

La següent figura mostra els esdeveniments implementats al *blueprint* del puzle anomenat *BP_PuzzleSeq* (Figura 242). Aquest puzle es troba a la zona anterior a la del final del joc. Resoldre'l és obligatori pel jugador, ja que si no ho fa, el portal que permet l'accés a la zona final es manté bloquejat.

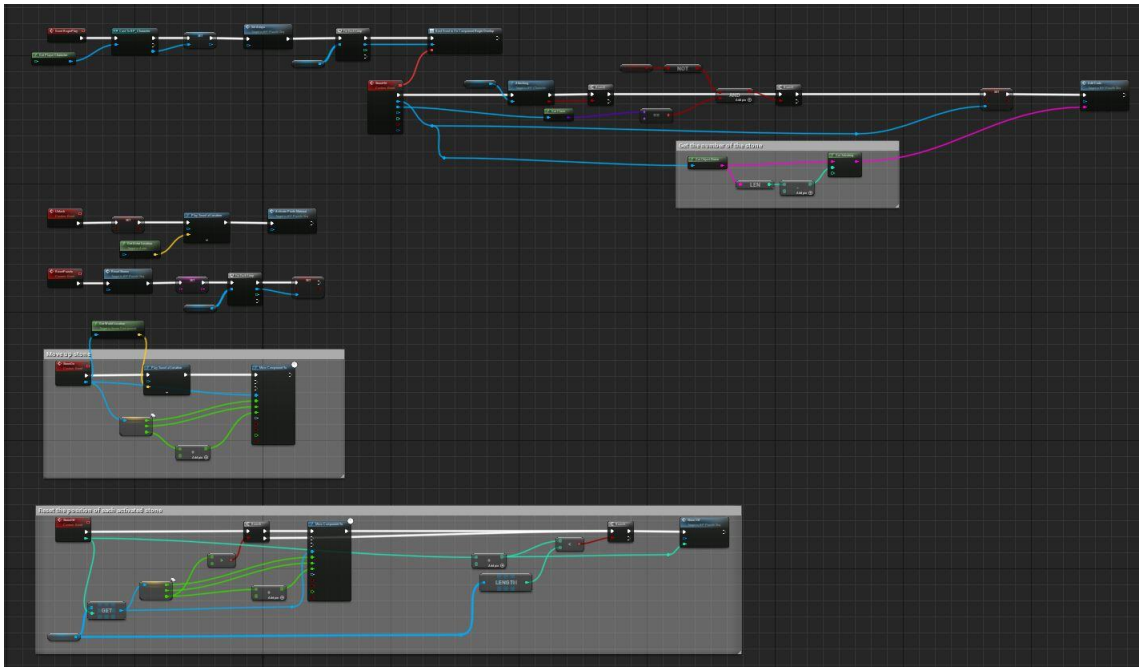

Figura 242. Visió global del graf de BP_PuzzleSeq

El puzle consta de sis pedestals que s'han d'activar en l'ordre correcte; indicat per la quantitat de flors (afegides a l'escenari) que té cadascun. Cada pedestal inclou un *collider* i una pedra, tal i com es pot veure en la Figura 243. A més, per a poder gestionar els diferents elements, saber quins estan activats correctament i l'ordre correcte, s'han definit les variables que es mostren en la Figura 244.

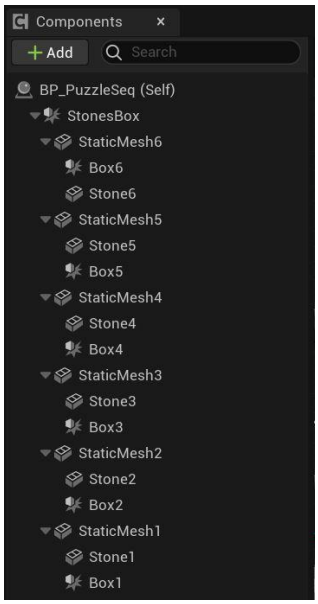

Figura 243. Component de BP_PuzzleSeq

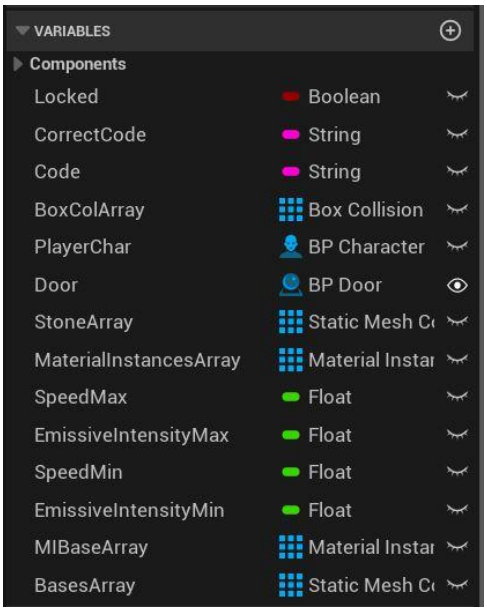

Figura 244. Variables de BP_PuzzleSeq

Inicialització

L'esdeveniment *BeginPlay* (Figura 245, Figura 246, Figura 247 i Figura 248) s'encarrega, primer de tot, d'inicialitzar els diferents *arrays* amb la funció *SetArrays* que s'explica amb més detall en el següent apartat.

A continuació, per cada *collider* s'espera a que un objecte hi col·lisioni, executant així, l'esdeveniment *StoneHit*. S'ha decidit utilitzar el *Bind Event to OnComponentBeginOverlap*, ja que, com tots els *colliders* han de tenir el mateix comportament, s'ha cregut que és molt redundant crear un esdeveniment per a cadascun que esperar a que s'executi el *bind*, tot i que el resultat és el mateix, ja que s'ha hagut de fer un bucle per a que tots els elements estiguin associats a aquest esdeveniment.

Així que, un cop iniciat l'*StoneHit*, es comprova si el personatge està atacant, a través de la crida de la funció *Attacking* (Apartat 6.3.16); si l'objecte que col·lisiona és l'arma, i si el puzle està desbloquejat. Si totes les condicions es compleixen, es deshabilita la col·lisió per a la pedra en qüestió i, seguidament, es crida a la funció *AddCode*, a la qual se li passa el número al que correspon la pedra (1-6), extraient-lo del nom.

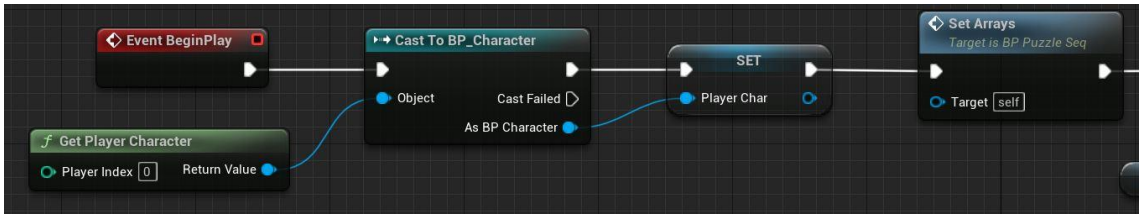

Figura 245. BeginPlay de BP_PuzzleSeq (primera part)

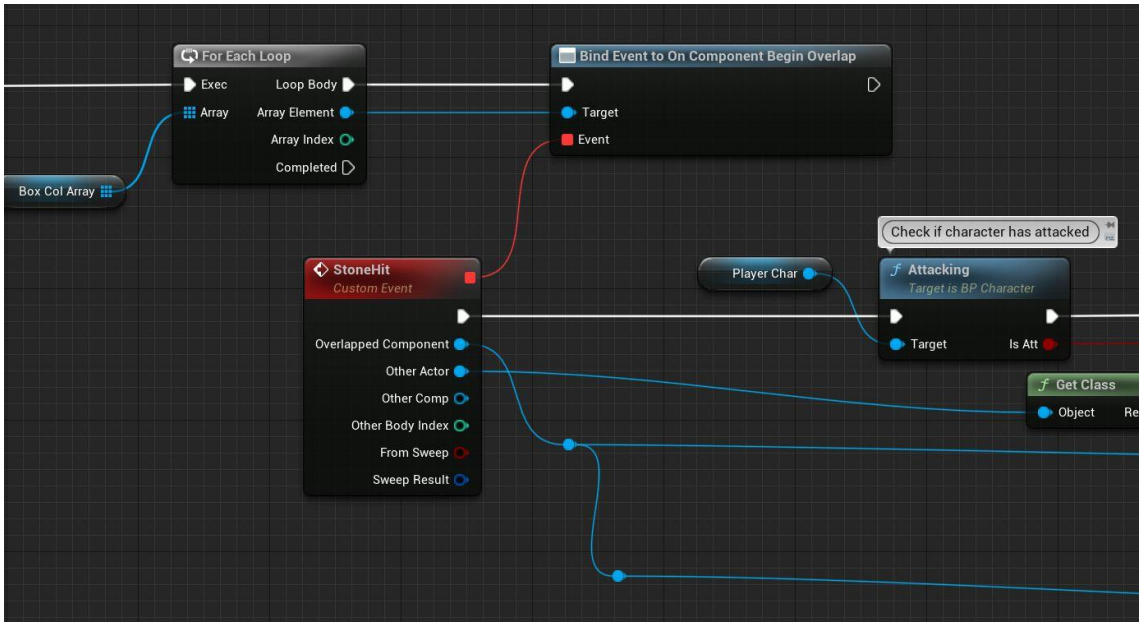

Figura 246. BeginPlay de BP_PuzzleSeq (segona part)

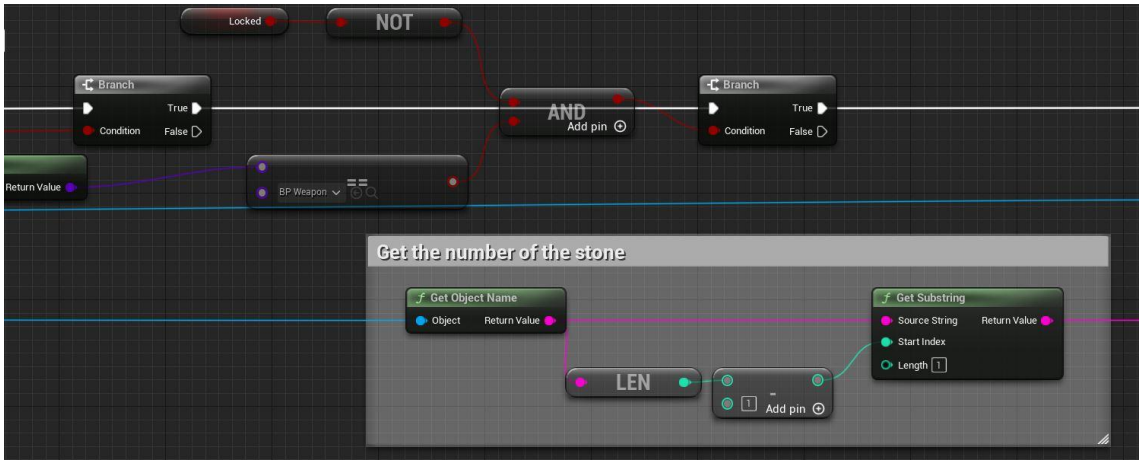

Figura 247. BeginPlay de BP_PuzzleSeq (tercera part)

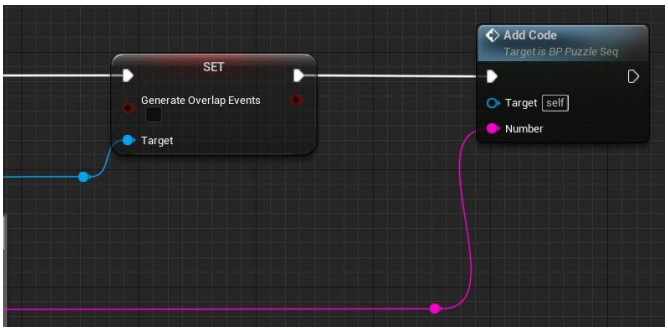

Figura 248. BeginPlay de BP_PuzzleSeq (quarta part)

SetArrays

La funció *SetArrays* (Figura 249, Figura 250 i Figura 251), inicialitza els *arrays* dels *colliders*, de les pedres i els pedestals amb els components existents. A més, també s'han creat altres dos *arrays* (per les pedres i els pedestals), als quals se'ls guarden instàncies d'un material (Apartat 6.8). D'aquesta manera, posteriorment, es poden modificar paràmetres d'aquests materials per a que canviïn en temps d'execució.

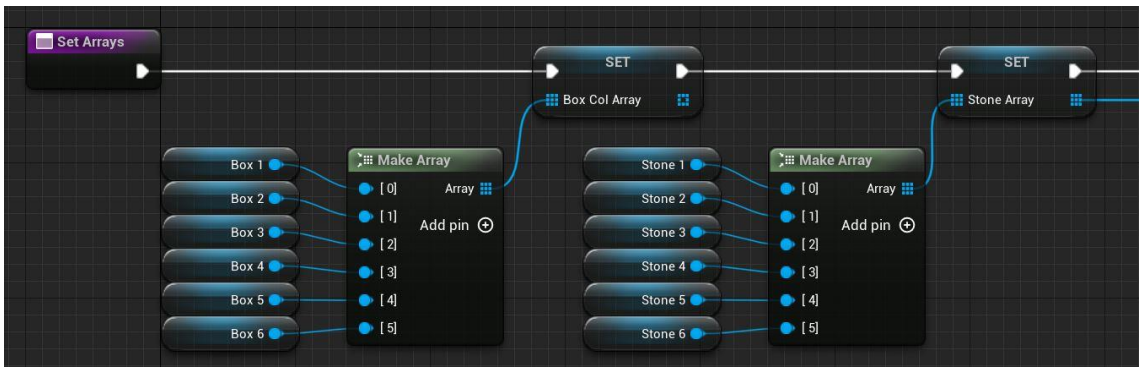

Figura 249. SetArrays (primera part)

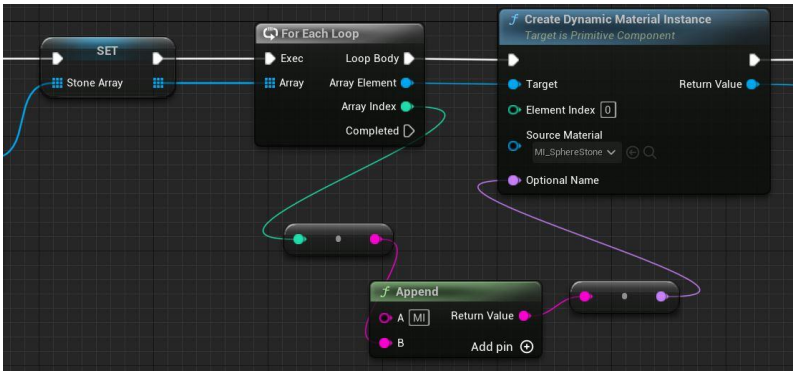

Figura 250. SetArrays (segona part)

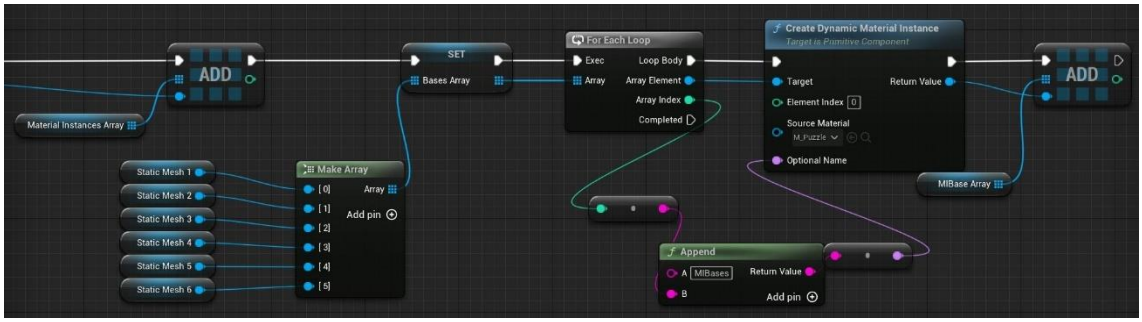

Figura 251. SetArrays (tercera part)

Unlock

L'esdeveniment *Unlock*, tal i com es veu a la Figura 252, canvia el valor de la variable *Locked*, que està a *true* per defecte; reprodueix un so, i crida a la funció *ActivatePuzleMaterial*. Aquest mètode es crida un cop s'ha eliminat l'últim grup d'enemics.

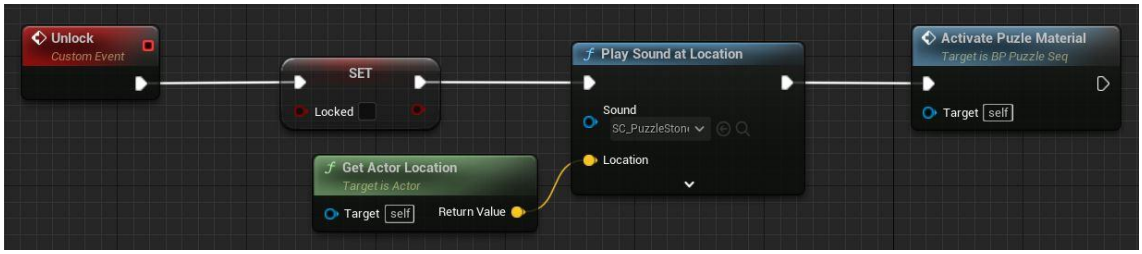

Figura 252. Unlock

ActivatePuzleMaterial

La funció *ActivatePuzleMaterial* (Figura 253) modifica el paràmetre *Emissive_Intensity* del material, fent això s'indica, visualment, que el puzle s'ha desbloquejat i ja es pot resoldre.

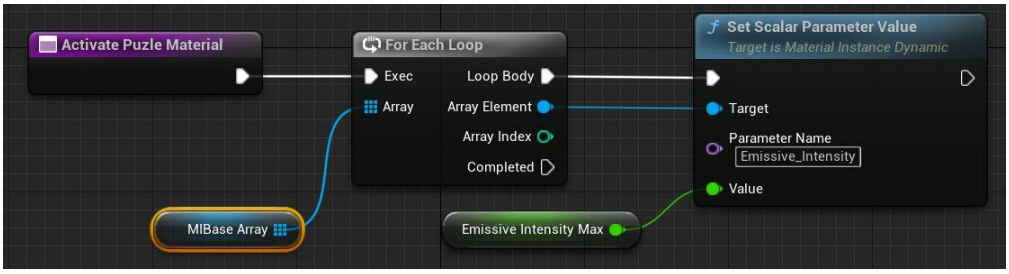

Figura 253. ActivatePuzleMaterial

AddCode

La funció *AddCode* (Figura 254 i Figura 255) s'utilitza per anar comprovant si l'ordre que segueix el jugador quan interactua amb les pedres és correcte o no. Primer de tot, s'afegeix el número que s'ha passat pel paràmetre *input* a la variable *Code*, que va guardant l'ordre per a poder-lo comparar amb el correcte, el *CorrectCode*.

Si la longitud de *Code* és la mateixa que la del codi correcte, vol dir que ja s'han activat totes les pedres, tot i així es comparen els dos valors, i si són correctes, es torna a bloquejar el puzle, s'activa l'última pedra amb la funció *ActivateStone* i es crida a la funció *Open* (6.5.8) del portal, per a que aquest s'obri. En canvi, si la longitud del codi no és la mateixa, significa que el jugador encara no ha acabat el puzle, així que es va comprovant si els codis coincideixen en cada moment, si és així, s'activa la pedra i si no, es crida a la funció *ResetPuzzle*, per a que el jugador torni a començar des de zero.

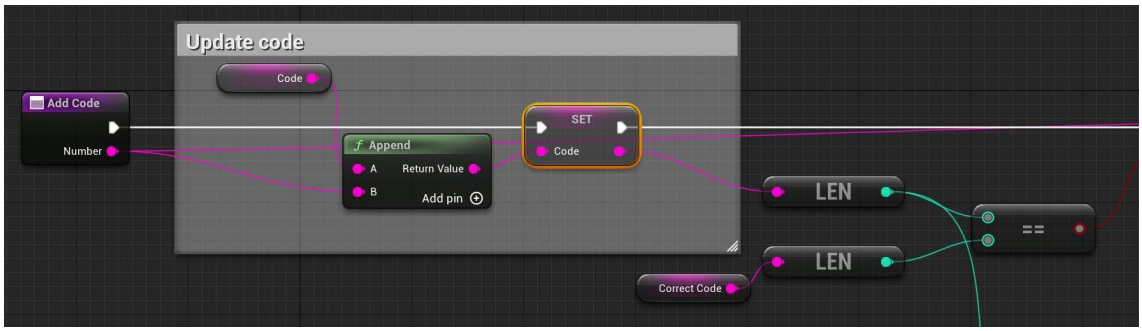

Figura 254. AddCode (primera part)

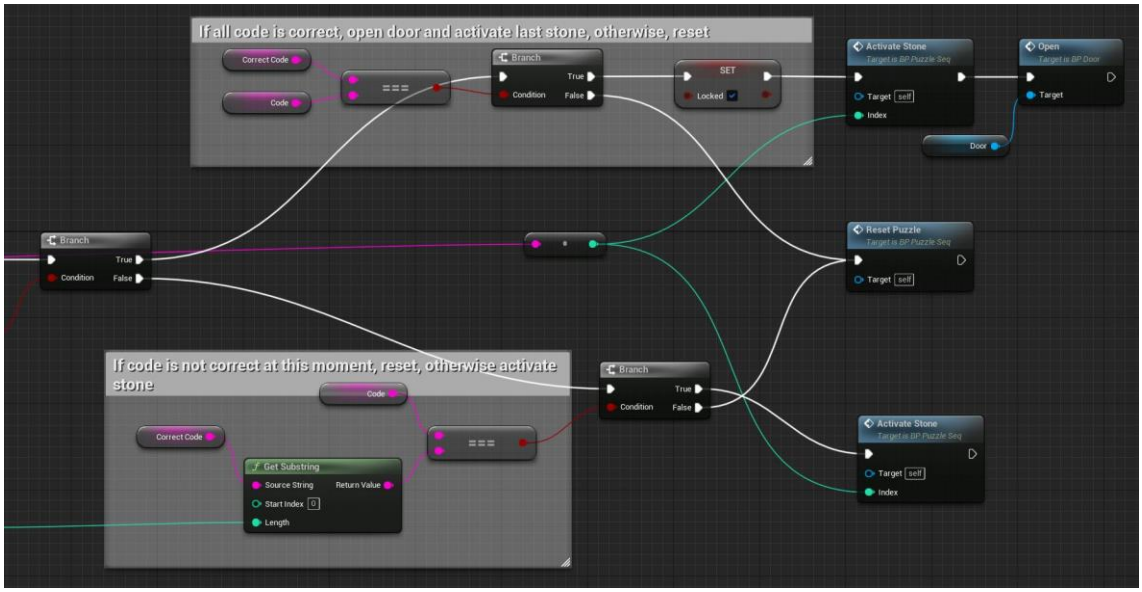

Figura 255. AddCode (segona part)

ActivateStone

ActivateStone (Figura 256 i Figura 257) s'encarrega de canviar els paràmetres *Speed* i *Emissive_Intensity* del material; la qual cosa aporta moviment a la textura i més brillantor per a indicar que l'ordre en el que s'ha activat la pedra és el correcte. A més crida a l'esdeveniment *StoneOn*, que s'explica més endavant.

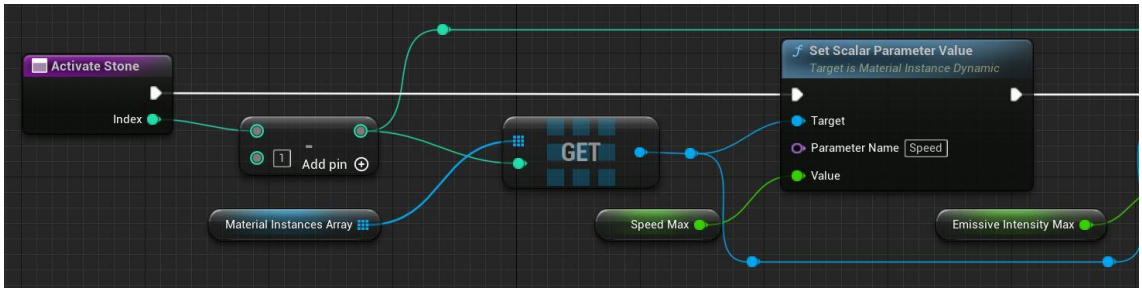

Figura 256. ActivateStone (primera part)

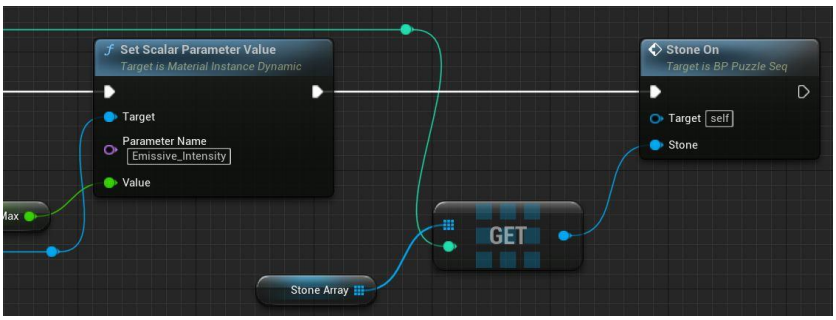

Figura 257. ActivateStone (segona part)

StoneOn

L'esdeveniment *StoneOn* (Figura 258) reprodueix un so que indica que la pedra és correcta i la mou fins a una certa altura amb el node *MoveComponentTo*.

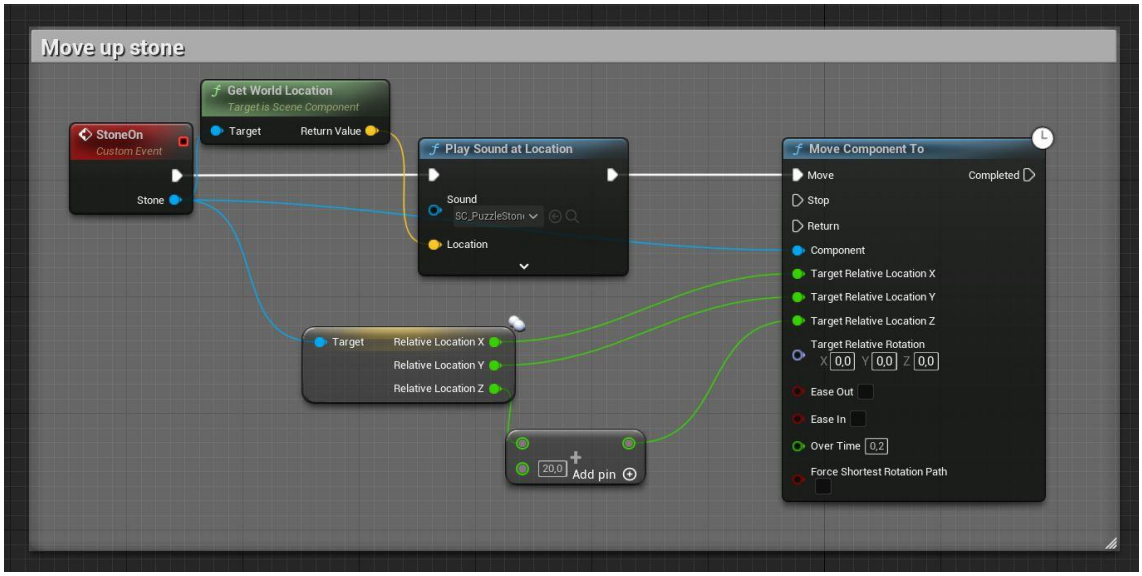

Figura 258. StoneOn

ResetPuzzle

L'esdeveniment *ResetPuzzle,* que es mostra a la Figura 259, s'encarrega de reiniciar tots els pedestals. Primer de tot, crida a la funció *ResetStones*; buida la variable *Code*, ja que aquesta s'ha de tornar a omplir, i per cada *collider,* es tornen a activar les col·lisions.

