Universitat de Girona
Escola Politècnica Superior

Grau en Disseny i Desenvolupament de Videojocs

PROJECTE FINAL DE GRAU

Creació d'un videojoc de fantasía i estratègia per torns amb Unreal Engine

Autor:
Roger Badenas Hernandez

Tutors:
Gustavo Patow

MEMÒRIA

Convocatòria:
Juny 2023

Departament:
Informàtica, Matemàtica Aplicada i Estadística

Índex

1. Introducció, motivacions, propòsit, objectius del projecte i distribució de tasques

1.1. Introducció

En l'actualitat, un dels pilars principals dels videojocs es tracta del gènere d'estartègia. Aquest gènere està vinculat als jocs de taula d'estratègia, ja que requereix que el jugador posi en pràctica habilitats de planejament de combat i gestió de recursos.

Dins d'aquest conjunt, trobem diferents subgèneres. Cadascun d'aquests ofereix una gran variació a la jugabilitat. Es separen en 4 diferents, l'estratègia en temps real o RTS, el qual, com el nom indica, són videojocs a on no hi ha un sistema per torns sinó que l'acció transcorre de forma contínua en el temps; la tàctica a temps real o RTT, és similar al RTS però excloent el control econòmic i de recursos i la construcció d'edificis dins de les batalles; l'estratègia per torns o TBS, aquest subgènere consisteix en el fet que la batalla disposa de torns, i abans d'actuar, el jugador disposa d'un període de temps d'anàlisis i planificació; i per acabar tenim la tàctica per torns o TBT, el qual es caracteritza per l'expectativa que el jugador completi el nivell només amb les utilitzant les forces de combat donades a l'inici de la partida.

Tenint en compte els 4 subgèneres dins dels videojocs d'estratègia, vaig veure que el que més s'apropava al videojoc que volia crear era la tàctica per torns.

Després d'investigar els principals pilars d'aquest gènere, ens vam trobar en el fet que el nostre projecte no es diferenciava gaire d'altres com XCOM, així que vaig decidir innovar en el sistema de combat afegint un sistema de màgia, agregant una capa de dificultat al joc.

A causa d'això, en veure que no era gens comú afegir-li un sistema de màgia a aquest gènere, vaig decidir-me continuar amb el projecte i aconseguir dissenyar i desenvolupar un videojoc basant-nos en el gènere de tàctica per torns a on el jugador hagi de lluitar amb màgia i no amb armes convencionals.

1.2. Motivacions

Les meves motivacions principals per crear aquest projecte han estat tres. La primera ha sigut crear un joc d'estratègia basant-nos enun dels jocs més coneguts del sector, la saga de XCOM. La segona ha estat crear un joc amb una Intel·ligència Artificial que pogués donar-li un repte al jugador, i la tercera, ha estat crear un joc de tàctica per torns amb un sistema de màgia en comptes d'armes.

1.3. Propòsit i objectius del projecte

El propòsit del projecte ha estat el de crear el prototip d'un videojoc que demostrés el futur que pot tenir un joc de tàctica per torns amb un sistema de màgia en el combat.

Els objectius a l'hora de fer aquest projecte han estat els següents:

- Demostrar la viabilitat que pot tenir un videojoc de tàctica per torns amb un sistema de màgia.
- Aprendre i millorar el meu nivell de programació amb Unreal Engine i el seu llenguatge visual, els Blueprints.
- Guanyar experiència en la creació d'intel·ligència artificial amb Unreal Engine.
- Crear un sistema de combat basat en la màgia que agregés certa dificultat i una mecànica addicional.
- Guanyar experiència i aprendre l'ús de les estructures de dades dins d'Unreal Engine.

1.4. Distribució de tasques

Donat el meu perfil tècnic, les distribucions de les tasques del projecte han estat les següents:

Estètica	15%
Narrativa	0%
Mecàniques	55%
Tecnología	30%

El percentatge referent a l'estètica es basa en la implementació de diferents continguts obtinguts de manera gratuïta dins de l'Unreal Engine Marketplace i en la cerca, modificació i implementació dels menús i HUDs.

La narrativa he decidit ignorar-la, ja que, des del principi, no he trobat una necessitat per crear-li un rerefons al projecte.

Respecte a les mecàniques i a la tecnologia, el joc disposa d'una gran varietat de personalització possible, des de crear un equip propi a partir d'un llistat de jugadors, la modificació de les habilitats de cada jugador de l'equip, fins al sistema de cobertures pertinents al gènere i la intel·ligència artificial, la qual és l'eix del videojoc. A causa d'això, he augmentat la dedicació a aquestes tasques.

2. Estudi de viabilitat

Prèviament al començament del disseny i desenvolupament del projecte, cal analitzar la viabilitat i assegurar-nos que la implementació es pot portar a terme sense grans problemes. Per fer això, caldrà estudiar produccions similars al mercat actual, definir el perfil del jugador objectiu i un model de negoci efectiu i comprovar la disponibilitat de les eines necessàries per al desenvolupament.