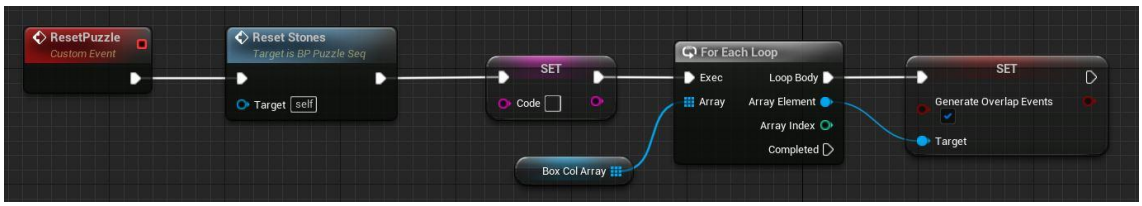

Figura 259. ResetPuzzle

ResetStones

La funció *ResetStones* (Figura 260) és la responsable de canviar totes les característiques visuals per a indicar que la pedra ja no està activa. Abans de fer-ho, reprodueix un so de fallida. Seguidament, per cada pedra, es canvien els paràmetres *Speed* i *Emissive_Intensity* de les instàncies del material. A més, un cop s'ha acabat de fer, també es crida al mètode *StoneOff* que es troba al graf principal.

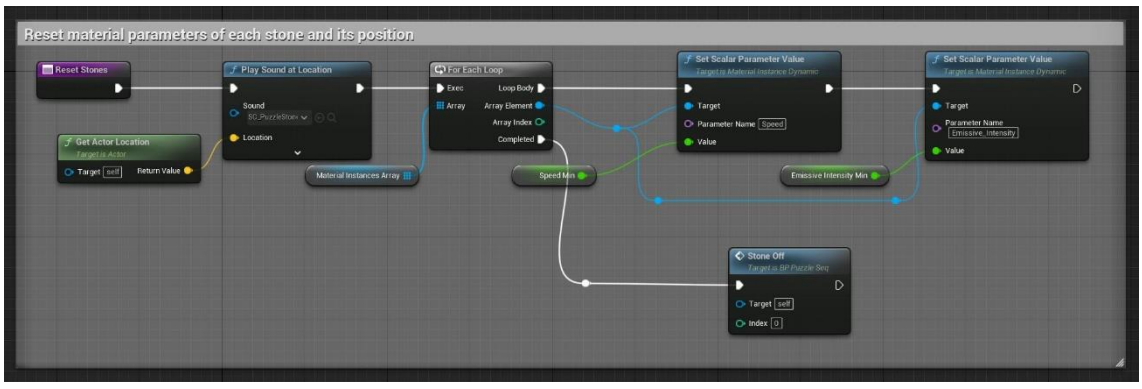

Figura 260. ResetStones

StoneOff

L'esdeveniment *StoneOff* (Figura 261) és l'encarregat de tornar a deixar en la posició inicial a totes les pedres activades. Per començar, es comprova que l'altura sigui major a 0, és a dir, que estigui activa; si ho està, es mou amb el node *MoveComponentTo* fins la posició original. Si encara queden elements per comprovar, es torna a crida a l'*StoneOff*. Inicialment, s'havia provat de fer això mateix amb un *loop*, però no funcionava correctament amb el node que mou l'objecte. Així que s'ha decidit fer el bucle a mà, i no passar al següent element fins que no s'ha completat el moviment.

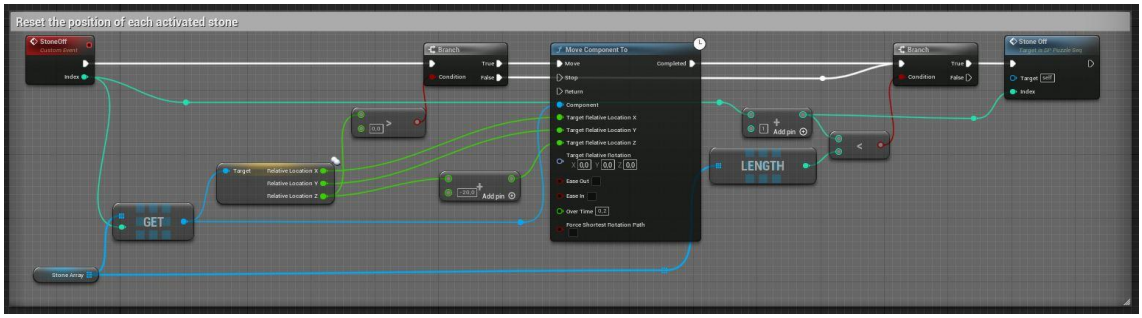

Figura 261. StoneOff

6.5.8. Portal

El portal que dona pas a l'última zona del mapa està implementat a *BP_Door*. En la Figura 262 es poden veure els esdeveniments que el composen.

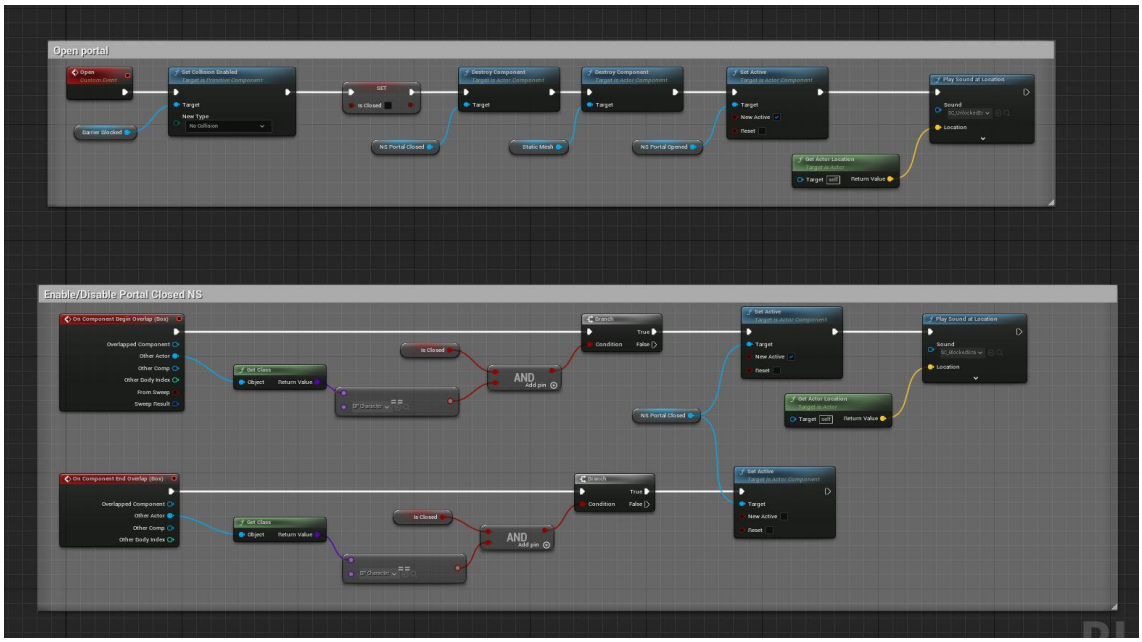

Figura 262. Visió global de BP_Door

Col·lisions

Els mètodes *OnComponentBeginOverlap* i *OnComponentEndOverlap* que es mostren en la Figura 263, s'executen quan el jugador entra o surt de l'àrea que ocupa el *collider Box*. Si el jugador intenta passar a través del portal mentre la variable *IsClosed* està a true; el

que vol dir que encara està bloquejat; les partícules s'activen i es reprodueix un so, en canvi, si el personatge surt de la zona, les partícules s'oculten.

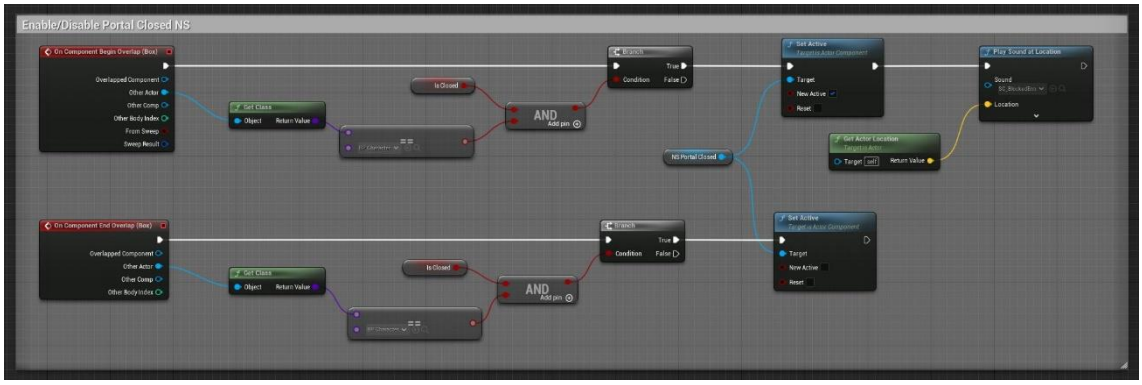

Figura 263. Overlaps de BP_Door

Desbloqueig

L'esdeveniment *Open* (Figura 264 i Figura 265) s'executa quan el jugador ha resolt el puzle de l'apartat anterior. Aquest mètode deshabilita la col·lisió de la barrera; canvia el valor del booleà *IsClosed* a *false*; destrueix el sistema de partícules i la boira que es mostra quan el portal està tancat; activa les noves partícules que indiquen que el portal està obert, i reprodueix un so.

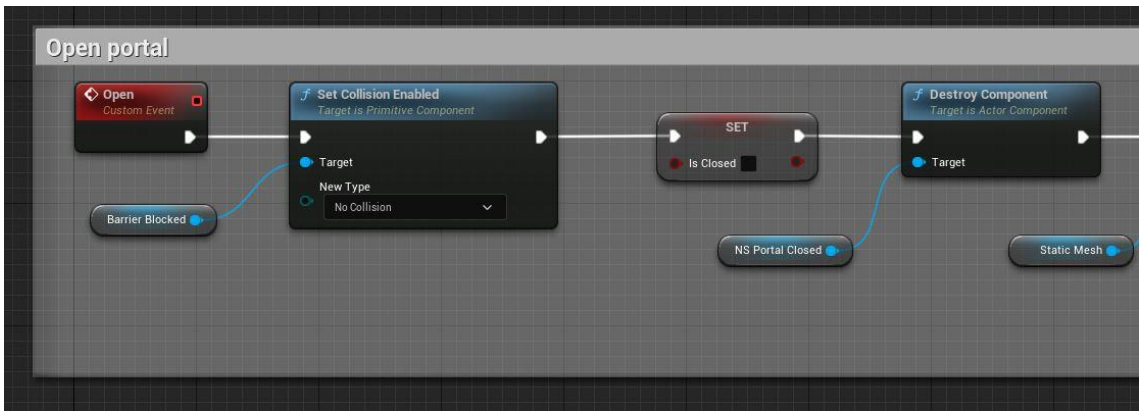

Figura 264. Esdeveniment Open (primera part)

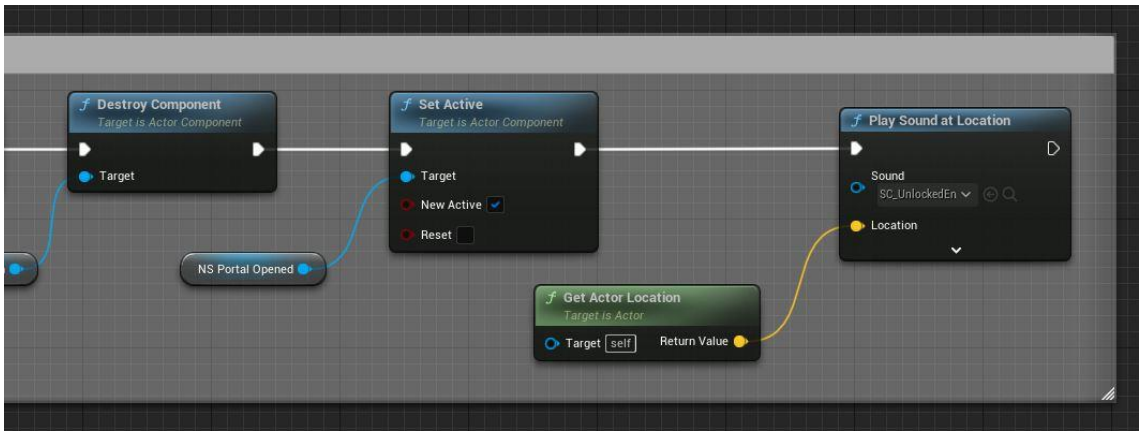

Figura 265. Esdeveniment Open (segona part)

6.5.9. Roca

El següent *blueprint,* anomenat *BP_BrokenRock* (Figura 266)*,* és l'encarregat d'enfonsar algunes roques que formen el camí que porta a la plataforma final, on es troba el Gran Esperit. En aquest camí, el jugador es pot trobar amb roques totalment estàtiques o amb instàncies d'aquest actor. Per diferenciar-les només s'ha de fixar en les esquerdes que aquestes tenen a la textura i les altres no.

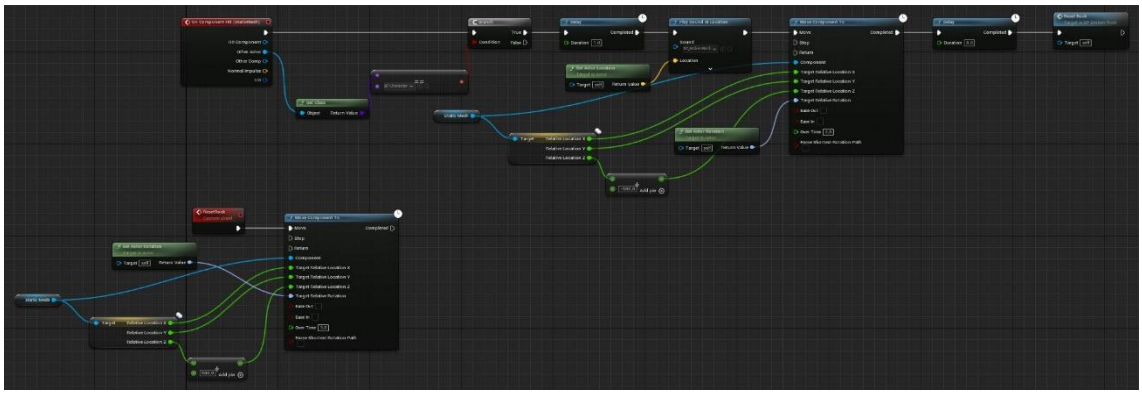

Figura 266. Visió general de BP_BrokenRock

Col·lisions

Tal i com es veu a la Figura 267 i la Figura 268 , quan el personatge toca l'objecte, l'actor cau una certa distància al cap d'1 segon. A més, es reprodueix un so, per a que el jugador se n'adoni què està passant, ja que és possible que essent suficientment ràpid no vegi que les roques s'enfonsen, tot i que, just a l'inici del recorregut se li avisa. Per altra part, una vegada la roca ha arribat al punt més baix, al cap de 10 segons, es crida a l'esdeveniment *ResetRock.*

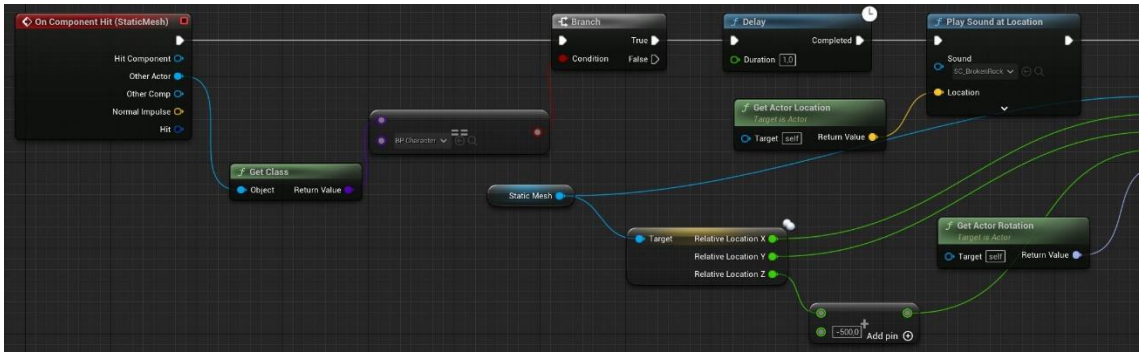

Figura 267. OnComponentHit de BP_BrokenRock (primera part)

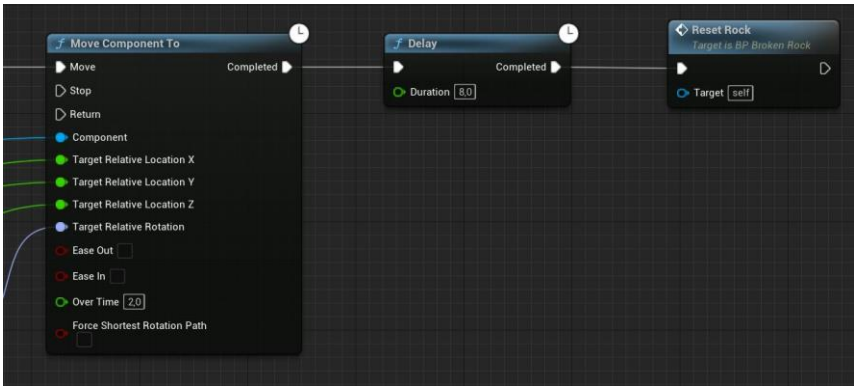

Figura 268. OnComponentHit de BP_BrokenRock (segona part)

ResetRock

El mètode *ResetRock* (Figura 269), mou la roca a la posició inicial, ja que és possible que si el jugador no ha obtingut la pedra màgica, hagi de tornar enrere.

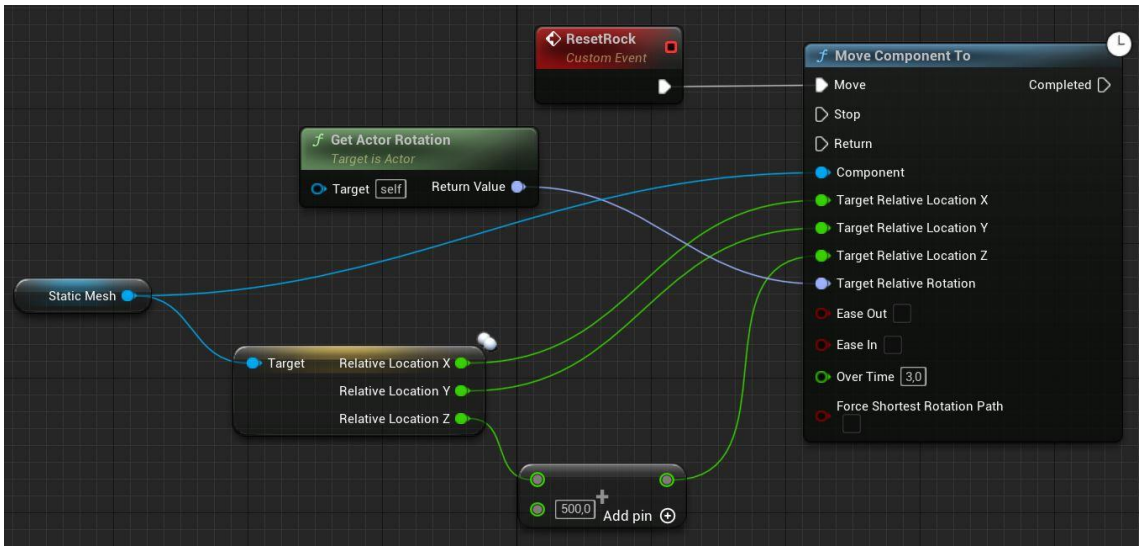

Figura 269. ResetRock

6.5.10. Aigua verinosa

L'aigua verinosa s'ha implementat al *blueprint BP_PoisonousWater* (Figura 270). Aquesta aigua és complementa amb les roques de l'apartat anterior, ja que si una s'enfonsa i el jugador no és capaç de reaccionar, cau a l'aigua i com a conseqüència perd vides.

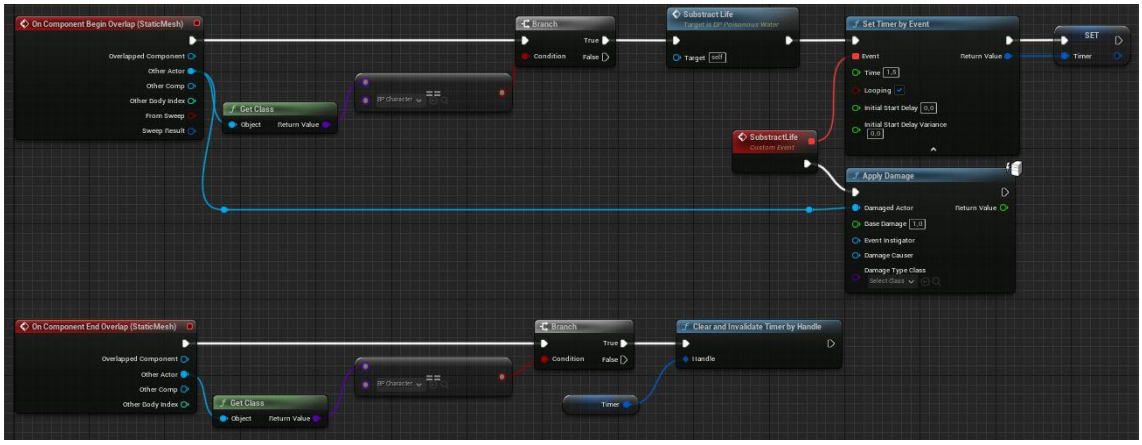

Figura 270. Visió global de BP_PoisonousWater

Col·lisions

L'esdeveniment *OnComponentBeginOverlap* (Figura 271 i Figura 272) s'executa en quant el personatge toca l'aigua de la zona final. El primer que fa es treure una vida utilitzant l'esdeveniment *SubstractLife*, que crida a l'*ApplyDamage*. Aquest mètode continua executant-se cada 1.5 segons fins que el jugador sigui capaç de sortir i pujar a una roca estable, en aquest moment es quan s'inicia l'esdeveniment *OnComponentEndOverlap* (Figura 273), el qual elimina el *timer*.

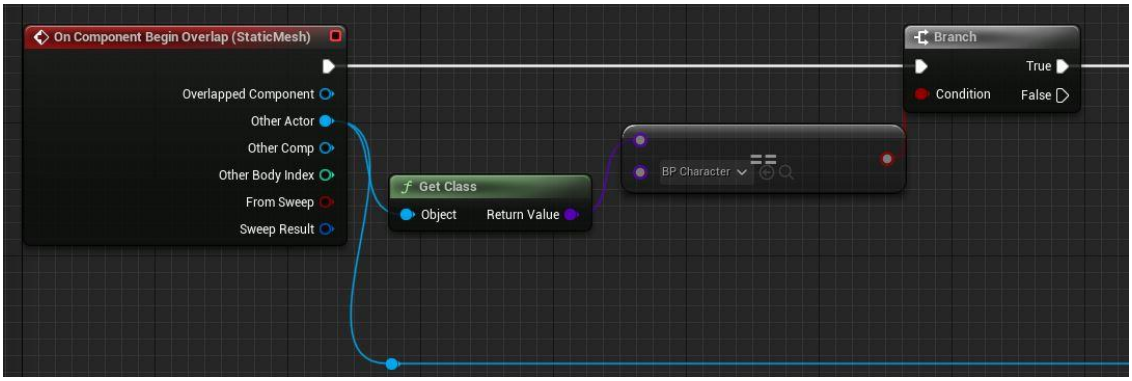

Figura 271. OnComponentBeginOverlap de BP_PoisonousWater (primera part)

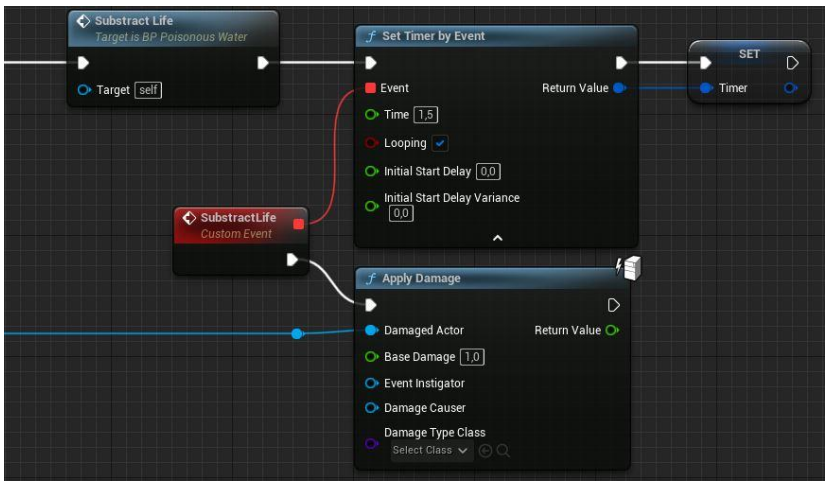

Figura 272. OnComponentBeginOverlap de BP_PoisonousWater (segona part)

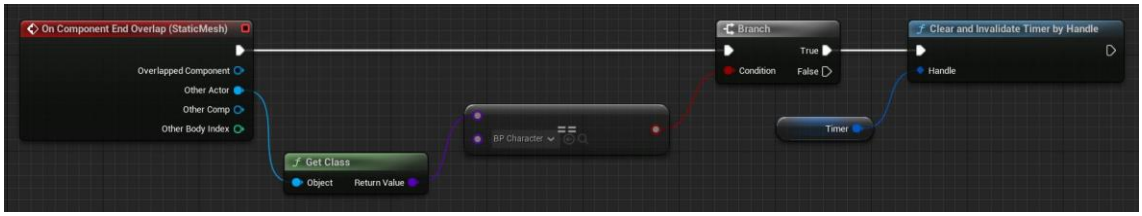

Figura 273. OnComponentEndOverlap de BP_PoisonousWater

6.5.11. Plataforma final

La plataforma final, implementada al *blueprint BP_StonePlatform* (Figura 274), és l'últim actor amb interacció que es troba al mapa, on el jugador ha d'entregar l'objecte clau aconseguit anteriorment. A la següent figura es mostra una visió general del graf:

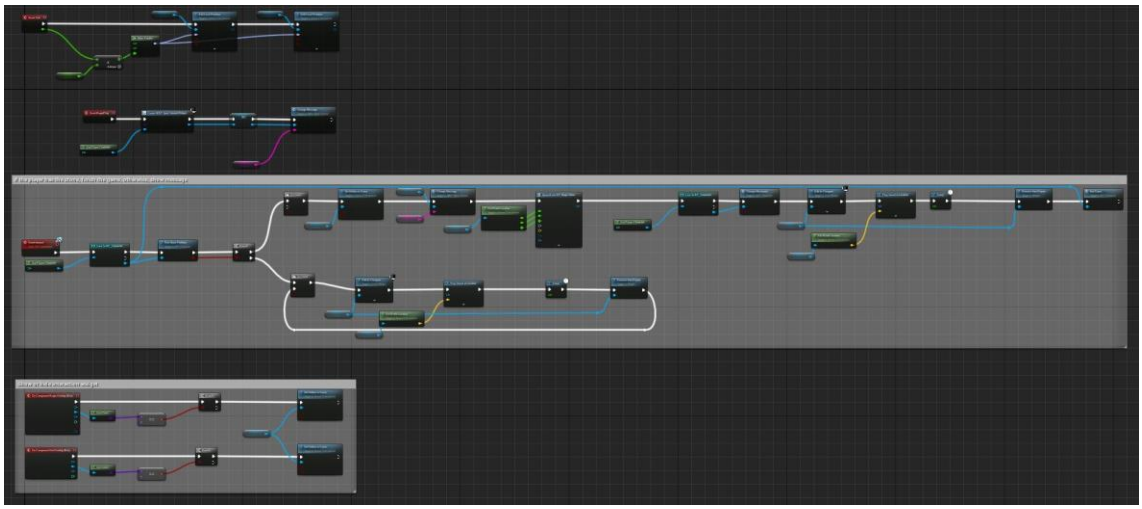

Figura 274. Visió general de BP_StonePlatform

Inicialització

El primer que es fa al *blueprint* és executar l'esdeveniment *BeginPlay*. En aquest mètode es crea el *widget WBP_SpiritTutorial* (Apartat 6.6.2), al qual se li afegeix un missatge per defecte utilitzant l'esdeveniment *ChangeMessage* del *widget* en qüestió. En aquest missatge se li demana al jugador que vagi a buscar la pedra.

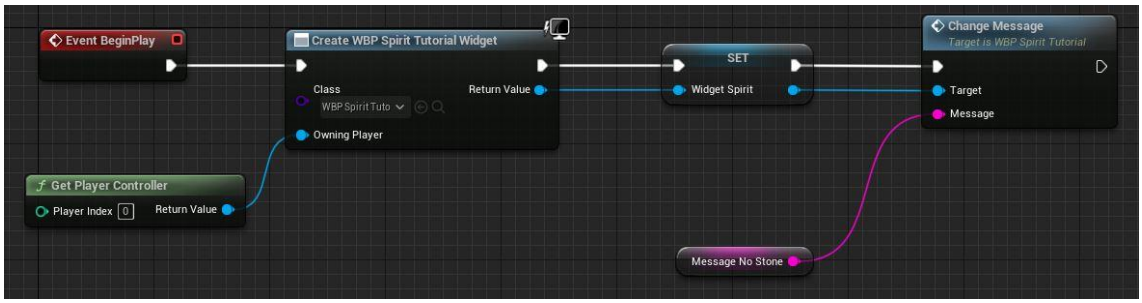

Figura 275. BeginPlay de BP_StonePlatform

Moviment

Com l'objectiu del joc es trobar la última pedra que li falta al Gran Esperit, s'han deixat col·locades les altres dues a la plataforma. A aquestes pedres se'ls ha afegit unes

partícules i a més se'ls ha aplicat moviment. El moviment s'ha pogut aconseguir gràcies a l'*Event Tick*, que aplica una certa rotació a l'eix Z a cada frame, tal i com es mostra a la Figura 276.

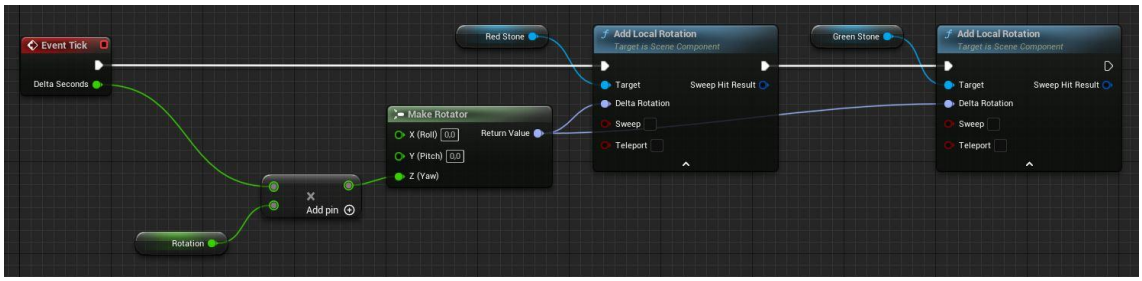

Figura 276. Event Tick de BP_StonePlatform

Col·lisions

Tant l'*OnComponentBeginOverlap* com l'*OnComponentEndOverlap* (Figura 277) mostren o oculten el missatge d'interacció del *widget* segons si el personatge s'apropa o s'allunya del *collider Box*.

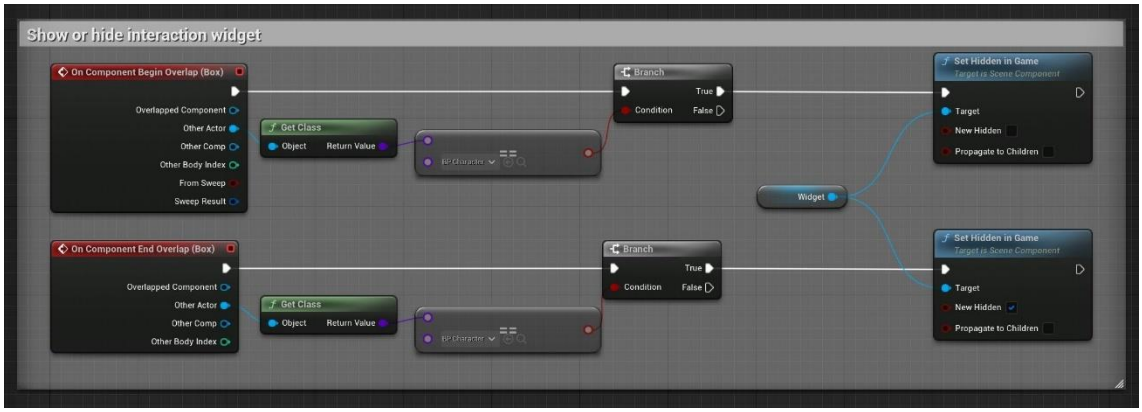

Figura 277. Overlaps de BP_StonePlatform

Interacció

En aquest *blueprint* es torna a trobar la implementació de l'esdeveniment *Interact* del *BPI_Interaction.* Com es pot veure a la Figura 278, el primer que fa és comprovar si el jugador ha aconseguit la pedra cridant a la funció *HasStonePlatform* (Apartat 6.3.14), segons el resultat es procedeix d'una manera o una altra.

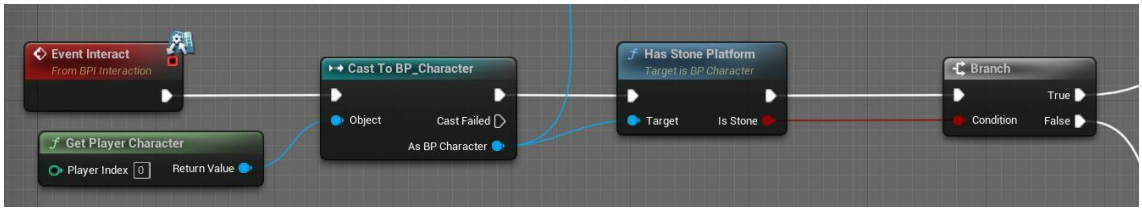

Figura 278. Esdeveniment Interact de BP_StonePlatform (primera part)

Si no s'ha aconseguit la pedra màgica (Figura 279), s'afegeix al viewport el *widget* creat al *BeginPlay,* on es demana al jugador que torni a buscar-la. A més, es reprodueix un so

per acompanyar al missatge que desapareix i s'elimina de la pantalla als 5 segons d'haver-lo mostrat.

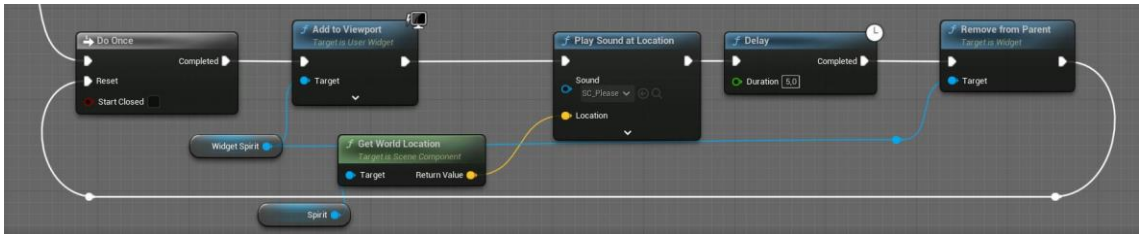

Figura 279. Esdeveniment Interact de BP_StonePlatform (segona part)

En canvi; tal i com es veu a Figura 280, Figura 281 i Figura 282; si el jugador sí té la pedra màgica, s'oculta el *widget* i es canvia el missatge que s'havia assignat per defecte per un altre. A continuació, s'instancia la pedra en la posició on es troba el component *arrow*; la funció és només aquesta, indicar la localització de l'objecte. Seguidament es deshabilita el moviment del jugador, ja que el joc ha finalitzat. A més, junt amb la reproducció d'un so, s'afegeix i mostra un missatge d'agraïment per part del Gran Esperit. Al cap de d'uns segons, s'elimina el missatge del *viewport* i es crida a la funció *EndGame* (Apartat 6.3.12), el qual porta al menú final. Lògicament, com aquesta és l'última acció que es realitza en el joc, no es pot tornar a executar.

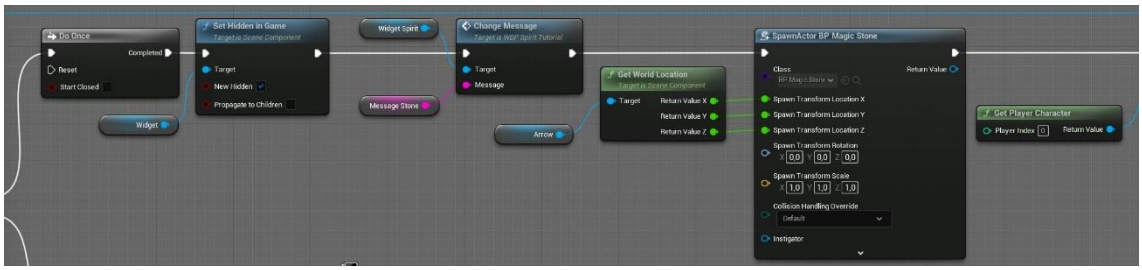

Figura 280. Esdeveniment Interact de BP_StonePlatform (tercera part)

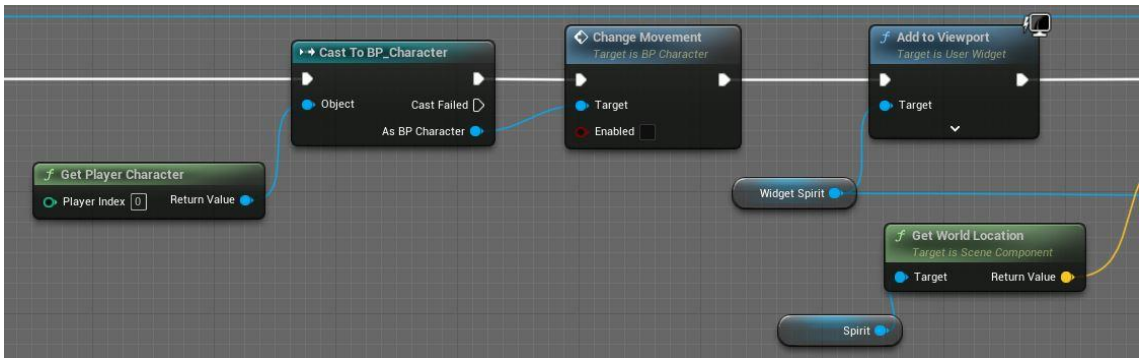

Figura 281. Esdeveniment Interact de BP_StonePlatform (quarta part)

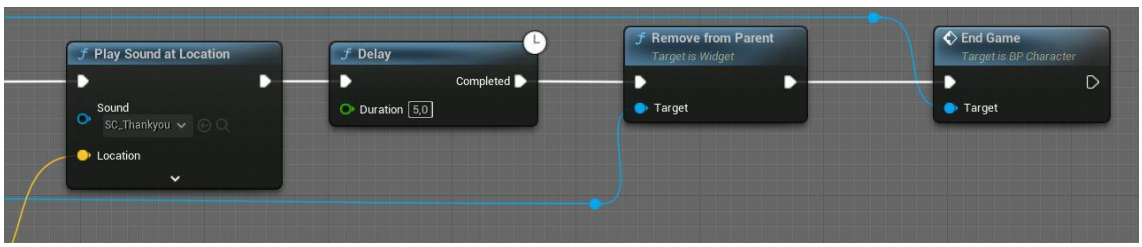

Figura 282. Esdeveniment Interact de BP_StonePlatform (cinquena part)

6.6. Interfícies

6.6.1. Menús

Tots els menús s'han implementat amb l'ajuda d'un *plugin* gratuït extret del Marketplace de l'Unreal, anomenat *UINavigation 2.0*, el qual facilita la navegació entre els botons dels menús, tant amb teclat i ratolí com amb comandament. Tot i així, s'ha hagut d'implementar la resta de funcionalitats.

UINavigation

Un cop instal·lat el *plugin*, s'ha canviar la classe pare del *CharacterController* per l'*UINavController*. Amb això, s'afegeix el component *UINav PCComp* (Figura 283). Totes aquestes modificacions permeten al jugador utilitzar les funcionalitats de l'UINavigation i poder implementar algunes accions al *blueprint* del *player controller*.

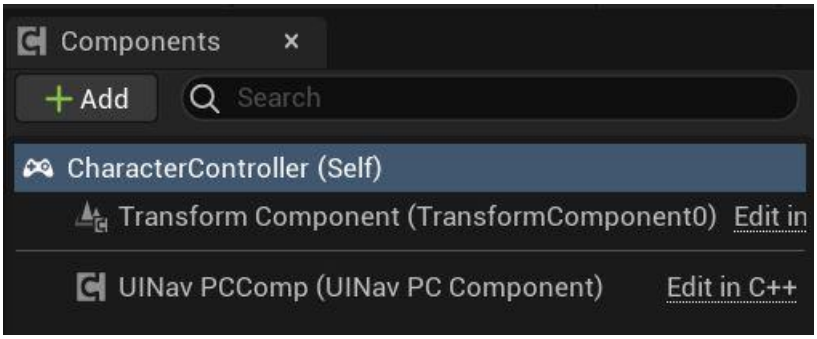

Figura 283. Components del CharacterController

A més, per a poder utilitzar l'*UINavigation* als menús, és necessari que es canviï el paràmetre *Parent Class* que es troba al *Class Settings* de cada menú, tal i com es veu a la Figura 284. A més, a la pestanya *Class Defaults* també s'ha d'activar la casella *Use Button States* i *Is Focusable*, com es mostra a la Figura 285 i Figura 286 respectivament. Activar aquests paràmetres, permet i facilita la navegació entre els botons.

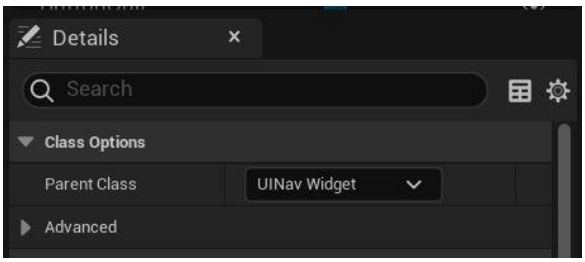

Figura 284. Class Settings dels menús de tots els menús

Figura 285. Class Defaults de tots els menús (primera part)

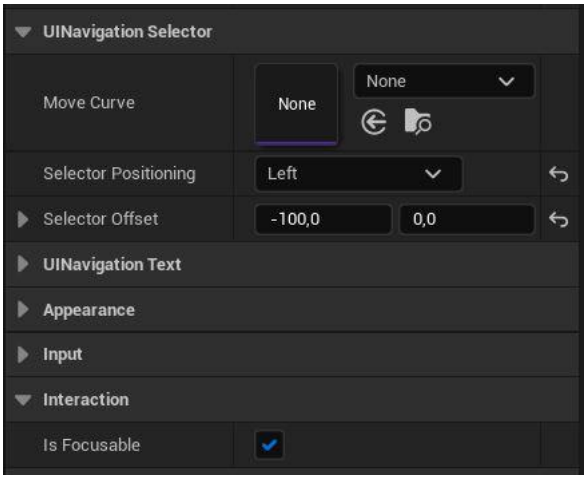

Figura 286. Class Defaults de tots els menús (segona part)

A més, pel correcte funcionament és necessari que s'utilitzin els components d'aquest plugin, com per exemple, l'UINavButton (Figura 287), que representa al *Button*, i el *Selector*, que permet afegir una icona al davant del botó seleccionat.

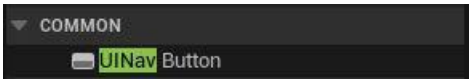

Figura 287. Component UINav

Per altra banda, pel que fa al graf principal dels *blueprints* dels diferents menús, és necessari cridar al mètode Esdeveniment *ReadyforSetup* (Figura 288) al qual se li ha de passar el component *Vertical Box* que recull els botons creats al node *AppendNavigationGrid1D* i assignar-li valor a la variable UINavComponent, que ja ve definida per defecte.

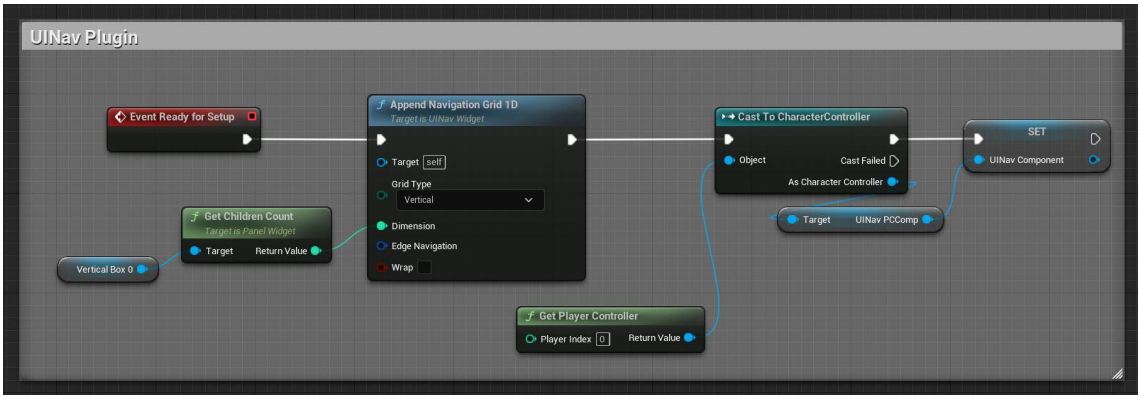

Figura 288. Esdeveniment Ready for Setup de tots els menús

Per acabar, també s'ha afegit l'esdeveniment *OnInputChanged* (Figura 289), en el que es controla si hi ha un canvi de dispositiu d'entrada. Això fa que es mostri o no el cursor en pantalla.

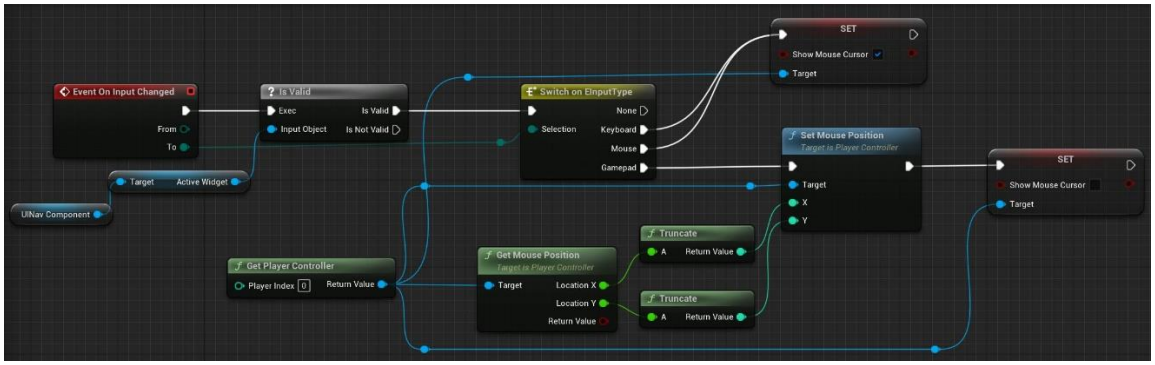

Figura 289. Esdeveniment OnInputChanged de tots els menús

MainMenu

El menú principal s'utilitza al mapa *MainMenuMap*, com s'ha comentat en apartats anteriors. En el *blueprint* del nivell és on s'afegeix el menú al *viewport* (Figura 290).

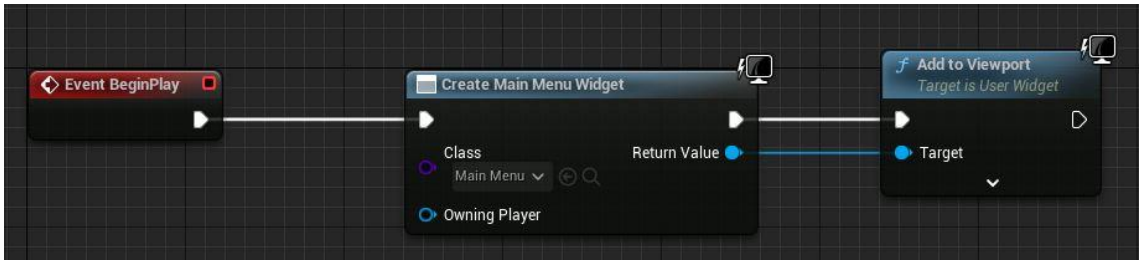

Figura 290. Level Blueprint de MainMenuMap

El menú principal consta de tres botons; entre altres components (Figura 291); el d'*Start,* per a iniciar la partida; el *Controls*, per a consultar els controls del teclat i comandament, i el *Quit*, per tancar el joc. Aquest menú és el primer que es mostra a l'executar el joc, també es pot tornar a ell des de qualsevol dels altres menús.

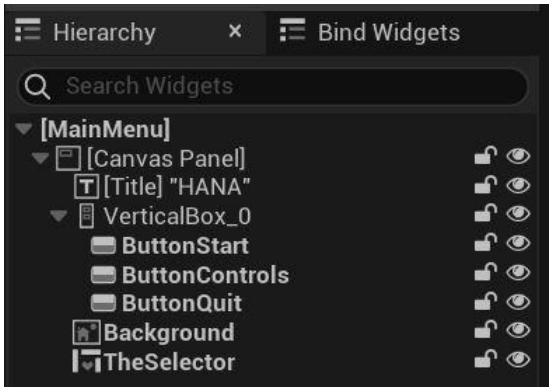

Figura 291. Components del menú principal

Tots els components i la implementació es troben al *widget blueprint MainMenu*, a la Figura 292 es mostra una visió global del graf.

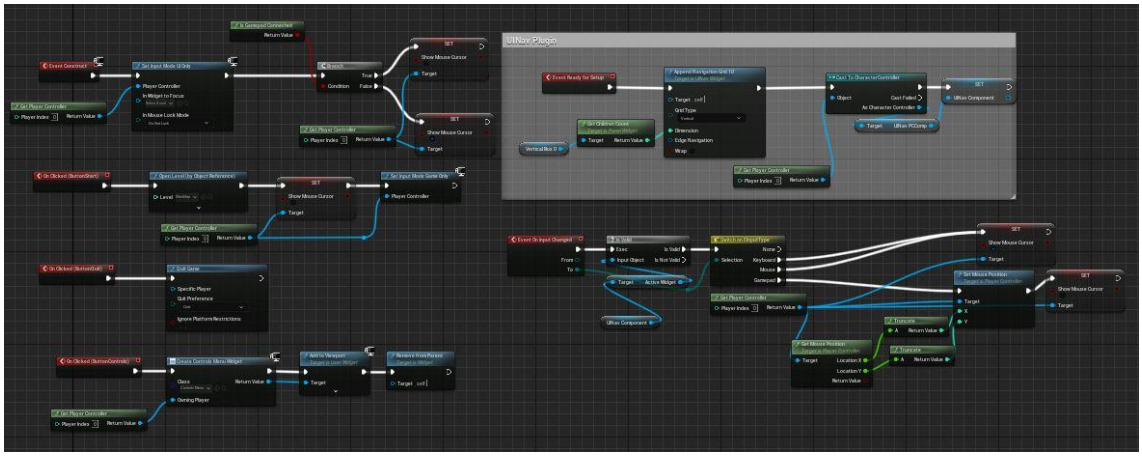

Figura 292. Visió global de MainMenu

Quan aquest menú s'afegeix al viewport, es crida a l'*Event Construct* (Figura 293), el qual canvia l'*input mode* a *UI Only*, és a dir, només estan disponibles els *inputs* per la interfície d'usuari. A més, segons si el comandament està connectat, es mostra el cursor o no.

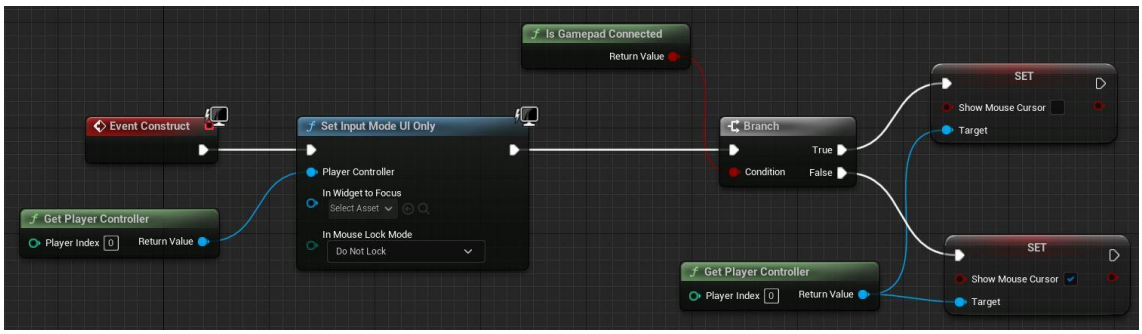

Figura 293. Event Construct del MainMenu

Per altra part, si es vol iniciar el joc, el jugador ha de prémer el botó *Start*. Si ho fa, s'executa l'esdeveniment *OnClicked* del *ButtonStart* (Figura 294). Aquest mètode obre el mapa *MainMap*, amaga el cursor i canvia l'*input mode* al de joc.

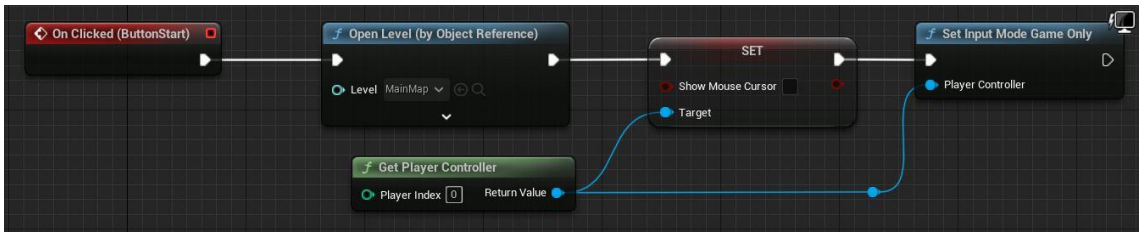

Figura 294. OnClicked del ButtonStart del MainMenu

Si el jugador vol consultar els controls que estan disponibles al joc, i prem el botó *Controls*, s'inicia l'esdeveniment *OnClicked* del *ButtonControls* (Figura 295), el qual afegeix el *widget ControlsMenu* al *viewport* i elimina l'actual.

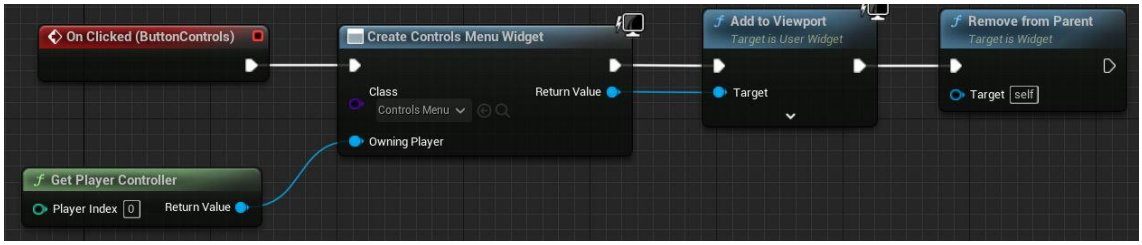

Figura 295. OnClicked del ButtonControls del MainMenu

Per últim, si el que es prem és el botó *Quit* per a poder sortir del joc, simplement es crida al node *QuitGame*, tal i com es pot veure a la Figura 296.

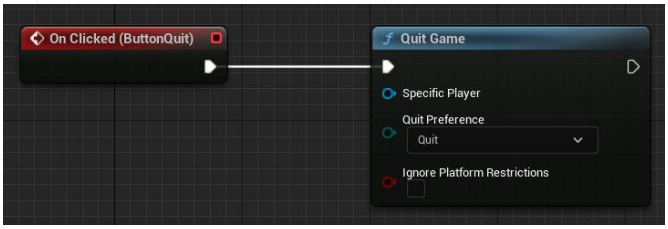

Figura 296. OnClicked del Quit del MainMenu

PauseMenu

El menú de pausa es pot invocar al llarg de tota la partida amb la tecla *Escape* del teclat o el botó menú del comandament. Aquest menú s'executa des del *Character Controller*, el qual consta d'un esdeveniment *InputAction PauseMenu*, com es pot veure a la Figura 297 i Figura 298. Aquest esdeveniment comprova si el joc està en pausa o no, ja que és possible que es cridi des del menú, el que vol dir que s'ha de tancar. En aquest últim cas, s'elimina el *widget actiu* (només si es tracta del menú de pausa o controls), s'indica que el menú no està en pausa amb el node *SetGamePause*, s'oculta el cursor, es canvia l'*input mode,* i per acabar, es crida al mètode *ToggleMusic* per canviar la música de fons. Aquest últim mètode s'explica a l'Apartat 6.9.2. En canvi, si el joc no està pausat, el primer que es fa és crear el *widget*, el qual s'afegeix al *viewport*. A continuació, es comprova si és necessari que es mostri el cursor o no, també canvia l'*input mode*. A més, com en l'altre cas, es torna a canviar la cançó de fons i, per acabar, es pausa el joc.

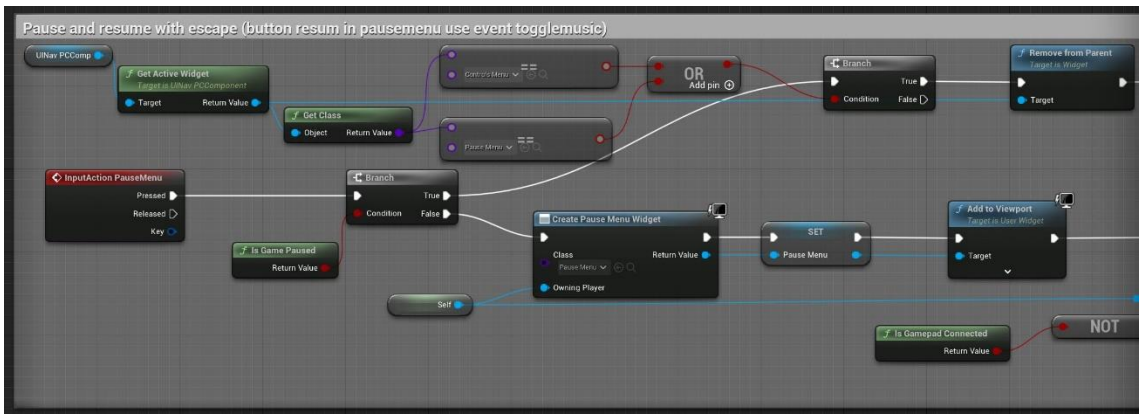

Figura 297. InputAction PauseMenu del CharacterController (primera part)

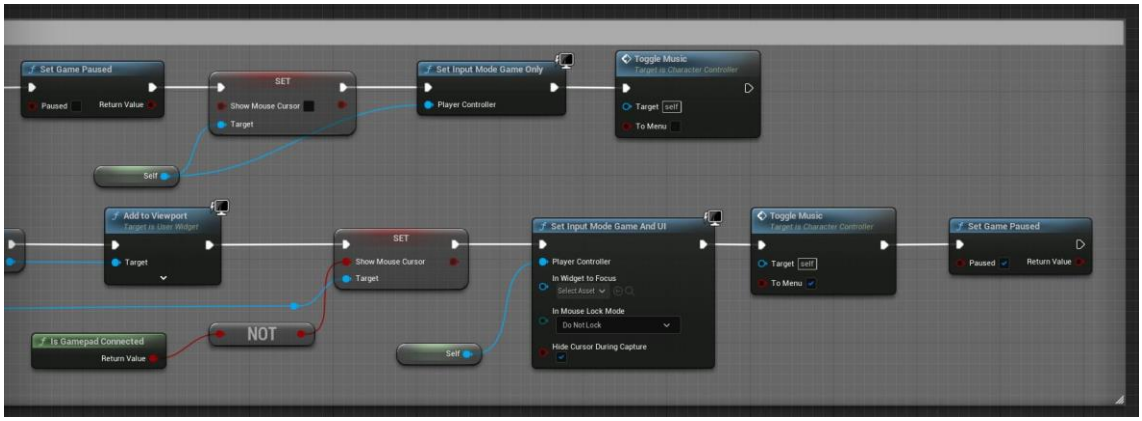

Figura 298. InputAction PauseMenu del CharacterController (segona part)

Un cop s'ha afegit el menú al *viewport*, es pot veure com aquest, a l'igual que el principal, consta de tres opcions: el botó *Resume*, per a reprendre el joc; el botó *Controls*, per a consultar els controls, i el botó *Return to Menu*, per a tornar al menú inicial. La implementació d'aquest menú es troba al *widget blueprint PauseMenu* (Figura 299).

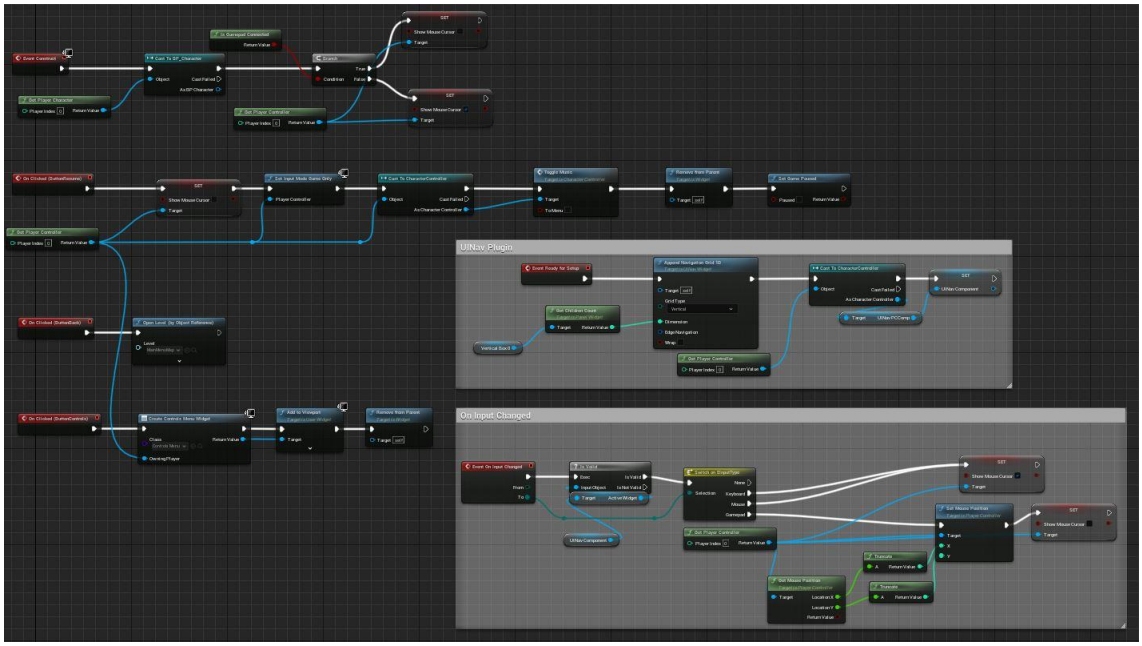

Figura 299. Visió global del PauseMenu

Quan el joc es pausa, el primer que s'executa és el mètode *Event Construct* (Figura 300), que s'assegura que el cursor es mostri o s'amagui segons el dispositiu d'entrada.

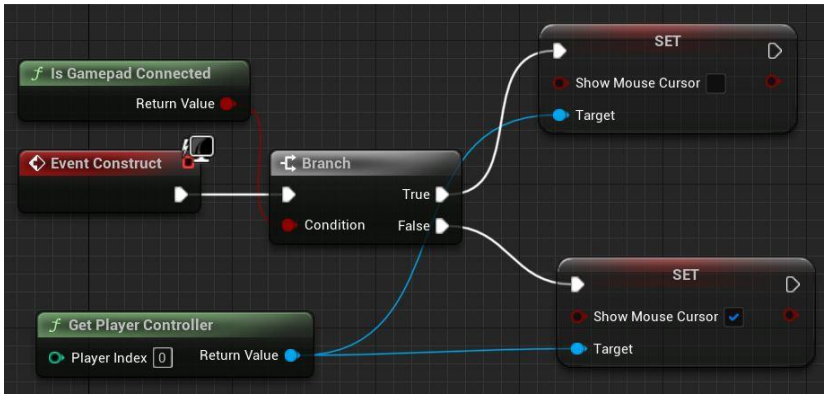

Figura 300. EventsConstruct del PauseMenu

En cas que el jugador vulgui tornar a reprendre la partida utilitzant el *button* i no tornant a prémer la tecla o botó que pausa, s'executa l'esdeveniment *OnClicked* del *ButtonResume* (Figura 301 i Figura 302). Aquest esdeveniment fa el mateix que el del *Character Controller.* Torna a canviar tots els valors i paràmetres necessaris per a que es torni a la partida i el joc deixi d'estar pausat.

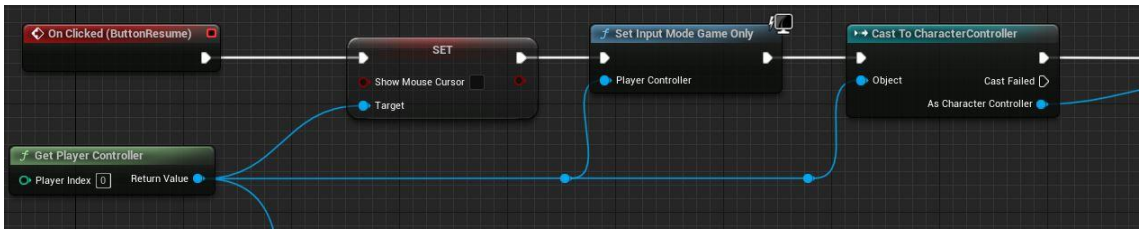

Figura 301. OnClicked del ButtonResume del PauseMenu (primera part)

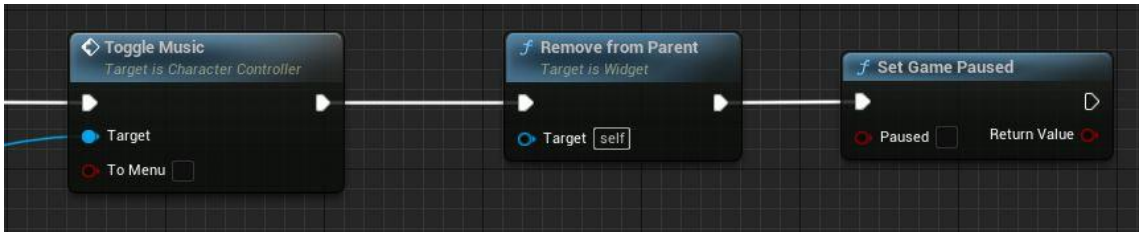

Figura 302. OnClicked del ButtonResume del PauseMenu (segona part)

L'*OnClicked* del *ButtonControls* (Figura 303) es crida quan el jugador escull l'opció *Controls*. Aquesta opció simplement crea el *widget ControlsMenu*, l'afegeix al *viewport* i elimina l'actual.

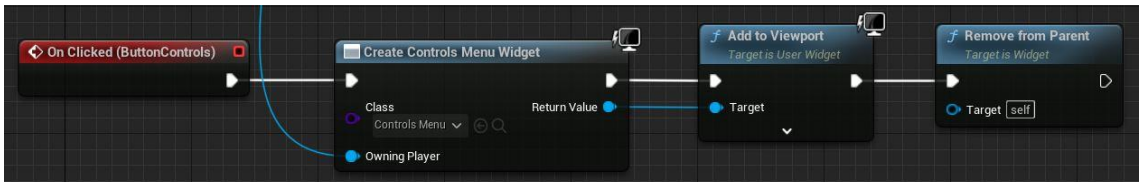

Figura 303. OnClicked del ButtonControls del PauseMenu

Per acabar, si el jugador prem el botó *Return to Menu,* s'executa *l'OnClicked* del *ButtonBack*, el qual, com es pot veure a la Figura 304, obre el mapa del menú principal.

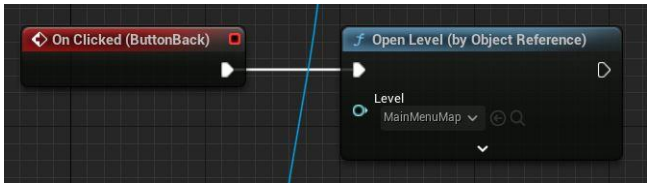

Figura 304. OnClicked del ButtonBack del PauseMenu

ControlsMenu

A la pantalla dels controls s'hi pot accedir des del menú principal o el de pausa. La implementació es troba al *widget blueprint* ControlsMenu (Figura 305).

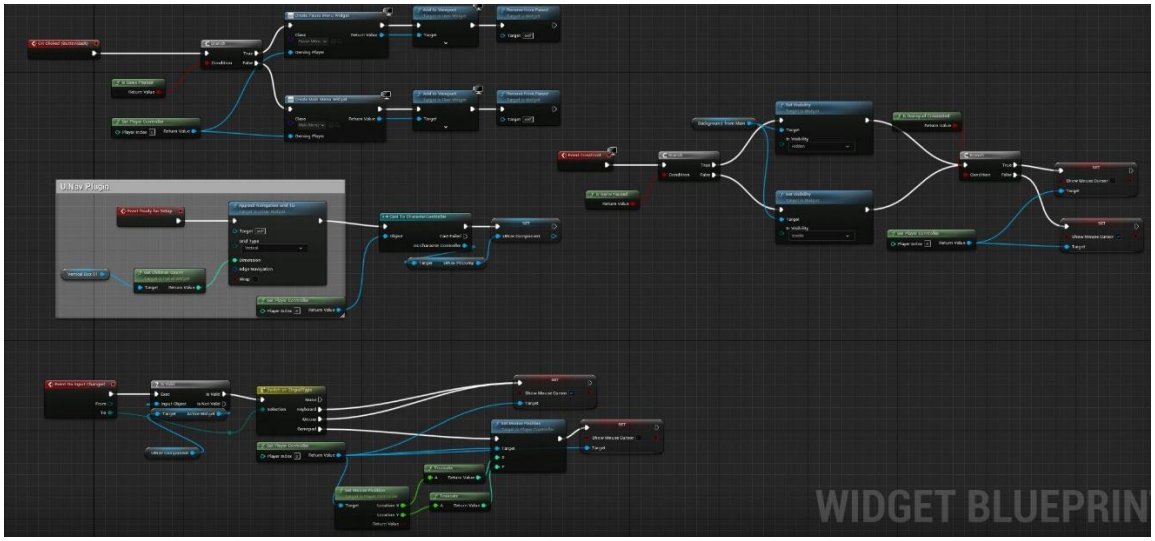

Figura 305. Visió general del ControlsMenu

En aquest *widget*, tot i tenir una gran quantitat de components que mostren els diferents controls (Figura 306 i Figura 307), només es pot interactuar amb el botó *Back* per a tornar al menú anterior.

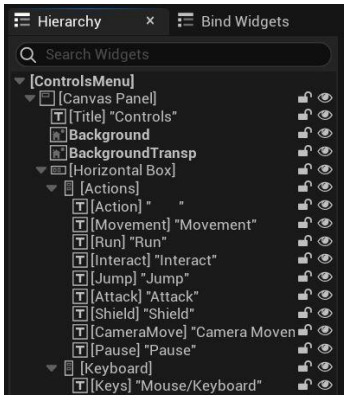

Figura 306. Components de ControlsMenu (primera part)

Figura 307. Components de ControlsMenu (segona part)

Quan els diferents menús el criden, el primer que s'executa és l'E*vent Construc*t (Figura 308), el qual mostra una imatge de fons segons si l'anterior menú és el principal o el de pausa. A més, comprova el dispositiu d'entrada, per així mostrar o no el cursor.

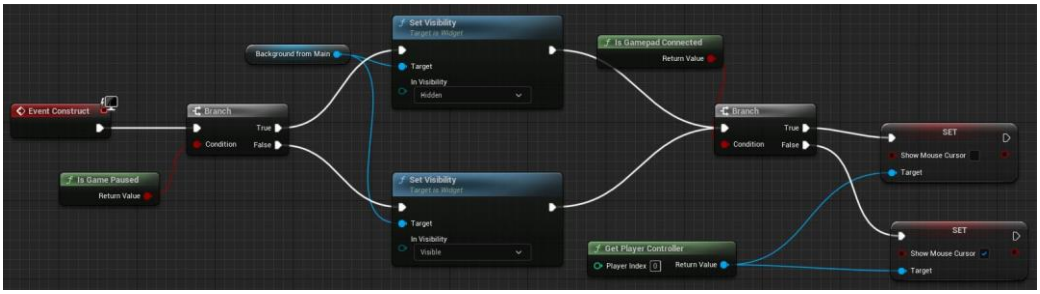

Figura 308. EventConstruct del ControlsMenu

Per últim, en cas que el jugador decideixi tornar enrere prement el botó *Back*, s'iniciaria el mètode *OnClicked* del *ButtonBack,* que es mostra a la Figura 309.

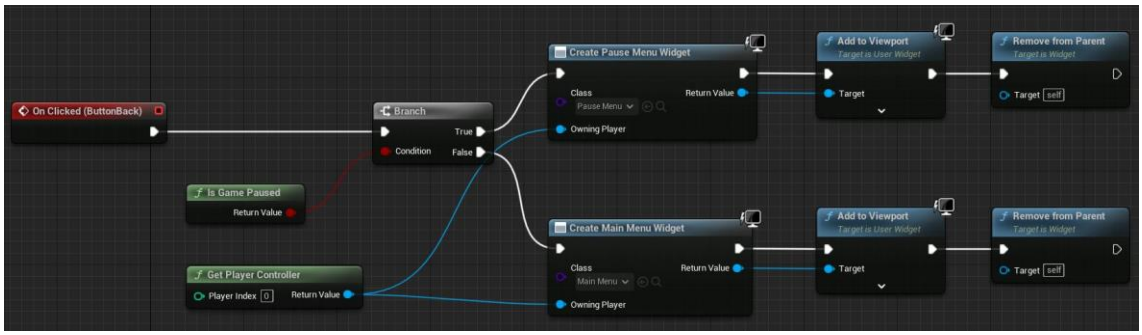

Figura 309. OnClicked del ButtonBack del ControlsMenu

DeadMenu

El menú que es mostra quan el personatge ha perdut totes les seves vides, és el *DeadMenu*, el qual dona l'opció de tornar a jugar o anar al menú principal. En la Figura 310 es mostra una visió global de la seva implementació.

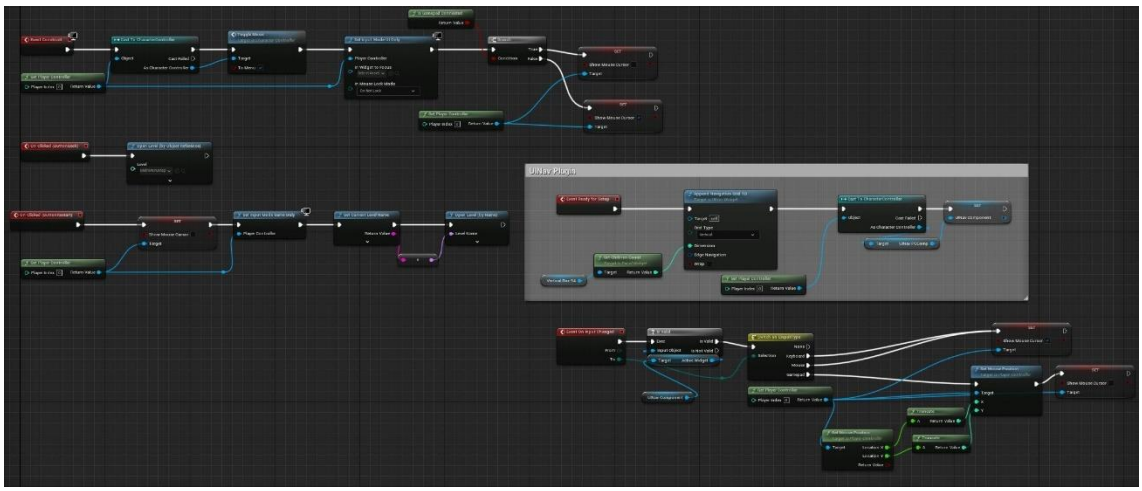

Figura 310. Visió global del DeadMenu

L'*Event Construct* (Figura 311) s'encarrega de posar a punt la pantalla. Aquest mètode canvia la música de fons, l'*input mode* i comprova el dispositiu d'entrada per a poder mostrar o amagar el cursor, tal i com s'ha estat fent en els altres menús.

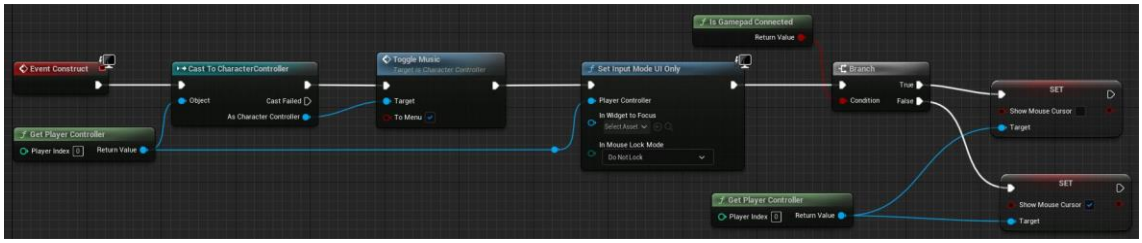

Figura 311. EventConstruct del DeadMenu

Si el jugador decideix prémer el botó *Restart*, l'esdeveniment *OnClicked* del *ButtonRestart* (Figura 312) amaga el cursor i canvia l'*input mode*. Per acabar, obre el mapa actual, el del joc. D'aquesta manera es reinicia la partida.

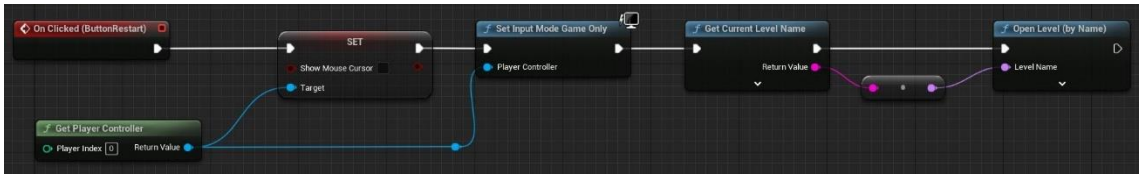

Figura 312. OnClicked del ButtonRestart del DeadMenu

Per últim, el mètode *OnClicked* del *ButtonBack,* torna al menú principal, per fer-ho obre el mapa *MainMenuMap*, tal i com es pot veure en la Figura 313.

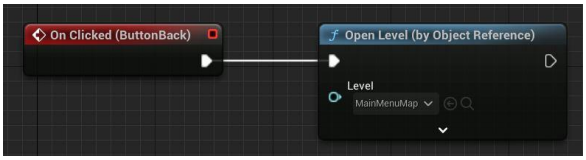

Figura 313. OnClicked del ButtonBack del DeadMenu

FinalMenu

En el moment en que el jugador entrega la pedra màgica al Gran Esperit, el joc s'ha acabat. Així doncs, als pocs segons s'obre el menú final, on es mostra un missatge i es dona l'opció de tornar al menú principal. La implementació d'aquest menú es troba al *widget blueprint FinalMenu* (Figura 314Figura 315).

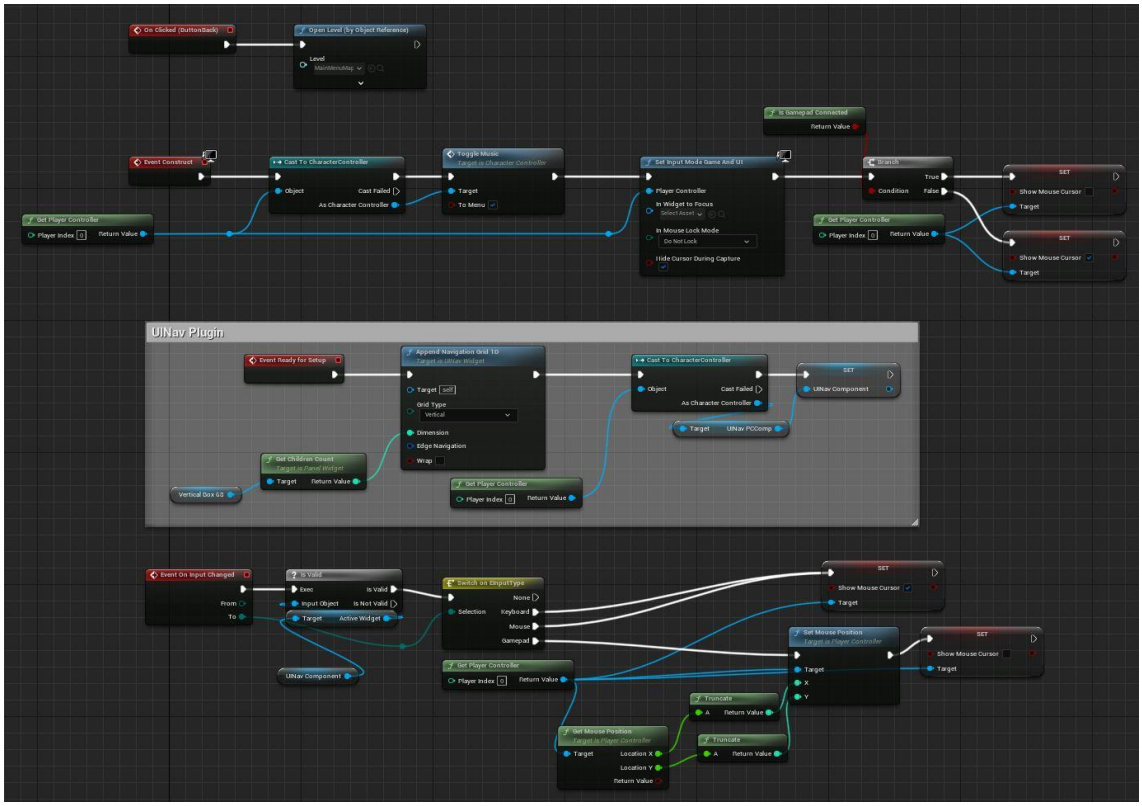

Figura 314. Visió global del FinalMenu

L'*EventConstruct* (Figura 315) és igual que al de molts dels altres menús. Primer de tot, canvia la música de fons i l'*input mode*, i per acabar, mostra o oculta el cursor, segons el dispositiu d'entrada connectat.

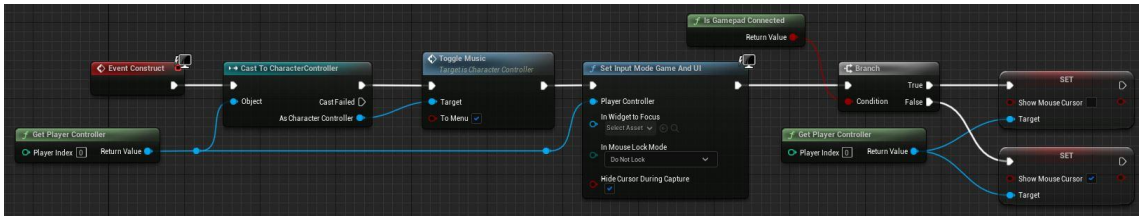

Figura 315. EventConstruct del FinalMenu

Finalment, l'*OnClicked* del *ButtonBack* (Figura 316) obre el mapa MainMenuMap, que porta al menú principal del joc.

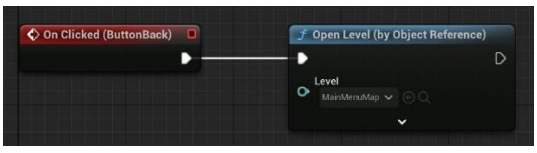

Figura 316. OnClicked del ButtonBack del FinalMenu

6.6.2. Altres Widgets

A part dels menús que s'han vist en l'apartat anterior, també s'han dissenyat i implementat altres *widgets*, els quals són molt importants, ja que són els encarregats de proporcionar ajuda i *feedback* al jugador.

HUD

El *WBP_HUD* és el *widget* encarregat de mostrar la vida actual del personatge. A més també mostra una imatge de la pedra màgica quan el jugador l'ha aconsegueix.

L'*EventConstruct* (Figura 317) s'executa en quant el *BP_Character* afegeix el *HUD* al *viewport*. Aquest mètode espera rebre la crida dels *Event Dispatchers UpdateLifeUI* i *UpdateStoneUI.*

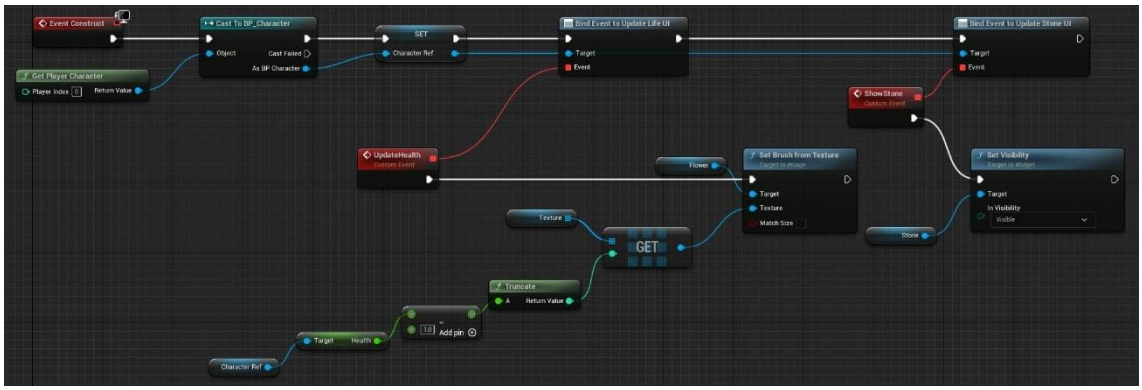

Figura 317. EventConstruct del WBP_HUD

El mètode *UpdateHealth* (Figura 318) s'executa quan, des del *blueprint* del personatge, es crida a l'*Event Dispatcher UpdateLifeUI*, per exemple quan perd una vida o es cura. Un cop iniciat, canvia la textura de la flor, la qual representa la vida, per a que indiqui el número de pètals que s'han de mostrar. Aquesta textura es troba dins d'un *array* que conté totes les imatges necessàries i, segons la vida actual del personatge, s'accedeix a una posició o una altra.

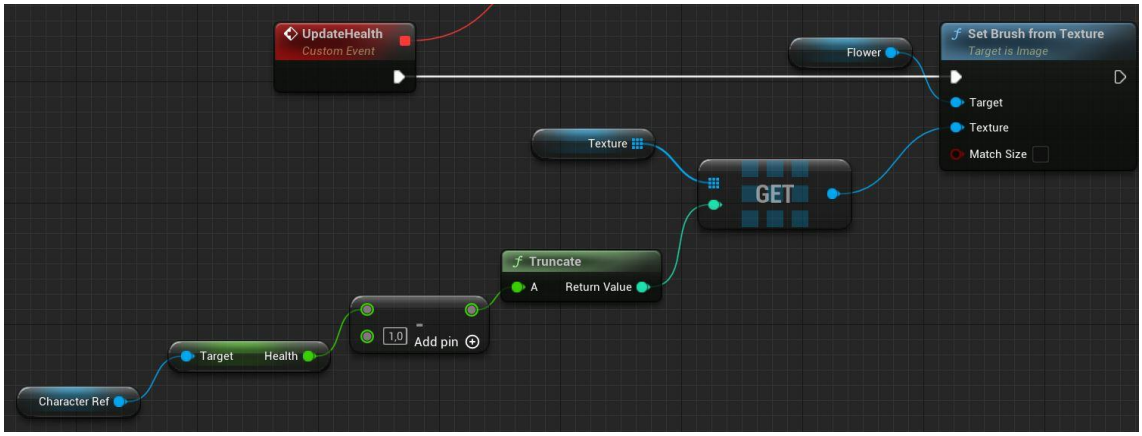

Figura 318. UpdateHealth

Per altra part, el mètode *ShowStone* (Figura 319) s'executa quan es crida l'*Event Dispatcher UpdateStoneUI*, això passa únicament quan el jugador elimina al primer grup d'enemics i aconsegueix la pedra màgica. Quan això ocorre, es canvia la visibilitat del component *Stone* a visible.

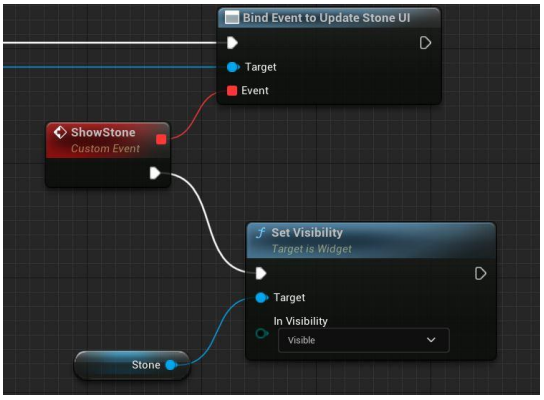

Figura 319. ShowStone

Interacció

Com s'ha vist en apartats anteriors, hi ha diversos objectes que mostren un missatge d'interacció quan el personatge s'apropa el suficient al *collider* d'aquests elements.

A l'iniciar aquest *widget* en els diferents objectes que l'implementen, es crida a l'*EventConstruct* (Figura 320), el qual executa el mètode *UpdateText* i espera a que es cridi l'*Event Dispatcher UpdateInteractText*, que també utilitza el mètode anterior.

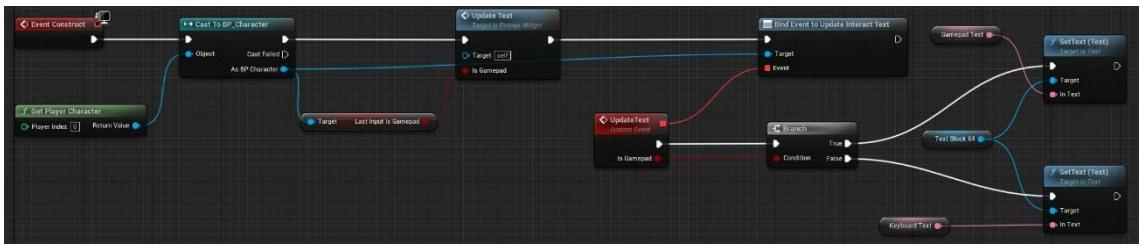

Figura 320. EventConstruct del PromptWidget

L'*UpdateText* (Figura 321) canvia el text del component que conté el missatge segons si s'està utilitzant comandament o teclat. Això s'executa cada cop que es canvia de dispositiu d'entrada.

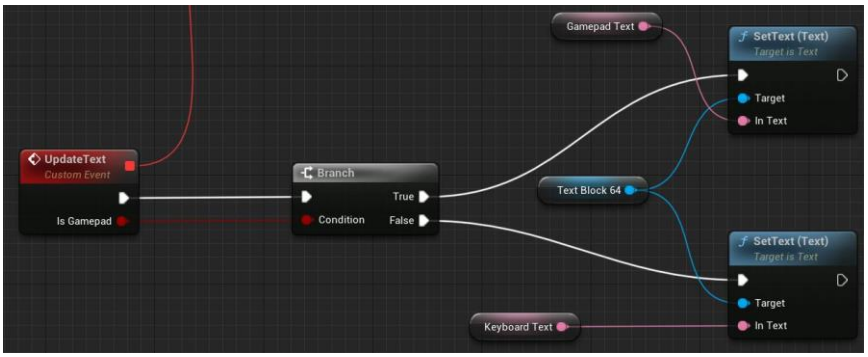

Figura 321. UpdateText

Missatges dels esperits ajudants

Els missatges i pistes dels esperits ajudants que es van trobant pel mapa tenen associat un *widget* anomenat *WBP_LilSpiritMessages*. Aquest *widget* només conté un mètode,

el *ChangeSpiritMessage*, el qual, simplement, canvia el text pel que se li passa pel paràmetre *input* de la funció, tal i com es pot veure a la Figura 322.

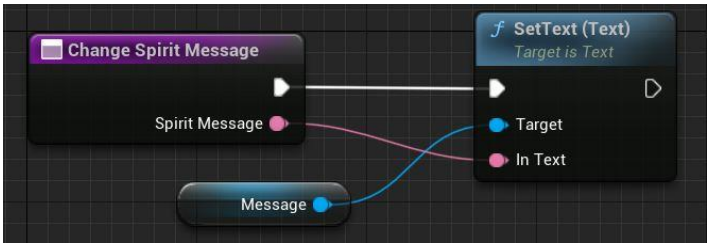

Figura 322. ChangeSpiritMessage

Missatges del Gran Esperit

Els missatges que es mostren a la part inferior de la pantalla són del Gran Esperit del bosc. En aquest cas, cap objecte conté un component *widget* d'aquest tipus, ja que no es mostra en un punt en concret del *blueprint* que l'invoca, sinó que s'afegeix al *viewport*. Aquest *widget* està implementat a *WBP_SpiritTutorial*, el qual conté dos mètodes: *StartTutorial* i *ChangeMessage*.

StartTutorial (Figura 323, Figura 324, Figura 325 i Figura 326) és un mètode que, a partir d'un *array* d'strings, escriu el missatge lletra per lletra. Quan acaba amb una part, continua amb la següent posició de l'*array TutorialArray*, fins que mostra tot el contingut. Primer de tot, es guarda en una variable el contingut de la posició de l'*array* d'*strings* que s'ha d'escriure, indicat per l'*IndexArray*. Seguidament, utilitzant un *timer*, es crida a l'esdeveniment *Texting*. Això es fa cada *n* segons, que es troben definits a la variable *TypeSpeed*. A continuació, a la variable *DisplayedText* se li va afegint cada caràcter del missatge, el qual també es tracta com un altre *array*, i es canvia amb el mètode *SetText*. Després, s'augmenta l'índex que correspon a una part del missatge. Si aquest és igual a la longitud de la part del missatge en qüestió, es reinicien les variables i s'augmenta l'índex de l'*array* que conté tot el missatge, d'aquesta manera es pot passar a la següent secció. Si s'han escrit totes les parts, s'elimina el *timer* i s'inicia un *delay* d'1 segon. Per acabar, es torna a habilitar el moviment del personatge que s'havia parat prèviament des del *BP_Character*, per a que aquest no es pogués moure durant el tutorial.

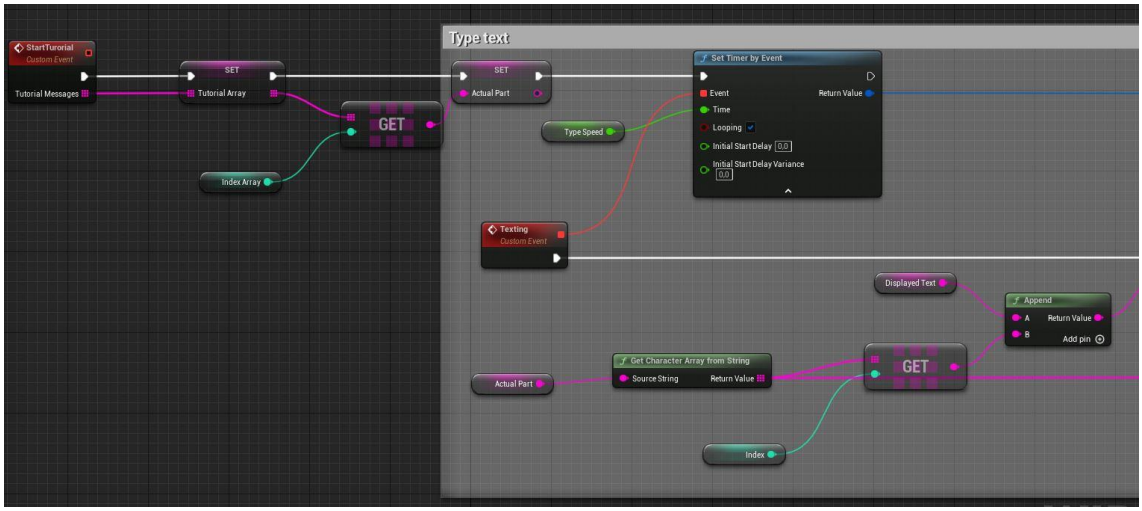

Figura 323. StartTutorial (primera part)

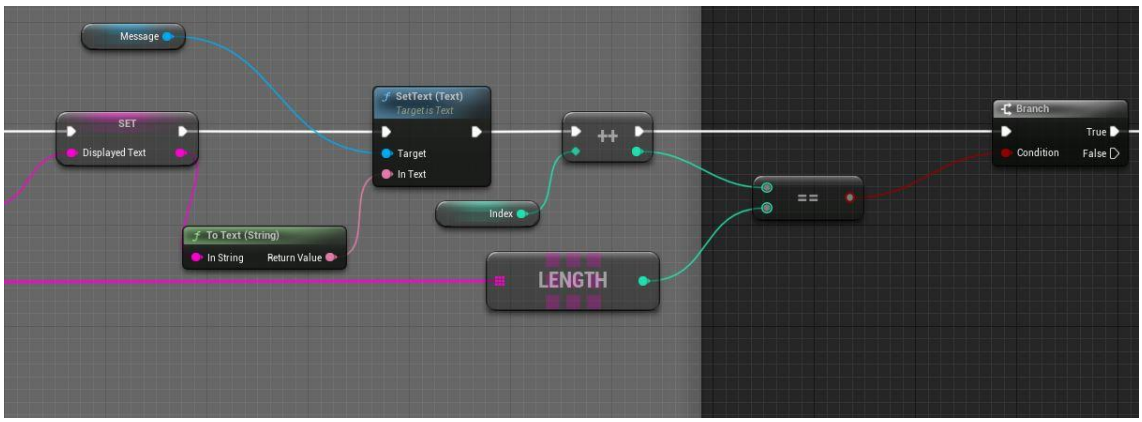

Figura 324. StartTutorial (segona part)

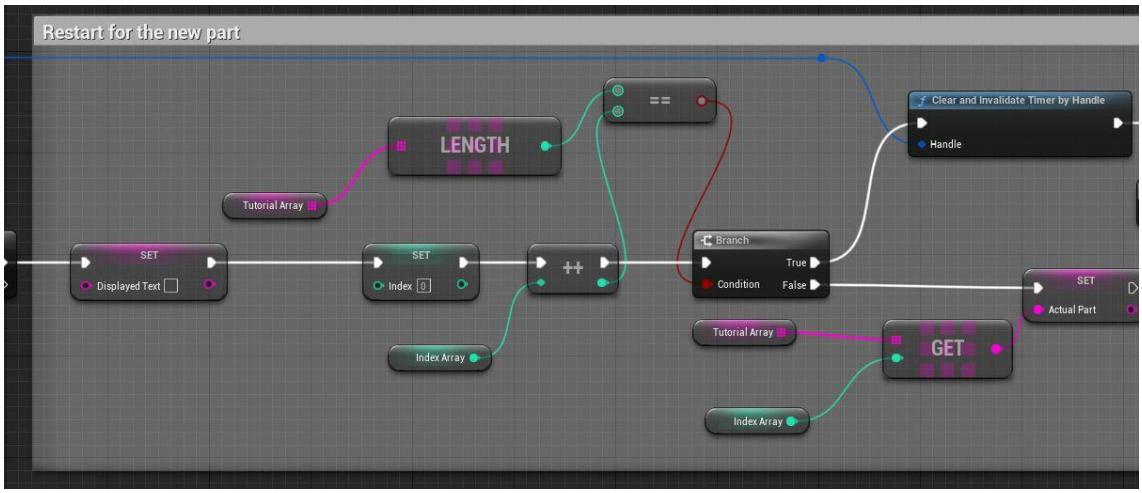

Figura 325. StartTutorial (tercera part)

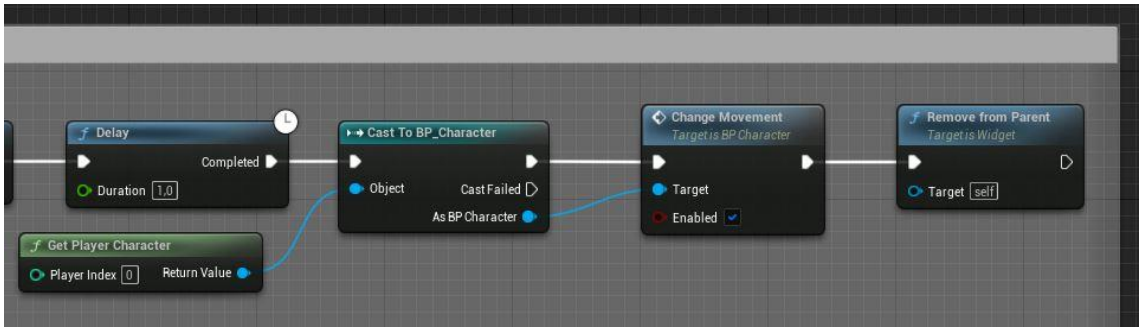

Figura 326. StartTutorial (quarta part)

ChangeMessage (Figura 327) s'utilitza per a la resta de missatges del Gran Esperit. Aquesta funció canvia el text del missatge pel que se li passa pel paràmetre *input*.

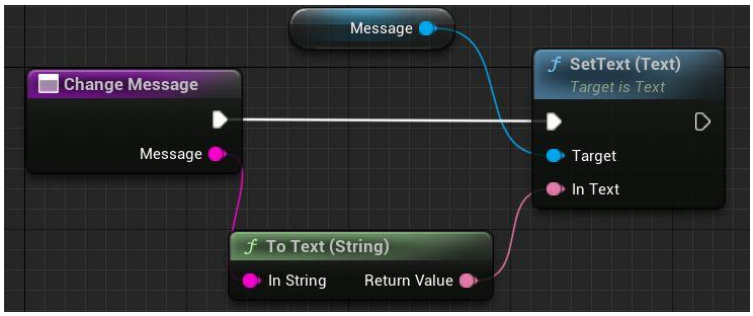

Figura 327. ChangeMessage

6.7. Runtime Virtual Texturing

Per a poder utilitzar l'RVT (Apartat 4.1.2) al projecte és necessari habilitar el paràmetre *Enable virtual texture support* (Figura 328), el qual es troba a *Project Settings.*

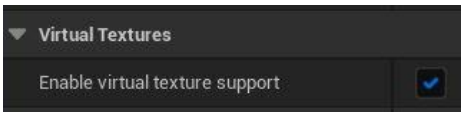

Figura 328. Habilitar RVT

6.7.1. Landscape

Per a fer servir l'RVT al terreny, primer de tot, s'han creat dos textures RVT (Figura 329), Color, que és la que conté la informació del color i altres característiques (Figura 330), i Height, la que conté la informació sobre l'altura del terreny (Figura 331).

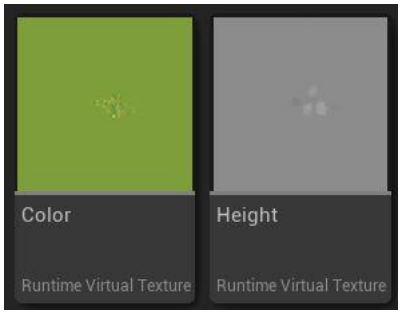

Figura 329. Textures RVT

Figura 330. Textura Color

Figura 331. Textura Height

A continuació, s'han creat els elements *RuntimeVirtualTextureVolume* (Figura 332), un per cada textura, ja que se les han d'assignar, tal i com es veu a la Figura 333 i la Figura 334. Aquests *volumes* han d'ocupar tot l'espai de joc, per facilitar-ho es pot associar el *landscape* a aquests actors i seguidament, clicar el botó *Set Bounds,* que els escala amb la mida del terreny.

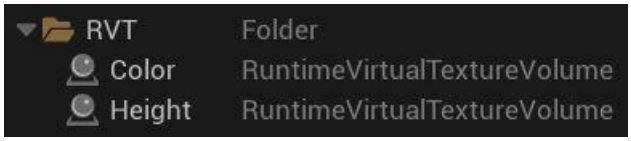

Figura 332. RVT Volume

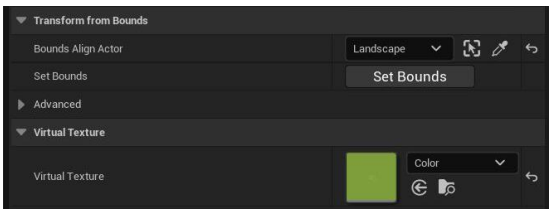

Figura 333. RVTVolume Color

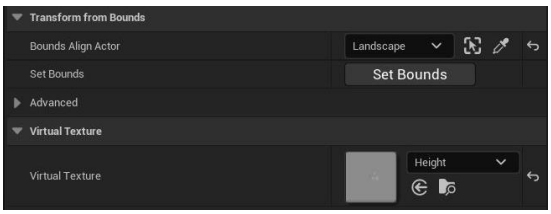

Figura 334. RVTVolume Height

Un cop estan definits els *volumes* i les seves textures, ja es pot crear el material pel terreny (Figura 335). En el material, que en aquest cas s'anomena *MM_Landscape_RVT,* es defineixen les diferents capes de color, que s'utilitzen per pintar el terra; el paràmetre *Specular* i el *Roughness*; i l'alçada. Tots aquests paràmetres s'assignen tant al node dels atributs del propi material, com al del *RVT Output*.

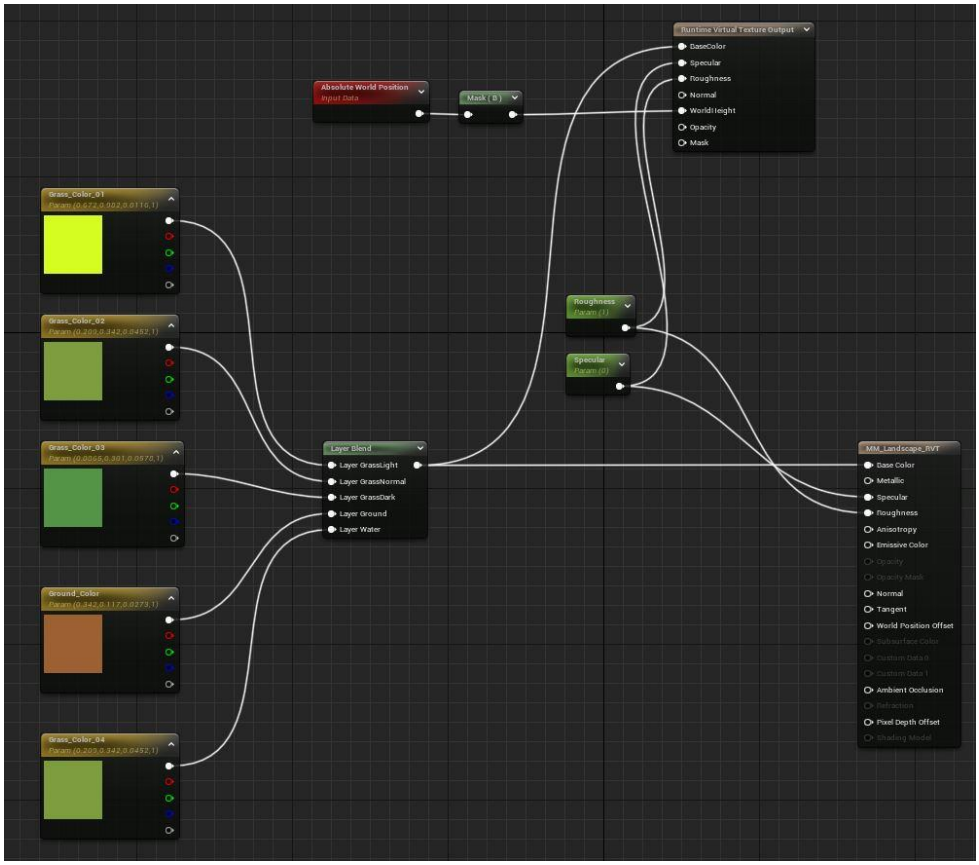

Figura 335. Material MM_Landscape

Un cop creat el material, s'assigna a l'objecte *Landscape* de l'escena. Al fer-ho, al mode *Landscape,* concretament a la subpestanya *Paint,* apareixen les capes definides al material, tal i com es pot veure a la Figura 336. Per a cadascuna s'ha de crear un fitxer *LayerInfo* (Figura 337), que conté informació de la capa. Aquest fitxer s'ha modificat per a afegir un *Physic Material*, això s'explica més endavant a l'Apartat 6.9.3.

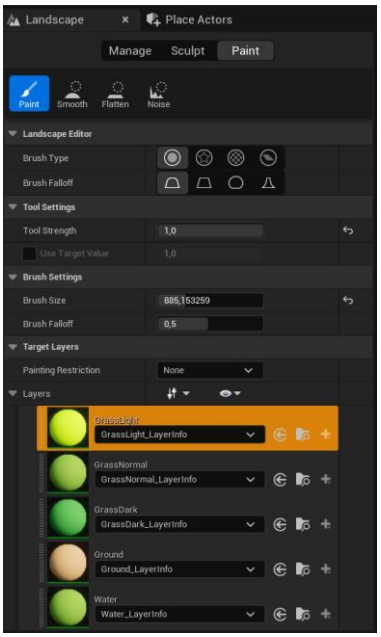

Figura 336. Layer en el mode paint

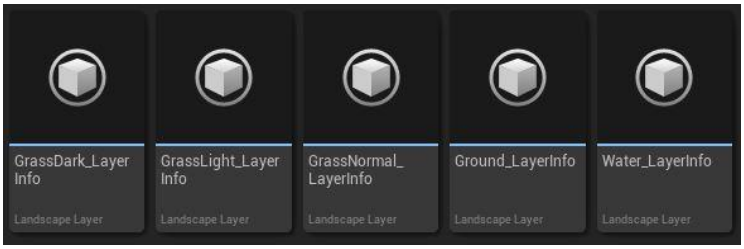

Figura 337. LayerInfo

Amb totes aquestes configuracions fetes, ja és possible començar a pintar el terreny dels diferents colors definits (Figura 338).

Figura 338. Terreny amb els diferents colors de les capes

6.7.2. Herba

El principal motiu pel que s'han creat les RVT, és l'herba. Es volia que l'herba canviés de color segons el del terreny. Això fa que la vegetació de l'entorn es vegi més plana i sense transicions estranyes entre el terra i el model, el que fa que s'integri molt bé amb tot els altres elements estilitzats.

Primer de tot, s'ha modelat l'herba a partir d'un pla i s'ha duplicat vàries vegades, modificant lleugerament la forma. A més, s'ha canviat la direcció de les normals dels plans (Figura 339): si es dirigeixen cap dalt no es generen ombres i contribueix a que es vegin més planes.

Figura 339. Model de l'herba

S'ha creat un material específic per aquesta herba, anomenat *MM_Grass_RVT,* tal i com es veu a la Figura 340.

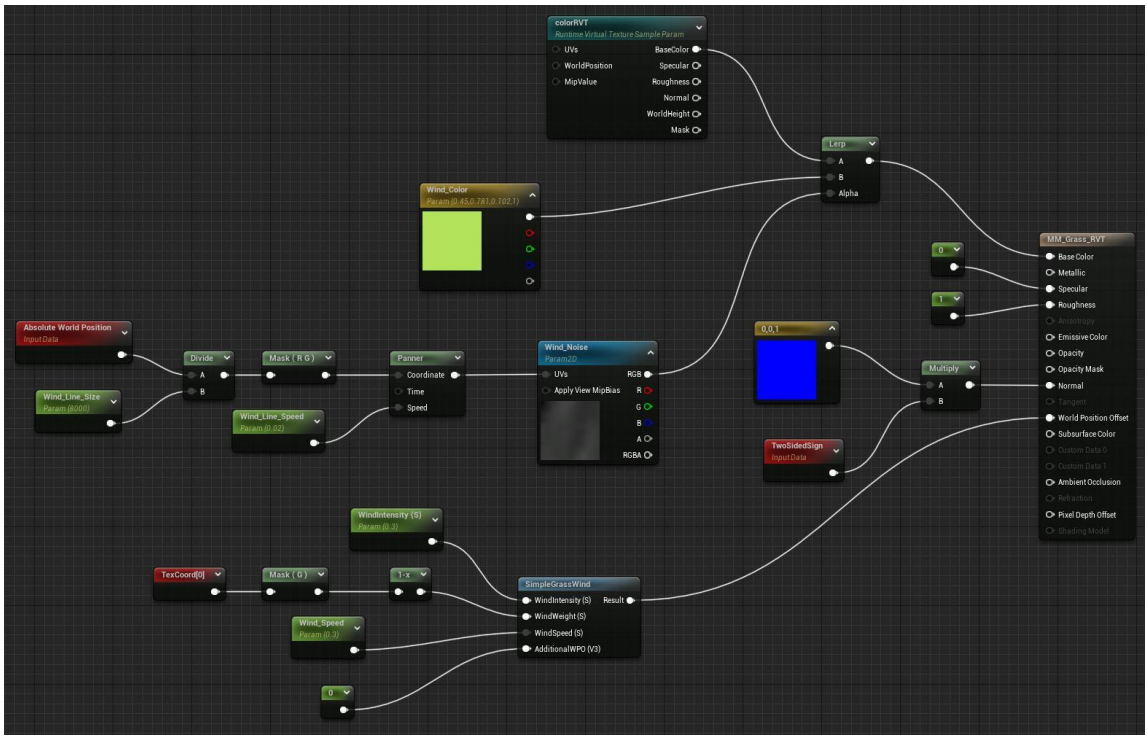

Figura 340. Visió general del material de l'herba

Per una part, es canvia el valor del paràmetre *Normal*, per a que, com es tracta d'un pla, les ombres es comportin correctament (Figura 341).

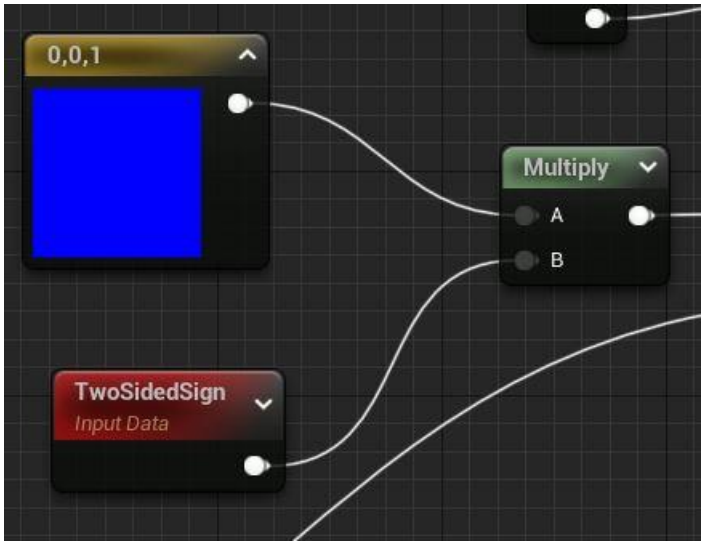

Figura 341. Normal

Per altra banda, s'utilitza el node RVT (Figura 342), al que se li assigna la textura Color, per a obtenir el valor del color, el qual es barreja amb el que se li vol aplicar al vent, que consta d'una textura que es va moment al llarg de l'escenari on hi ha vegetació i que representa les onades de vent. Aquestes onades tenen una grandària i van a una velocitat en concret que es pot modificar des de la instància del material.

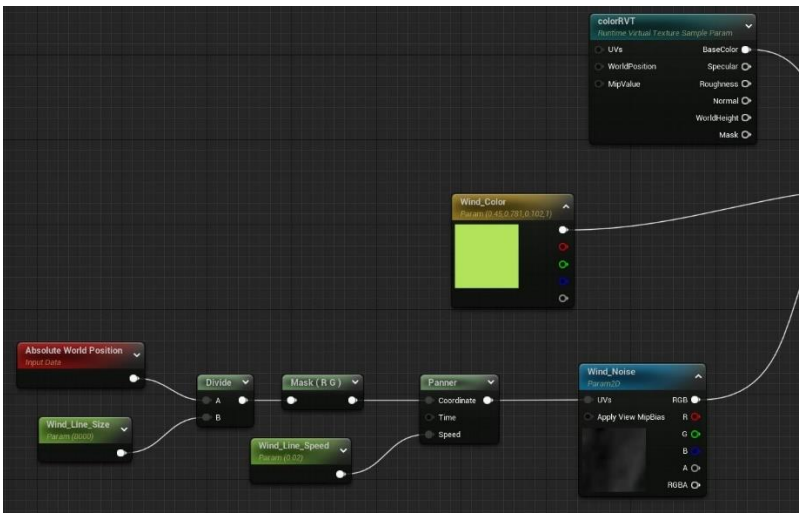

Figura 342. Material de l'herba (primera part)

Aquest vent que s'acaba de comentar és només un efecte visual, però, tal i com es pot veure a la Figura 333, també se li ha aplicat moviment, amb valors parametritzables, al model.

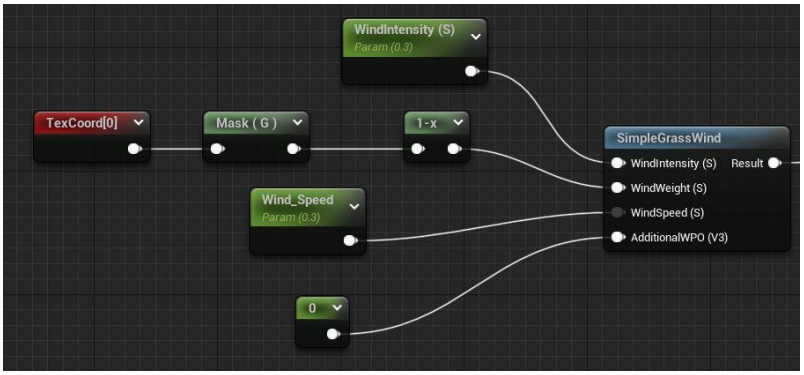

Figura 343. Material de l'herba (segona part)

En la Figura 344 es pot veure el resultat d'aplicar aquest material al model de l'herba, instanciada mitjançant l'eina *Foliage* de l'*Unreal,* i configurant els paràmetres per a aconseguir els resultats esperats (Figura 345).

Figura 344. Herba amb el material aplicat

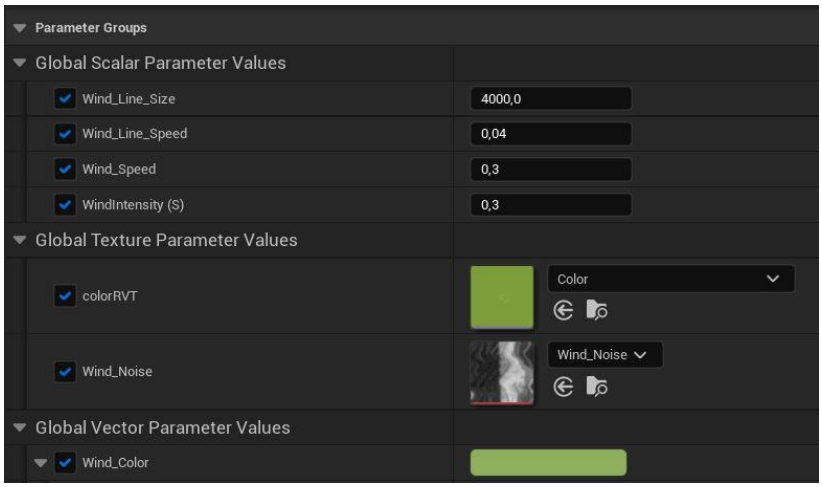

Figura 345. Paràmetres del material de l'herba

6.7.3. Blend

Per a millorar la integració dels objectes amb l'entorn, s'ha creat un material (*MF_RVT_Blend*) que barreja la textura del terreny amb la del model (Figura 346). Això s'ha implementat en un *Material Function* (Apartat 6.8.2), per a poder cridar aquesta part de codi des d'altres materials.

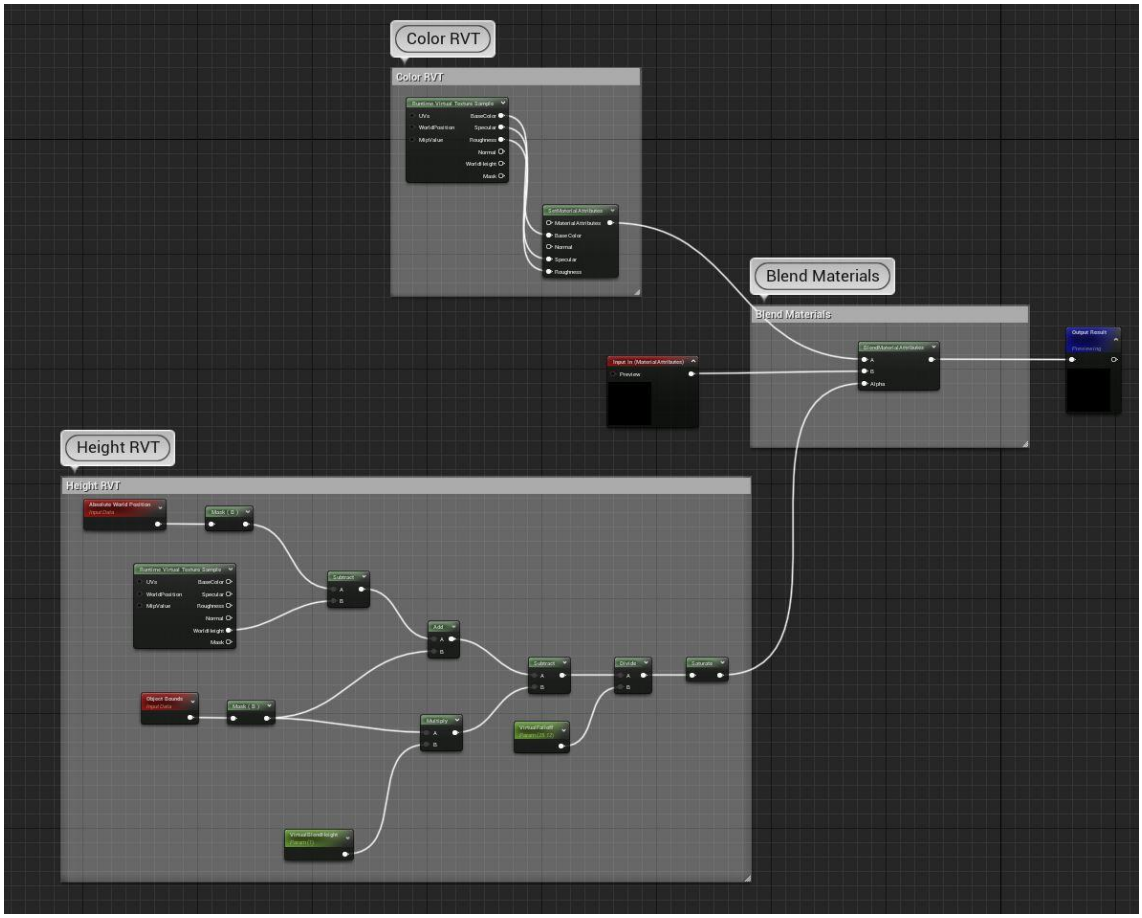

Figura 346. Material Function pel blending dels objectes amb el terreny

Per una banda, s'extreuen les característiques (*Base Color*, *Specular* i *Roughness*) de la textura *RVT* de *Color* (Figura 347) que s'assigna des del panell de detalls del material.

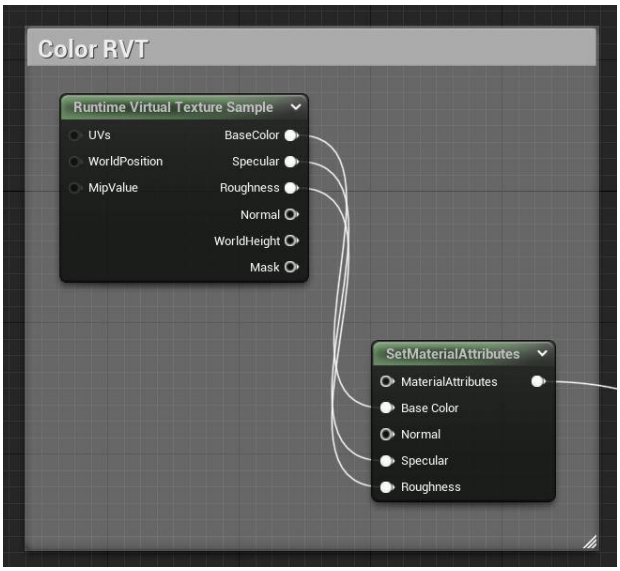

Figura 347. MF_RVT_Blend (primera part)

Seguidament, es fa el mateix però amb la textura *RVT Height* (Figura 348), la qual extreu l'altura del terreny i la forma de l'objecte per a que la textura/color s'adapti al model i el terra.

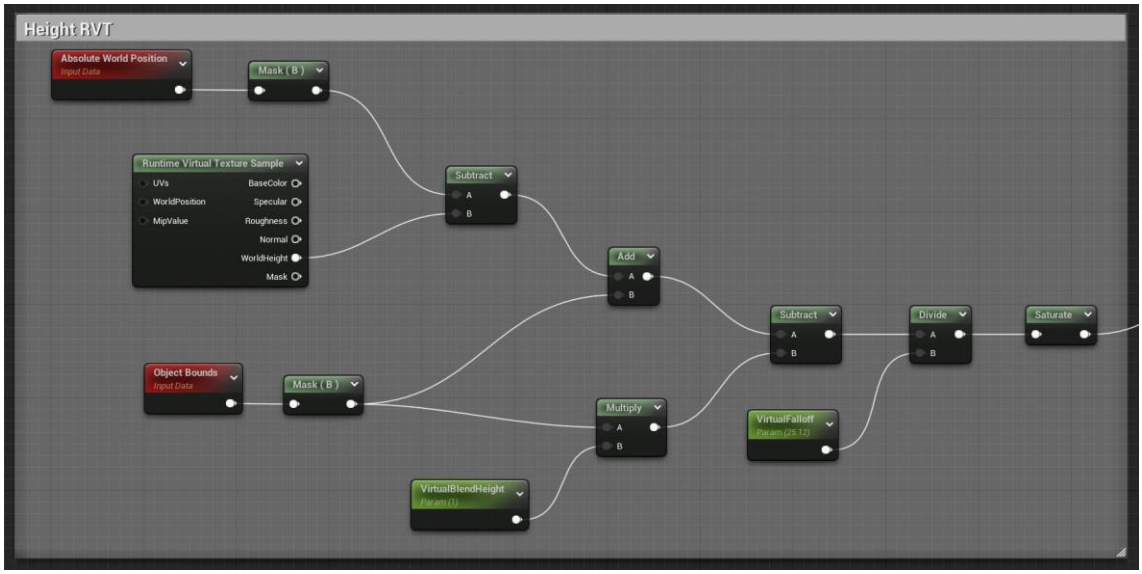

Figura 348. MF_RVT_Blend (segona part)

Per acabar, es barregen tots els atributs del material amb els de les dos textures *RVT* (Figura 349).

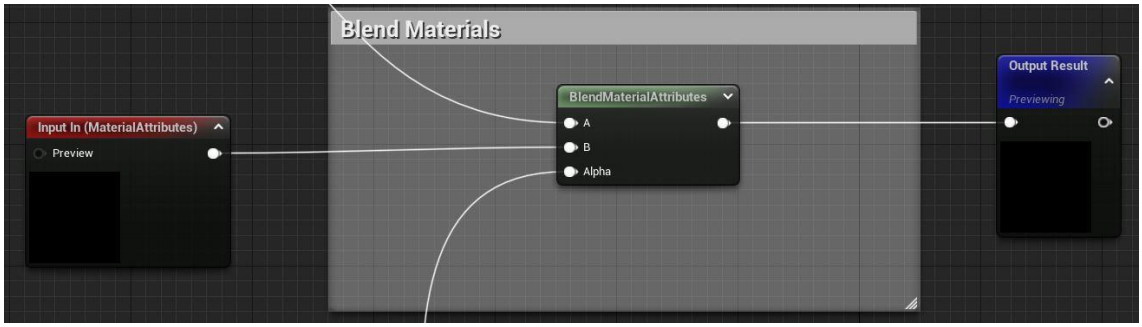

Figura 349. MF_RVT_Blend (tercera part)

En la Figura 351 es pot veure el resultat d'aplicar aquesta funció als materials dels models i canviar els valors dels diferents paràmetres (Figura 350), per a veure quins són els més adients per cada cas. Com es pot apreciar a la figura, les parts que intersecten amb el terreny agafen el color d'aquest i l'integren en el model.

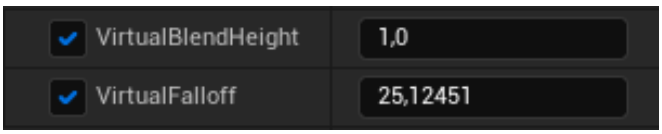

Figura 350. Paràmetres del blend

Figura 351. Resultat d'aplicar el blend als objectes

6.8. Materials

Al llarg del projecte s'han implementat diversos materials i algunes funcionalitats d'aquests que s'ha cregut oportú mencionar.

6.8.1. Material Instances

Els *Material Instances* són materials que hereten d'un pare. Crear aquests fills permet canviar els valors de les variables que s'han indicat que són parametritzables dins de la implementació del pare (Figura 352). D'aquesta manera, els canvis només afecten al fill i permeten, utilitzant la mateixa base, crear una gran varietat de materials que semblen diferents.

Un exemple d'això, és el material dels arbres i arbustos. Com es pot veure a la Figura 354, en aquest *material instance* s'ha canviat el color i textura respecte al pare (Figura 353).

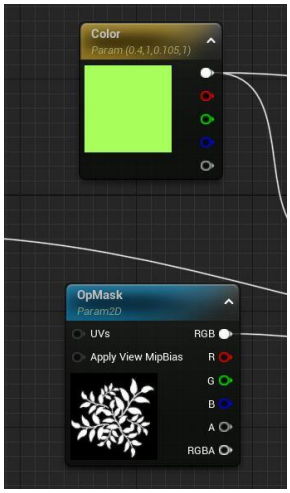

Figura 352. Exemple de variables parametritzables

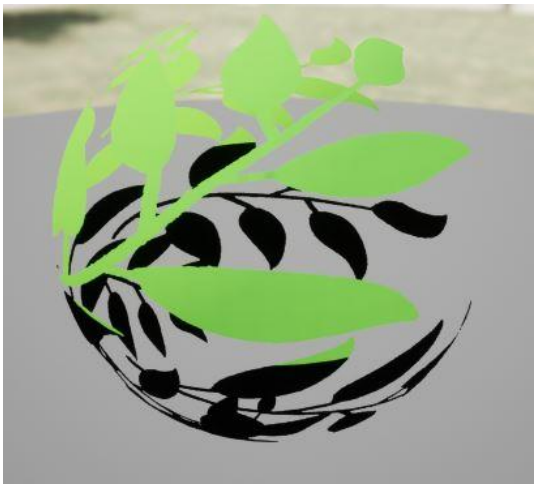

Figura 353. Previsualització del material pare

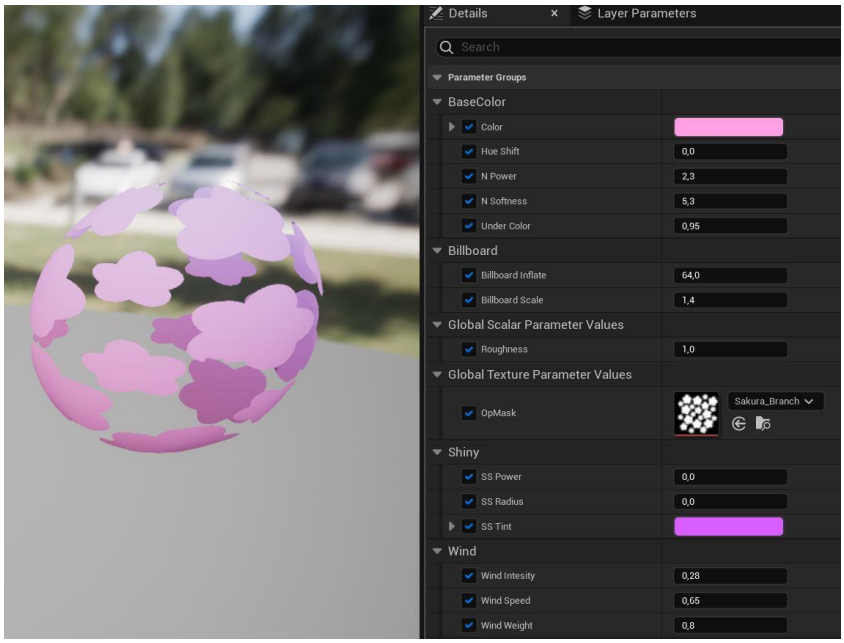

Figura 354. Previsualització i variables del material fill

6.8.2. Material Functions

Els Material Functions són materials que inclouen nodes amb paràmetres d'entrada (opcional) i de sortida. Aquest tipus de material encapsulen un tros d'implementació molt concreta que es pot utilitzar en qualsevol altre material, es fan servir com si fossin les funcions del *blueprint*.

En el joc només s'ha creat un *material function*, que ja s'ha comentat en apartats anteriors, el *MF_RVT_Blend*, el qual es crida a múltiples materials per a que les textures dels objectes es barregi amb la del terreny. Com es pot veure a la Figura 355, per utilitzar-lo correctament se li han de passar els atributs del material per l'entrada i recollir modificar de la sortida.

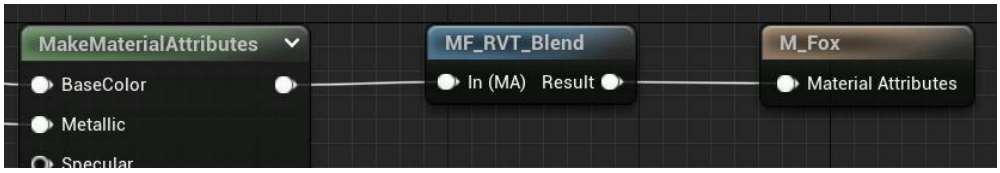

Figura 355. Crida al material function MG_RVT_Blend

6.8.3. Aigua

Per a fer l'aigua s'ha creat un material anomenat *M_Water* (Figura 356) que defineix i parametritza diferents valors i característiques que, de forma conjunta, permeten crear un efecte d'aigua, ja que aquest material s'aplica a un pla, és a dir, no genera ones que modifiquin la malla.

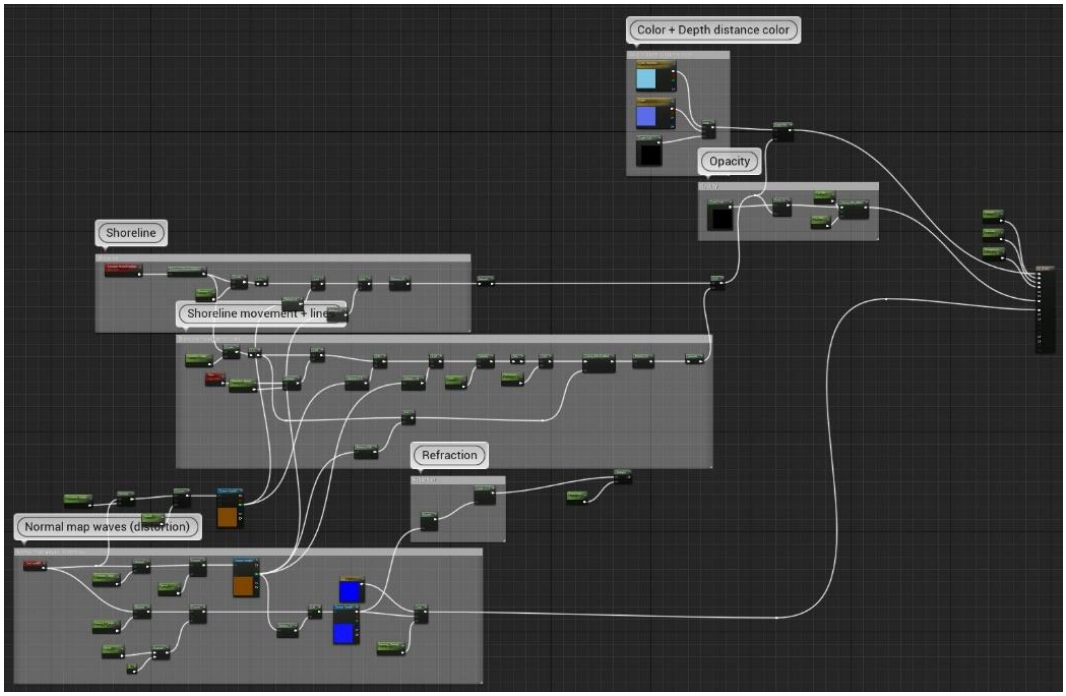

Figura 356. M_Water

Primer de tot, com es pot veure a la Figura 357, es defineixen els colors de l'aigua, tant el de la superfície com el de la part més profunda. Aquest últim es tornar més opac segons la profunditat real del terreny. A més, també es defineix la opacitat segons aquesta distància.

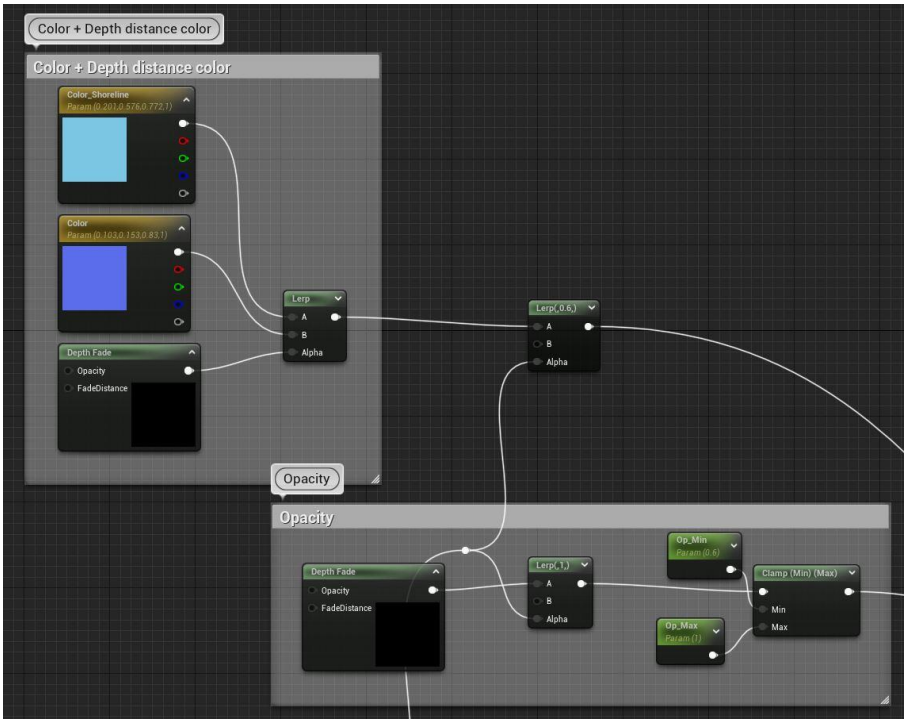

Figura 357. Color i opacitat de M_Water

Per altra part, com es mostra a la Figura 358 i Figura 359, aquest material genera efectes d'escuma en moviment a la vora de l'aigua i dels objectes que col·lisionen amb ella. Aquestes línies són totalment parametritzables, es pot escollir el gruix, la quantitat de línies al voltant dels objectes, el detall que aquestes tenen, com per exemple, si es vol més pla o amb més soroll, etc.

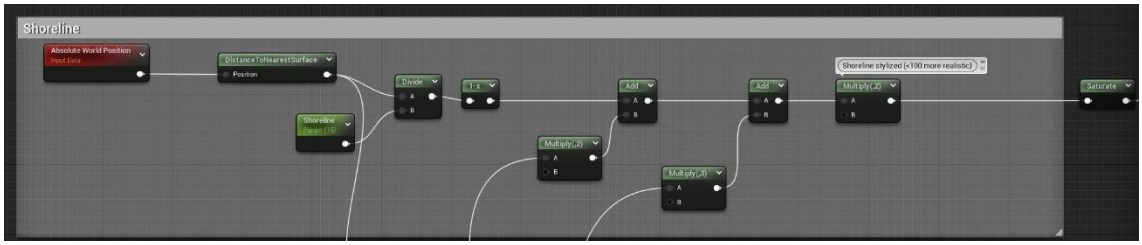

Figura 358. Línies d'escuma de M_Water (primera part)

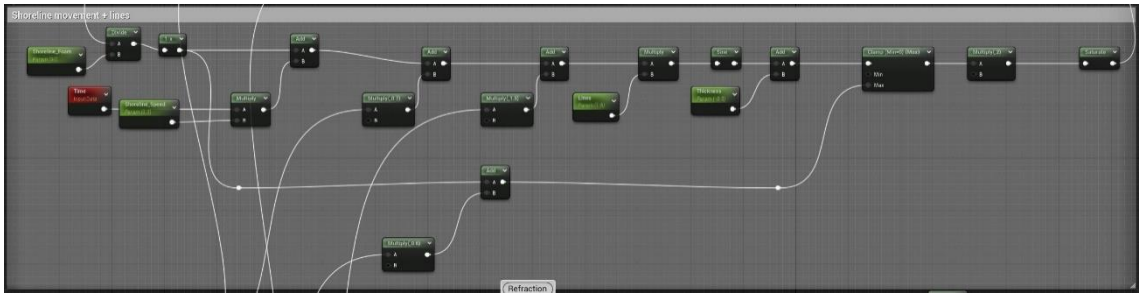

Figura 359. Línies d'escuma de M_Water (segona part)

Per altra banda, com es mostra a la Figura 360, es genera una textura de normals ue representen les ones. Aquesta textura es desplaça per tota l'àrea de l'objecte, el que dona l'efecte que el moviment és real i la malla està essent deformada, encara que no sigui així.

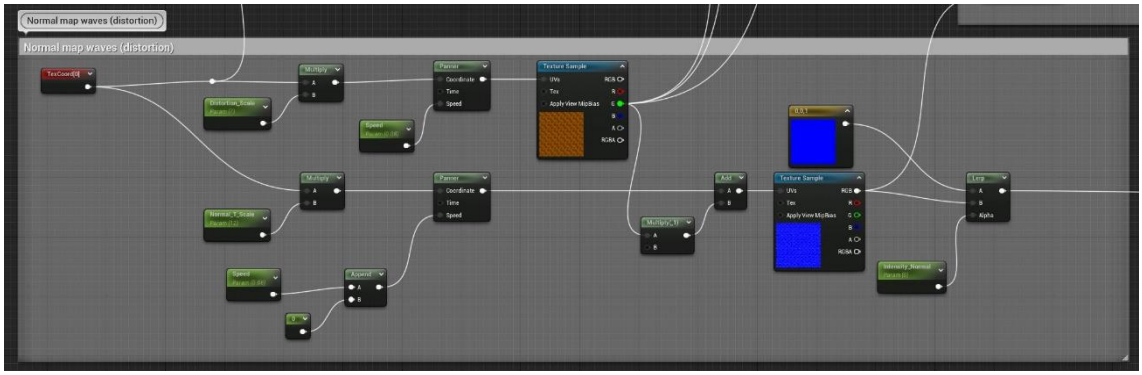

Figura 360. Ones de M_Water

En la Figura 361 es pot veure el resultat aconseguit aplicant el material a un pla i modificant els paràmetres per a aconseguir un aspecte estilitzat que encaixa amb l'entorn.

Figura 361. Resultat de M_Water aplicat a un pla

6.8.4. Molsa

La majoria dels objectes presenten molsa en les zones més fosques i superiors del model, la qual cosa ajuda a que s'integrin millor en un entorn com un bosc. Això s'ha aconseguit des del *Substance Painter*, però hi ha alguns objectes, com les roques; les quals s'han d'instanciar molts cops a l'entorn i es necessita que es puguin rotar, escalar i moure; que si es texturitzen amb la molsa, a l'hora de rotar-les no tindria sentit el lloc on es presenta aquest verdor o ni tan sols es veuria. Així que, s'ha decidit generar-la directament al material (Figura 362), on a partir de la textura de les normals, s'aplica una barreja de la textura de color importada del *Substance* amb un color pla que representa la molsa i que és capaç d'interpretar la malla de l'objecte per a situar-la sempre a la part superior, tot i que el model es roti. Si es genera un *material instance* d'aquest material, les variables parametritzables permeten decidir si es vol molsa o no i el color d'aquesta. En la Figura 363 es pot veure el resultat.

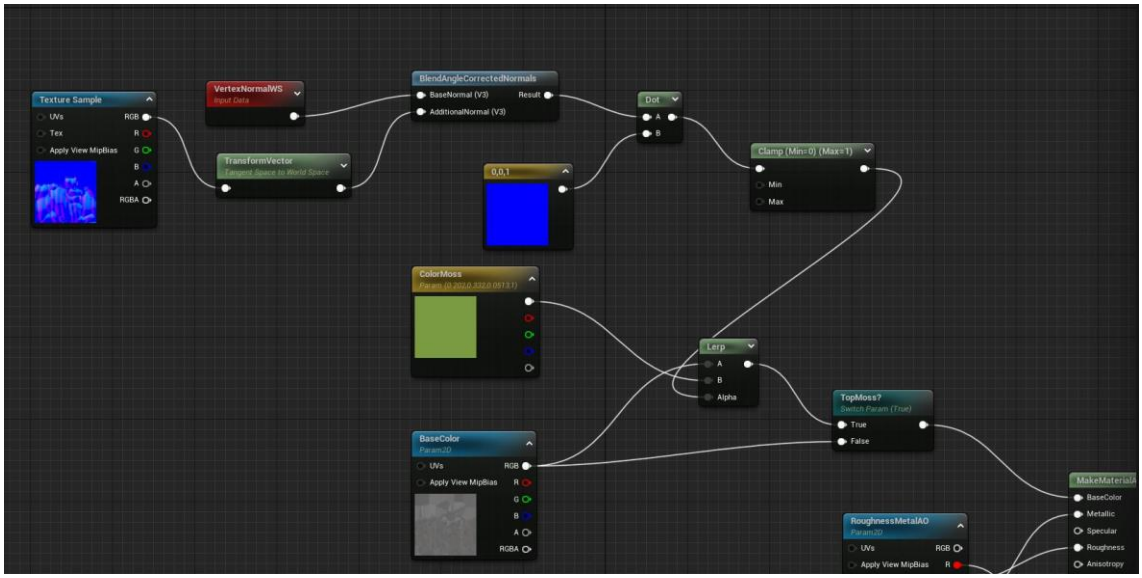

Figura 362. Implementació de la molsa en els materials

Figura 363. Roques amb molsa

6.8.5. Materials amb textures alpha

Les textures *alpha,* com s'ha comentat en apartats anteriors, són textures dissenyades en escala de grisos o, directament, en blanc i negre. Aquestes textures són molt útils per crear vegetació i efectes. Pel que fa als colors de les textures, al color negre se li aplica transparència, en canvi, totes les figures que estiguin en blanc es visible al material. A més, se li pot assignar un color o textures que s'apliquen només a aquesta part blanca, la qual cosa aporta molta versatilitat al material.

Arbres i arbustos

Els arbres i arbustos són elements fonamentals de l'entorn i, per fer-los coherents amb tots els altres models, s'ha decidit utilitzar textures *alpha* per a dissenyar les fulles. A més, posteriorment s'han creat diversos *Material Instances,* per generar més varietat

tant de colors com de textures. El material pare implementat com a base per a tots els arbres i arbustos s'anomena *M_Tree* (Figura 364 i Figura 365). Totes les variables utilitzades són parametritzables, així que els fills d'aquest material tenen moltes possibilitats de canvi.

Primer de tot, se li defineix un color base que es pot modificar amb diferents valors, d'aquesta manera se li poden aplicar altres tonalitats per a que no es vegi completament pla, si és el que es vol. També se li afegeix un color emissiu, el qual aporta brillantor a l'arbre. A més, se li aplica el moviment del vent, semblant a com s'ha fet amb l'herba (Apartat 6.7.2). Per acabar, s'arregla la direcció de les normals de l'arbre, com ja s'ha vist en altres apartats amb els materials que només tenen una cara.

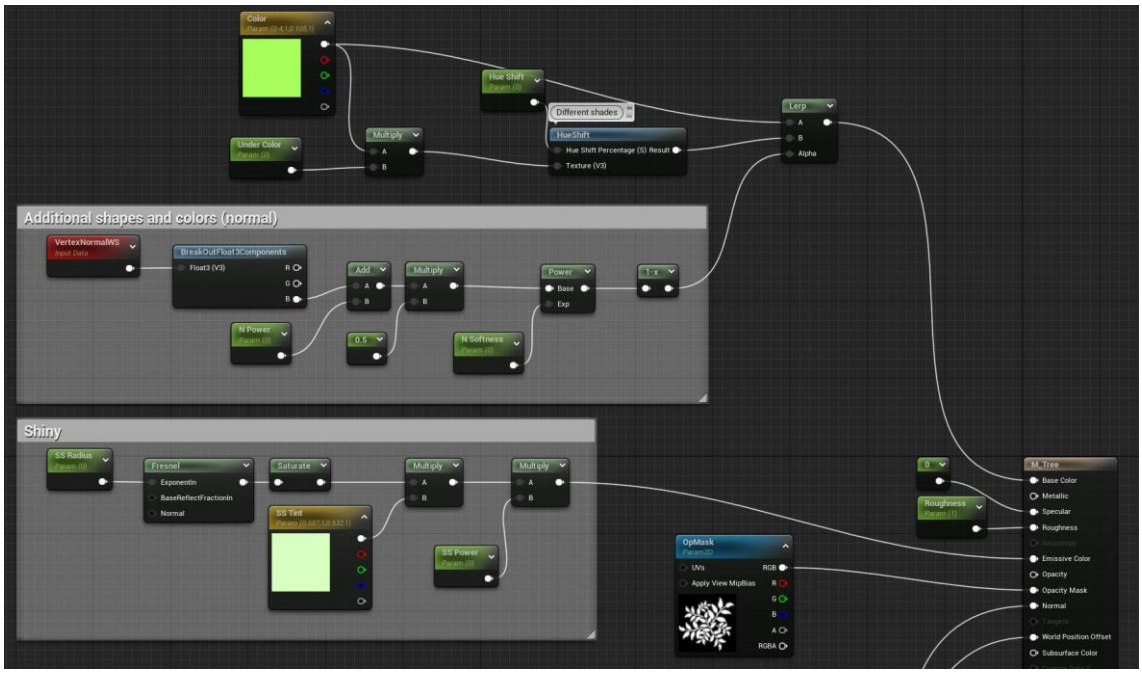

Figura 364. M_Tree (primera part)

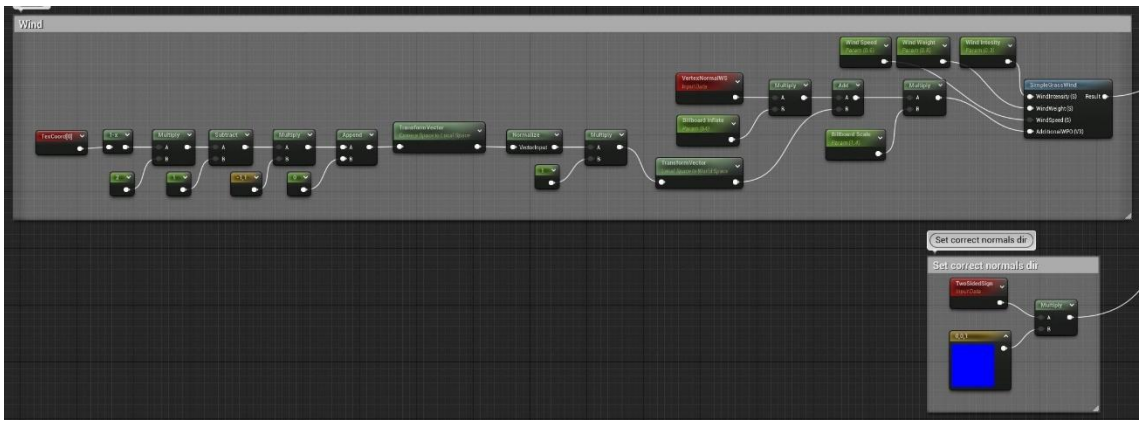

Figura 365. M_Tree (segona part)

A la Figura 366 es pot veure el resultat d'uns dels arbres després d'haver aplicat el material fil d'*M_Tree.*

Figura 366. Resultat de l'arbre amb el material M_Tree

Partícules

Alguns dels efectes creats amb sistemes de partícules parteixen d'un material generat prèviament, en gran part amb una textura *alpha.* Per exemple, el material *M_Ring* (Figura 367), s'ha utilitzat per crear l'efecte que s'afegeix al portal quan es desbloqueja, vist a la Figura 28. Molts dels paràmetres que es necessiten per a generar-ho es defineixen al material i assignen valor al propi component, en aquest cas, el *Niagara* (Figura 368).

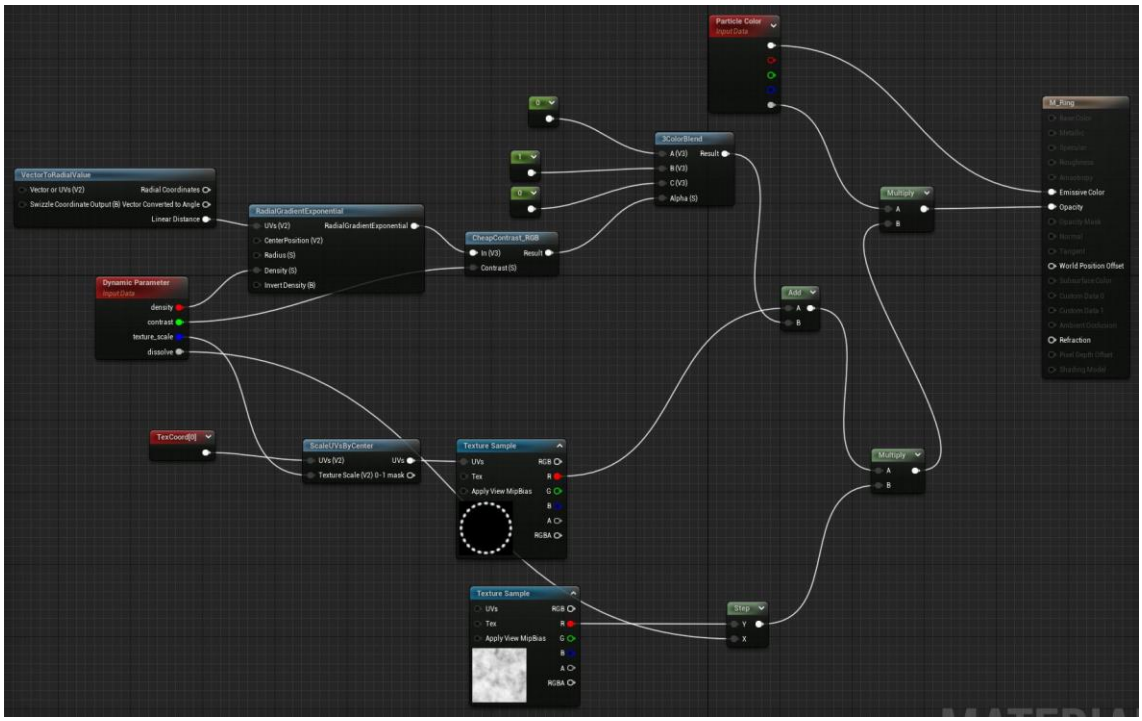

Figura 367. M_Ring

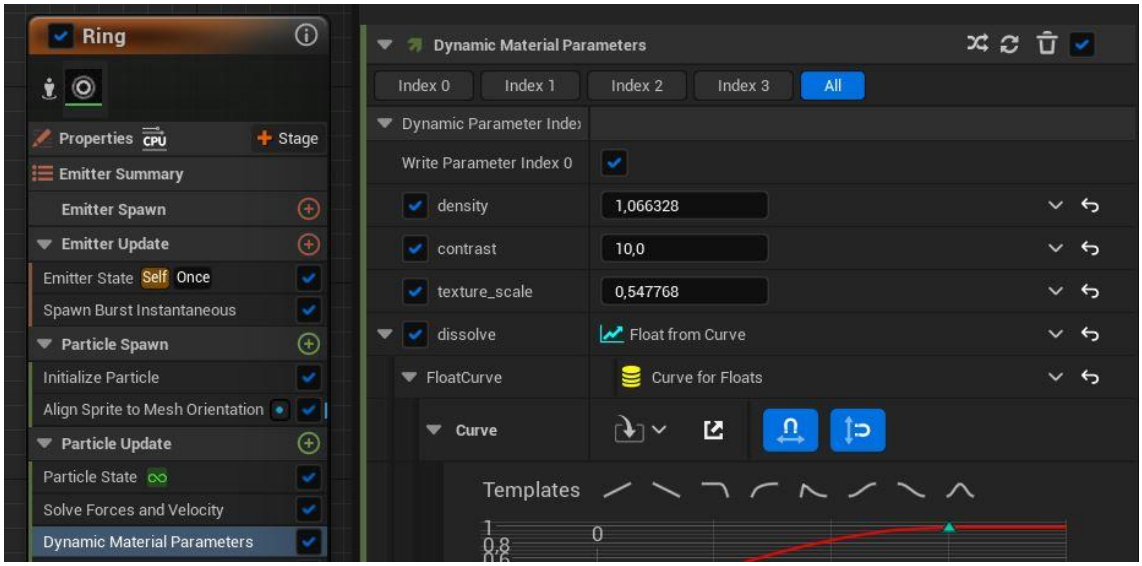

Figura 368. Paràmetres dinàmics del material M_Ring

6.9. Sons

Els sons utilitzats en el joc s'han extret, majoritàriament, de les pàgines web *freesound* i *pixabay*. Un cop descarregats i importats al motor, abans d'implementar-los als diversos *blueprints,* s'ha creat un *Sound Cue* per a cadascun d'ells o per agrupacions. El *Sound Cue* és un component de l'*Unreal* que permet modificar els paràmetres, característiques i comportaments dels sons. Per exemple, a la Figura 369 s'assigna només un so, al que se li han pogut modificar paràmetres com el volum. Per altra part, a la Figura 370 es pot veure un exemple de com tractar múltiples sons. En aquest cas, s'escull un de forma aleatòria i se li modifica el volum, també aleatòriament.

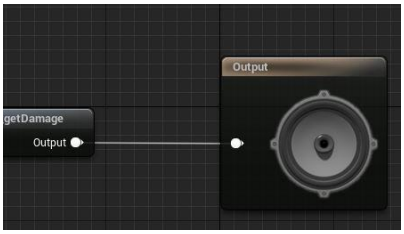

Figura 369. Sound Cue simple

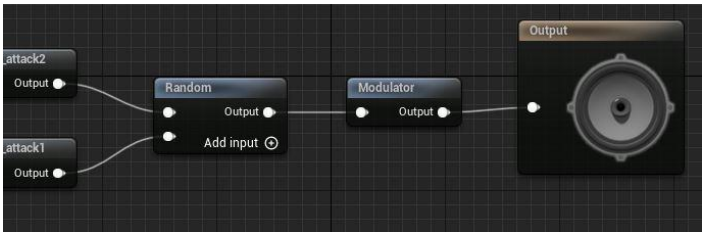

Figura 370. SoundCue amb múltiples sons

6.9.1. So bàsic

La majoria de sons que es reprodueixen al llarg del joc, ho fan en un instant en concret i no se'ls fa cap altre tractament des dels *blueprints*. Els nodes utilitzats per a reproduir-los són l'*Spawn Sound 2D* i el *Play Sound at Location*. L'*Spawn Sound 2D* (Figura 371) és un node que s'utilitza, majoritàriament, per a reproduir sons que no es troben en una localització en concret o bé, per a les interfícies d'usuari.

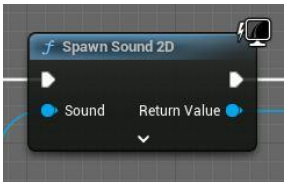

Figura 371. Node Spawn Sound 2D

El node *Play Sound at Location* (Figura 372) reprodueix un so a la posició que se li indica. Aquest so no viatja amb l'actor, així que és important que siguin curts, d'altra manera, en cas que l'objecte tingui moviment, es notaria que els dos estan en llocs diferents.

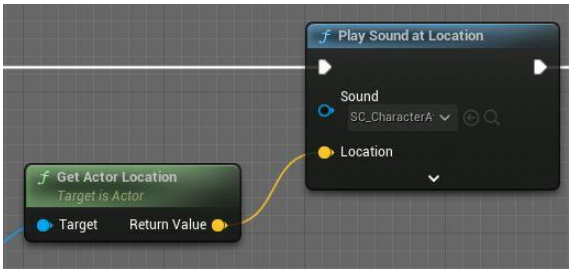

Figura 372. Node Play Sound at Location

6.9.2. Música de fons

La música de fons té un tractament diferent. Aquesta música es controla des del *blueprint CharacterController*. A l'esdeveniment *BeginPlay* (Figura 373 i Figura 374) es defineixen i guarden en variables *AudioComponent,* les dos músiques de fons (joc i menús). Els *AudioComponent* permeten reproduir, aturar i reprendre els sons. A continuació, es comprova en quin mapa es troba el jugador, i segons quin, es para un so o l'altre.

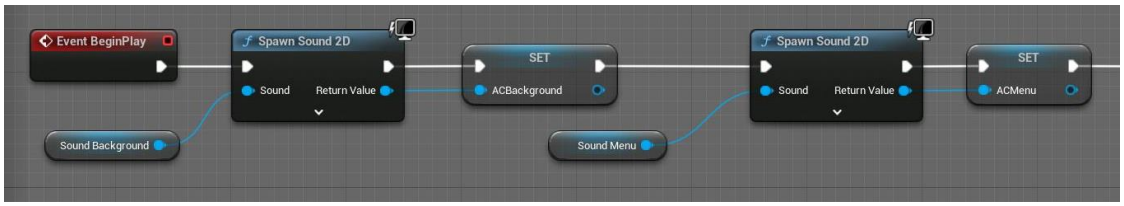

Figura 373. BeginPlay del CharacterController (primera part)

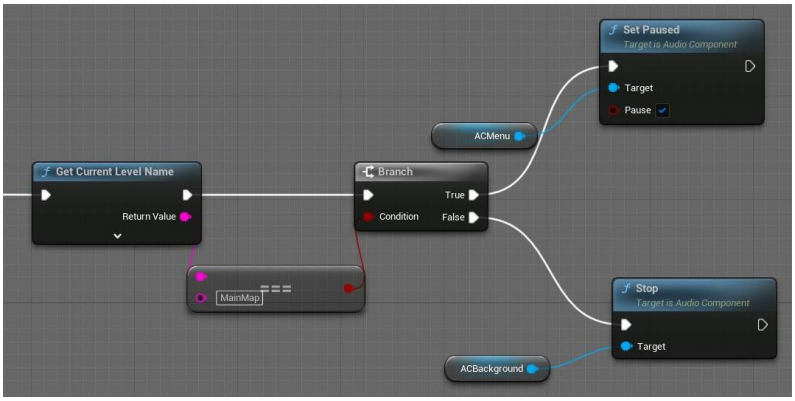

Figura 374. BeginPlay del CharacterController (segona part)

Per últim, s'ha implementat l'esdeveniment *ToggleMusic* (Figura 375). Aquest mètode s'encarrega d'aturar i reproduir la música segons el valor del paràmetre definit a *l'input,* el qual indica si el jugador està en un menú o en la partida.

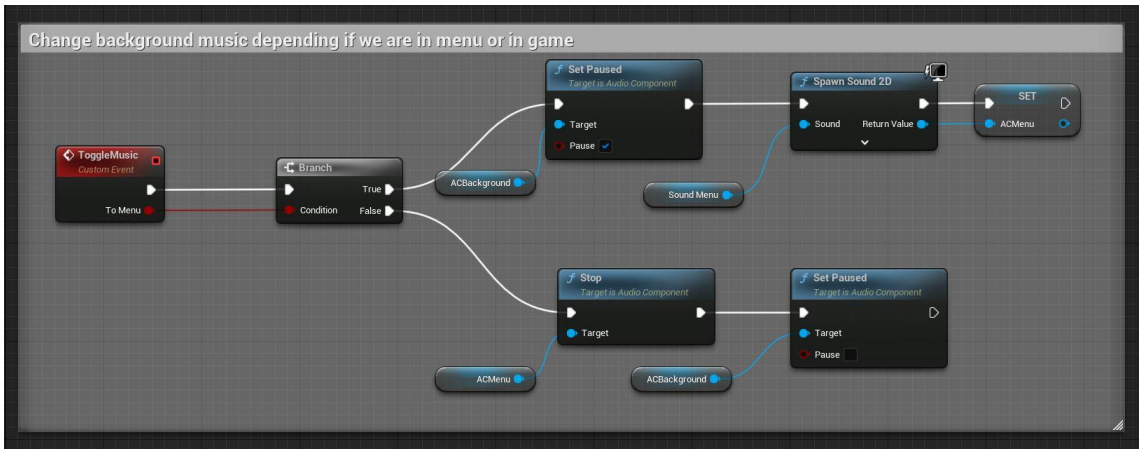

Figura 375. ToggleMusic

6.9.3. Passes

Els sons de les passes també han rebut un tractament diferent. S'ha volgut canviar el so d'aquestes segons el material del terra. Per a fer-ho s'han definit diferents *Physical Surface* a la finestra de *Project Settings* (Figura 376).

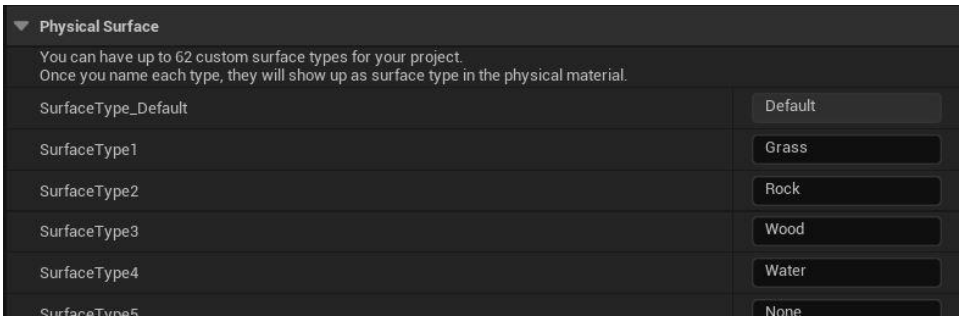

Figura 376. Physical Surfaces

Per cadascuna d'aquestes superfícies, s'ha creat un *Physical Material* (Figura 377), on se li assigna el *Physical Surface* que s'acaba de mencionar a les propietats del material, tal i com es veu a la Figura 378 amb l'exemple de l'herba.

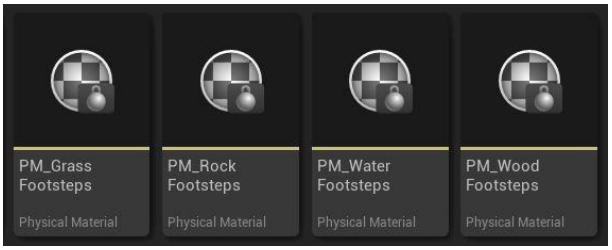

Figura 377. Physical Materials

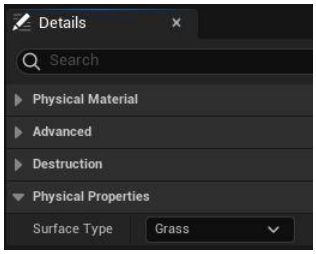

Figura 378. Propietats del Physical Material de l'herba

A continuació, a cada objecte que se li vulgui aplicar un so de passes en concret, se li ha d'assignar el *Physical Material* al material normal. Aquests materials també es poden afegir als *LayerInfo* (Figura 379), la qual cosa és molt útil per canviar fàcilment de so d'un terreny a un altre.

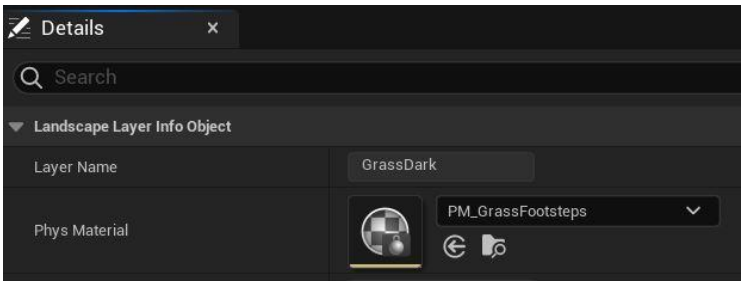

Figura 379. Physical Material assignat a un LayerInfo

Per a que els sons es reprodueixin, cal crear un *notify* a les animacions de caminar i córrer. S'han d'inserir tantes crides com passes doni l'animació. A la Figura 380 es pot veure com s'han distribuït al llarg del *timeline.*

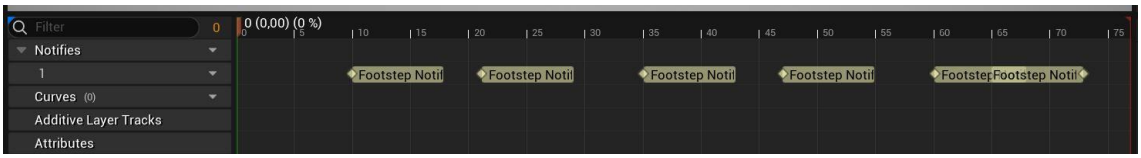

Figura 380. Timeline amb els notify per les passes

Per acabar, a l'*AnimationBlueprint* del personatge, es crea l'esdeveniment *AnimNotify_Footstep Notify* (Figura 381 i Figura 382) que es crida cada cop que hi ha un *notify* a l'animació. Aquest mètode mira el material del terra traçant una línia del personatge cap al terreny que, quan hi col·lisiona, extreu el tipus de superfície. A continuació, amb un *switch* es selecciona el tipus que coincideix i es reprodueix el *Sound Cue* de les passes. En cas que no hi hagi cap *Physical Material* assignat al terreny o l'objecte, es reprodueix el so de la branca *Default*.

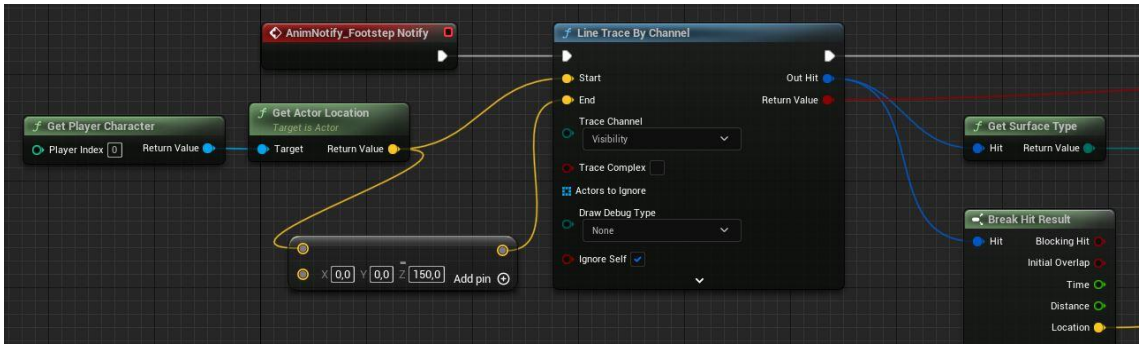

Figura 381. FootstepNotify (primera part)

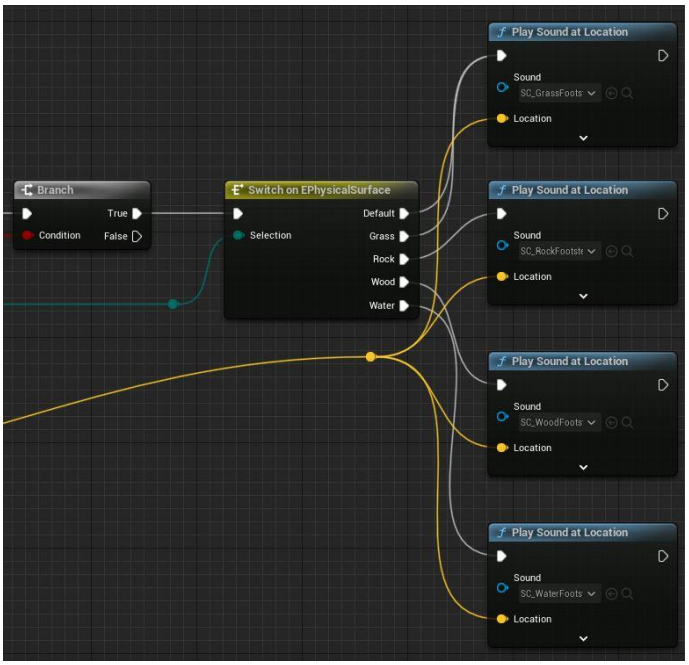

Figura 382. FootstepNotify (segona part)

7. Resultats

7.1. Legislació vigent

El joc no guarda cap tipus d'informació de caràcter personal del jugador, el que vol dir que en cap moment s'aplica la LOPD (Llei Orgànic de Protecció de Dades). A més, com aquest projecte no constitueix una activitat econòmica, tampoc s'aplica la LSSICE (Llei de Serveis de la Societat de la Informació i Comerç Electrònic).

Per altra banda, tampoc s'infringeix cap normativa de drets d'autor, ja que tots els recursos extrets de fonts externes són gratuïts i d'ús lliure.

7.2. PEGI

PEGI (*Pan European Game Information*) és un sistema de classificació europeu per edat per a videojocs. La classificació confirma si un joc és apropiat per a jugadors de certa edat. Així doncs, PEGI considera la idoneïtat de l'edat per jugar d'un joc, no el nivell de dificultat d'aquest. En la Figura 383 es poden veure les etiquetes PEGI.

Figura 383. Etiquetes PEGI

Seguint aquesta classificació, s'ha decidit etiquetar el joc com a PEGI 7, ja que conté violència suau, però podria ser capaç d'espantar als nens més petits per la presència de fantasmes i, en un futur, altres criatures.

7.3. Resultat

En les següents imatges, de la Figura 384 a la Figura 391, es mostren diferents localitzacions de l'entorn i captures de pantalla *in-game*:

Figura 384. Captura de la zona inicial

Figura 385. Captura in-game del primer grup d'enemics

Figura 386. Captura de la zona on es troba la pedra màgica

Figura 387. Captura in-game de l'estàtua curativa

Figura 388. Captura in-game del puzle bloquejat

Figura 389. Captura in-game del puzle resolt i el portal obert

Figura 390. Captura in-game de la zona final del joc

Figura 391. Captura in-game de la plataforma final

8. Conclusions

Un cop finalitzat el projecte es pot dir que s'han complert tots el objectius proposats a l'inici amb un resultat totalment satisfactori. S'ha desenvolupat un joc on s'han planificat les diferents fases del desenvolupament i dissenys del joc; s'ha fet un estudi dels elements i la cultura tradicional japonesa; s'han dissenyat, modelat i texturitzat els diferents *assets* que, posteriorment, s'han integrat a l'entorn; s'ha fet una recerca d'animacions i sons que són coherents amb els altres elements del joc; s'han creat diferents materials i sistemes de partícules; s'han dissenyat totes les interfícies del joc (*HUD*, menús i missatges); s'han implementat les mecàniques del personatge i la *IA* dels enemics, i s'ha revisat i finalitzat la documentació desenvolupada durant el projecte.

Al llarg del desenvolupament del joc, s'han pogut posar en pràctica els coneixements adquirits en els estudis. A més, s'ha estat capaç de millorar en el modelat i texturitzat 3D, s'han pogut implementar diverses mecàniques i la IA dels enemics, coses en que no s'havia tingut l'oportunitat de treballar en profunditat al grau i, també, s'han adquirit nous coneixements, com per exemple, sobre la implementació de materials, els diferents sistemes de partícules i la il·luminació.

Així doncs, es creu que, després del temps i les ganes dipositades en el joc, l'esforç empleat durant el procés de desenvolupament s'ha vist recompensat amb l'experiència i el resultat final que s'ha obtingut, el qual ha superat totes les expectatives.

9. Treball futur

Un cop acabat el projecte i havent fet múltiples proves, es creu que hi ha aspectes del joc que es podrien seguir desenvolupant i millorant. A continuació es llisten alguns d'aquests aspectes:

- Ampliar l'entorn, afegint noves zones per explora, i repartir les altres dos pedres màgiques, per a que el jugador comenci la partida havent d'anar a recuperar totes tres, i així fer un joc encara més complet.
- Millorar el moviment en lluita, com per exemple, ser capaç de moure el personatge mentre bloqueja. D'aquesta manera el combat seria més dinàmic i el jugador podria rectificar la seva posició.
- Afegir la mecànica per a fixar un enemic com a objectiu. Aquesta mecànica també facilitaria el combat i ajudaria al jugador a no fallar tants atacs.
- Implementar combinacions d'atacs i crear-ne de nous, com per exemple, un atac giratori.
- Fer les *IA*s dels enemics més complexes i diferents per a donar més varietat als combats i així no deixar que el jugador acabi d'aprendre's el patró de cada enemic.
- Crear més tipus d'enemics, com altres criatures de la mitologia japonesa. A més afegir un *boss* final li donaria més epicitat al final del joc.
- Afegir més puzles de diferents tipus.
- Implementar un sistema de guardat per a que el jugador pugui sortir de la partida sempre que vulgui i no hagi de començar de zero quan mor.
- Afegir més animacions i crear-ne de noves.
- Implementar un apartat Opcions al menú principal i de pausa.
- Canviar la llum del dia segons el jugador avança pel joc.
- Continuar investigant i aprenent sobre il·luminació per a poder crear una ambientació més professional.

10. Bibliografia

1. Artell. (2023). *Welcome to the Auto-Rig Pro documentation!* AutoRigPro Doc. http://lucky3d.fr/auto-rig-pro/doc/index.html

2. Basic UE4 Developer. (5 de agost de 2022). *Advanced Combat System Tutorial - Line Trace/Dealing Damage [UE4]*. [Vídeo]. https://www.youtube.com/watch?v=Qjul1dAKChs

3. Ben Cloward. (26 de desembre de 2019). *Environment Blending - UE4 Materials 101 - Episode 6*. [Vídeo]. https://www.youtube.com/watch?v=QwVDlS8uiYg

4. Britez, D. (30 de gener de 2022). *Kitsune, el zorro de la mitología japonesa*. Mirando hacia Japón. https://mirandohaciajapon.com/kitsune-zorro-japones/

5. Catalunya, Institut Obert de. (Setembre de 2018). *Desenvolupament d'entorns interactius multidispositiu - Disseny de nivells.* Institut Obert de Catalunya. https://ioc.xtec.cat/materials/FP/Recursos/fp_a3d_m07_/web/fp_a3d_m07_htmlindex/WebContent/u3/a3/continguts.html

6. Cghow. (17 de maig de 2022). *Magical Portal in UE5 Niagara Tutorial | Download Files*. [Vídeo]. https://www.youtube.com/watch?v=5v7Xk0Na_bQ

7. Crecente, B. (19 de octubre de 2020). *The magic of creating Kena: Bridge of Spirits.* Unreal Engine. https://www.unrealengine.com/en-US/developer-interviews/the-magic-of-creating-kena-bridge-of-spirits

8. Dean Ashford. (9 de abril de 2017). *UE4 -Tutorial - Falling Leaves Particles! (Request)*. [Vídeo]. https://www.youtube.com/watch?v=aJe1ysBGbXs

9. Dean Ashford. (4 de juliol de 2020). *UE4 - Tutorial - Stylised Wind Particles (Cascade)*. [Vídeo]. https://www.youtube.com/watch?v=6HHHAcaiQkI

10. Epic Game, Inc. (sense data). *Unreal Engine Forum*. https://forums.unrealengine.com/categories?tag=unreal-engine

11. Epic Games, Inc. (sense data). *Unreal Engine 5 Documentation*. https://docs.unrealengine.com/5.0/en-US/

12. Fischer, A. (24 de abril de 2021). *¿Qué significan los arcos torii, las puertas sagradas que se encuentran en todo Japón?* National Geographic. https://www.ngenespanol.com/el-mundo/que-simbolizan-los-arcos-torii-las-puertas-sagradas-que-se-encuentran-en-todo-japon/

13. Go Tokyo. (27 de desembre de 2022). *Todo lo que necesitas sobre Sakura – los cerezos en flor japoneses* . Go Tokyo. https://www.gotokyo.org/es/story/guide/the-japanese-cherry-blossom-trees/index.html

14. Gonçalo Marques. (2 de juliol de 2023). *UI Navigation Tutorials*. [Vídeo]. https://www.youtube.com/playlist?list=PLAcWSem_HT4h8759U8Lbh7VfjdNGhKq3v

15. Gorka Games. (15 de setembre de 2022). *How to Make a 3D Interaction Prompt in Unreal Engine 5*. [Vídeo]. https://www.youtube.com/watch?v=VU7jdKPdjLw

16. Institut Obert de Catalunya. (setembre de 2018). *Desenvolupament d'entorns interactius multidispositiu - Disseny de nivells.* Institut Obert de Catalunya. https://ioc.xtec.cat/materials/FP/Recursos/fp_a3d_m07_/web/fp_a3d_m07_htmlindex/WebContent/u3/a3/continguts.html

17. It's Me Bro. (8 de agost de 2022). *UE5 Pressure Plate Puzzle*. [Vídeo]. https://www.youtube.com/watch?v=HFarVFT6KgA&list=PLNJ8lQUDngfuM8vC2mQq52uJ597mJJBGf&index=111

18. Javad Rajabzade. (15 de juny de 2022). *Texturing & Rendering Stylized Crystal In Substance 3d Painter and marmoset toolbag 4*. [Vídeo]. https://www.youtube.com/watch?v=Xjbp-ahP14M

19. Laura. (9 de juny de 2023). *Chōchin, los farolillos de papel japoneses*. Japonismo. https://japonismo.com/blog/chochin-farolillos-papel-japoneses

20. MadPonyInteractive. (10 de desembre de 2019). *UE4 Character Cloth Simulation*. [Vídeo]. https://www.youtube.com/watch?v=X5acomKAux4

21. Matt Aspland. (18 de novembre de 2022). *How To Interact In Unreal Engine 5 | How To Use Blueprint Interfaces In Unreal Engine 5 (Tutorial)*. [Vídeo]. https://www.youtube.com/watch?v=5-UJT4U-jeg

22. Matt Aspland. (23 de setembre de 2023). *Dynamic Footstep System | Different Sounds On Different Surfaces - Unreal Engine 4 Tutorial*. [Vídeo]. https://www.youtube.com/watch?v=1TTey8wUUvU

23. Pèrez, S. (2019). *Level Design: Resources and Techniques to Guide the Player.* [Treball de fi de Grau, Universitat Politècnica de Catalunya]. Repositori Digital de la UPC. http://hdl.handle.net/2117/173957

24. Quiskeya Studio. (30 de agost de 2022). *UE5 Simple Stylized Fire FX Tutorial | In 10 Minutes!* [Vídeo]. https://www.youtube.com/watch?v=utxuj5dsKCI

25. Rimaye. (15 de març de 2021). *Unreal Engine : How to create Stylized Water Material - UE4 tutorial [Distance fields - reflection]*. [Vídeo]. https://www.youtube.com/watch?v=Ix7K_IubTig

26. Rodríguez, P. (2021). *Tipos de máscaras japonesas y su significado*. Japón alternativo. https://www.japonalternativo.com/blog/tradiciones-japonesas/tipos-de-mascaras-japonesas/

27. Rosie Jarvis. (4 de maig de 2022). *Fireflies in UE5 - under 2 minute tutorial*. [Vídeo]. https://www.youtube.com/watch?v=6-yPNPd1s14

28. Ryan Laley. (16 de agost de 2022). *Learn all About AI in Unreal Engine 5*. [Vídeo]. https://www.youtube.com/playlist?list=PL4G2bSPE_8uklDwraUCMKHRk2ZiW29R6e

29. Ryan Laley. (2022 de abril de 7). *Unreal Engine 4 Tutorial - Niagara VFX: Sparkles*. [Vídeo]. https://www.youtube.com/watch?v=eQvdmlFTvvs

30. Sarkamari. (3 de juny de 2023). *UE5 Series: Mastering Emissive Materials in Unreal Engine*. [Vídeo]. https://www.youtube.com/watch?v=jUrRklZRNrU

31. Uisco. (16 de febrer de 2022). *How To Make Melee Combat System In Unreal Engine 5*. [Vídeo]. https://www.youtube.com/watch?v=iPfU1SmzkkY

32. Universitat de Girona. (juny de 2023). *Estil APA*. Universitat de Girona. https://biblioteca.udg.edu/ca/com-citar-documents/estil-apa?language_content_entity=ca

33. Unreal Engine. (28 de maig de 2014). *Blueprint Essentials: Enum Variables | 05 | v4.2 Tutorial Series | Unreal Engine*. [Vídeo]. https://www.youtube.com/watch?v=5nLqEW6YXPY

34. Unreal Sensei. (23 de abril de 2021). *How to Blend Objects with Your Landscape - UE4 Runtime Virtual Texturing (RVT) Tutorial*. [Vídeo]. https://www.youtube.com/watch?v=xYuIDFzKaF4

35. Victoria Zavhorodnia. (2 de novembre de 2022). *Stylized Flowers Tutorial*. [Vídeo]. https://www.youtube.com/watch?v=5PovTfBD29I

36. Victoria Zavhorodnia. (26 de juliol de 2022). *Stylized Fluffy Trees Tutorial*. [Vídeo]. https://www.youtube.com/watch?v=1BfZSfX6_Go

37. Victoria Zavhorodnia. (26 de octubre de 2022). *Victoria Zavhorodnia*. [Vídeo]. https://www.youtube.com/watch?v=AEMe-kcZBLw

38. Wikipedia. (11 de maig de 2022). *Kitsune*. Wikipedia. https://es.wikipedia.org/wiki/Kitsune

39. Zenn, J. (21 de novembre de 2017). *Understanding Your Audience – Bartle Player Taxonomy*. GameAnalytics. https://gameanalytics.com/blog/understanding-your-audience-bartle-player-taxonomy/

40. Phil Stoltz. (agost de 2022). *How to Texture Stylized Rocks in Substance Painter*. [Vídeo]. https://www.artstation.com/marketplace/p/oVp1l/how-to-texture-stylized-rocks-in-substance-painter

11. Annexos

11.1. Projecte

Com a Annex a aquest treball, a l'arxiu comprimit *TFG_ElenaTera.zip* s'ha adjuntat un document de text amb un enllaç al *Drive.* Dins de la carpeta del *Drive* s'ha inclòs el projecte sencer d'*Unreal Engine*, on es poden trobar tots els *blueprints*, materials, sistemes de partícules, textures i models creats; l'executable, i un *gameplay* del joc.

11.2. Recursos utilitzats

11.2.1. Sons

https://freesound.org/people/Sheyvan/sounds/519006/

https://freesound.org/people/EminYILDIRIM/sounds/563662/

https://freesound.org/people/LorenzoTheGreat/sounds/417794/

https://freesound.org/people/JiggleSticks/sounds/634986/

https://freesound.org/people/kretopi/sounds/406741/

https://freesound.org/people/duckduckpony/sounds/204017/

https://freesound.org/people/Samulis/sounds/197781/

https://freesound.org/search/?q=222655

https://pixabay.com/es/music/ambiente-180126-ambient-morning-chill-with-birds-21348/

https://freesound.org/people/miiumiiu/sounds/550889/

https://freesound.org/people/Hawkeye_Sprout/sounds/469066/

https://freesound.org/people/EminYILDIRIM/sounds/634365/

https://freesound.org/people/newlocknew/sounds/582486/

https://freesound.org/people/Luzanne0/sounds/445328/

https://freesound.org/people/MortisBlack/sounds/385046/

https://freesound.org/people/DneproMan/sounds/377223/

https://freesound.org/people/AmeAngelofSin/sounds/264982/

https://freesound.org/people/Nomfundo_k/sounds/408548/

https://freesound.org/people/Wdfourtee/sounds/192052/

https://pixabay.com/es/sound-effects/084373-heal-36672/

11.2.2. Addons

https://www.unrealengine.com/marketplace/en-US/product/ui-navigation-3

11.2.3. Animacions

https://www.mixamo.com/#/?page=1&type=Motion%2CMotionPack

11.2.4. Pinzells

https://www.brittneymurphydesign.com/downloads/free-ink-brushes-for-photoshop/

https://myphotoshopbrushes.com/brushes/id/3792/

11.2.5. Textures

https://github.com/GDi4K/unreal-openland

11.2.6. Tipografia dels menús

https://www.dafont.com/es/hiro-misake.font

12. Manual d'usuari i d'instal·lació

12.1. Manual d'usuari

En la següent taula es poden trobar els controls del joc, tant per teclat i ratolí com pel comandament d'Xbox:

	Teclat i ratolí	Comandament
Moviment	W-A-S-D	*Joystick* esquerre
Córrer	*Shift*	*Joystick* esquerre
Interactuar	E	B
Saltar	Barra espaiadora	A
Atacar	Clic esquerre del ratolí	X
Bloquejar	Clic dret del ratolí	Y
Càmera	Ratolí	*Joystick* dret
Pausa	*Escape*	Botó del menú

12.2. Instal·lació

El joc s'ha pujat al següent enllaç, on es pot descarregar l'executable i, a més, veure un vídeo del joc: https://elenatera.itch.io/hana

Per iniciar-lo, un cop descarregat, s'ha de descomprimir l'arxiu *.zip* i fer doble clic a l'executable anomenat *Hana.exe*.