2.1. Viabilitat tecnològica

Un dels requisits fonamentals per al desenvolupament d'un videojoc és la disponibilitat d'un hardware que ofereixi un rendiment acceptable en utilitzar el programari escollit pel desenvolupament.

2.1.1. Hardware

El hardware utilitzat ha estat el següent:

- Ordinador
    - CPU: AMD Ryzen 5 5600X
    - Ram: 16Gb
    - GPU: NVIDIA GeForce RTX 3060 TI

- Perifèrics
    - Teclat
    - Ratolí

2.1.2. Software

El software utilitzat ha estat el següent:

- Motor de joc: Unreal Engine 4.22.3
- Editor d'imatges: Adobe Photoshop

El motor utilitzat és d'ús públic però l'editor d'imatges no. Ja que jo disposava d'una llicència, he utilitzat aquest programa, però en cas de no tenir-ne, hi ha un programa similar que es diu GIMP i és gratuït.

2.2. Recursos humans

Donat el gènere i objectius del projecte entre mans, el desenvolupament del joc en la seva totalitat requeriria, desitjablement, dels següents rols:

- Dissenyador de nivells
- Artista 3D
- Dissenyador de so
- Dissenyador UI/UX
- Animador
- Artista VFX
- Programador de IA
- Programador de Gameplay
- Programador UI/UX
- Tester

No obstant això, el desenvolupament del prototipus es realitzarà per una sola persona, que realitzarà les tasques adients de cada rol.

2.3. Viabilitat econòmica

Com s'ha comentat anteriorment, el prototipus del projecte es desenvoluparà en la seva totalitat per una sola persona sense cap cost. A la següent taula podem veure el cost aproximat dels recursos tecnològics.

Recursos tecnologics	**Cost (anual)**
Hardware	1500€
Software	290,17€ (24,19€/mes)
Total	**1790,17€**

Per una altra banda, el cost dels recursos humans es podria estimar en cas de desenvolupar el joc en la seva totalitat amb l'equip adequat:

- Dissenyador de nivells(16€/h)
- Artista 3D(15€/h)
- Dissenyador de so(12€/h)
- Dissenyador UI/UX(16€/h)
- Animador(15€/h)
- Artista VFX(15€/h)
- Programador de IA(18€/h)
- Programador de Gameplay(13€/h)
- Programador UI/UX(13€/h)
- Tester(8€/h)

Aquestes estimacions s'han realitzat utilitzant com a font els resultats de payscale.com aplicats a Espanya. En qualsevol cas, a causa que la totalitat del projecte ha estat duta a terme per mi amb el propòsit de crear el prototipus, el cost dels recursos humans és de 0€.

2.4. Públic objectiu i perfil del jugador

El perfil de jugador objectiu són adults i joves a partir de 16 anys, que busquin una experiència de joc desafiadora. Aquest tipus de joc pot ser atractiu per als jugadors que gaudeixen resoldre situacions tàctiques i estratègiques i per jugadors que gaudeixin de jocs de fantasia i màgia i busquin nous reptes.

A part, el fet de no tenir narrativa aconsegueix arribar a un tipus de jugador més casual, que jugui mes per diversió que per endinsar-se a una història.

Si fem ús de la taxonomia de Richard Bartle per classificar els tipus de jugadors, veiem que el nostre perfil encaixa amb el *Killer* (busquen la competència amb jugadors o personatges no jugables i són bastant competitius).

2.5. Estudi de mercat

Abans de començar a prendre decisions sobre el disseny i les necessitats del joc, cal analitzar i estudiar què ofereix la competència en l'actualitat, veure que coses fan bé i quins aspectes són millorables. D'aquesta manera, podrem trobar un espai al mercat i oferir un producte que es diferenciï de la resta.

Per la part de mecàniques i jugabilitat, ja he comentat prèviament que pel projecte he ajuntat les mecàniques de tàctica per torns amb un sistema de combat de màgia.

2.5.1. Estat de l'art

A continuació, llistaré diferents exponents actuals del gènere d'estratègia i de fantasia/magia

- ***Fire Emblem Engage***

Figura 1: A l'esquerra: portada del Fire Emblem Engage. A la dreta: captura in-game del joc.

Al Fire Emblem, el jugador pren el control d'un grup de personatges, a on cada un d'ells té una classe i diferents habilitats. Coloquen al jugador dins d'un mapa i quan el jugador entra en combat, aquest es desenvoluparà en torns i contra diferents monstres controlats per una IA. El combat es pot realitzar utilitzant diferents armes, les quals permeten al jugador adaptar-se a la millor situació.

- ***Civilization VI***

Figura 2: A l'esquerra: portada del Civilitzation VI. A la dreta: captura in-game del joc.

En el cas del Civilitzation VI, el jugador escull una civilització amb l'objectiu de desenvolupar-la a través de l'explotació de recursos naturals i desenvolupar la tecnologia i la cultura d'aquesta.

Mitjançant els torns, el jugador ha de desenvolupar la civilització, mentre tracte amb els països veïns. El jugador pot reinar com vulgui, des d'una tirania a una monarquia i pot fer aliances o iniciar guerres amb altres països.

- ***XCOM 2***

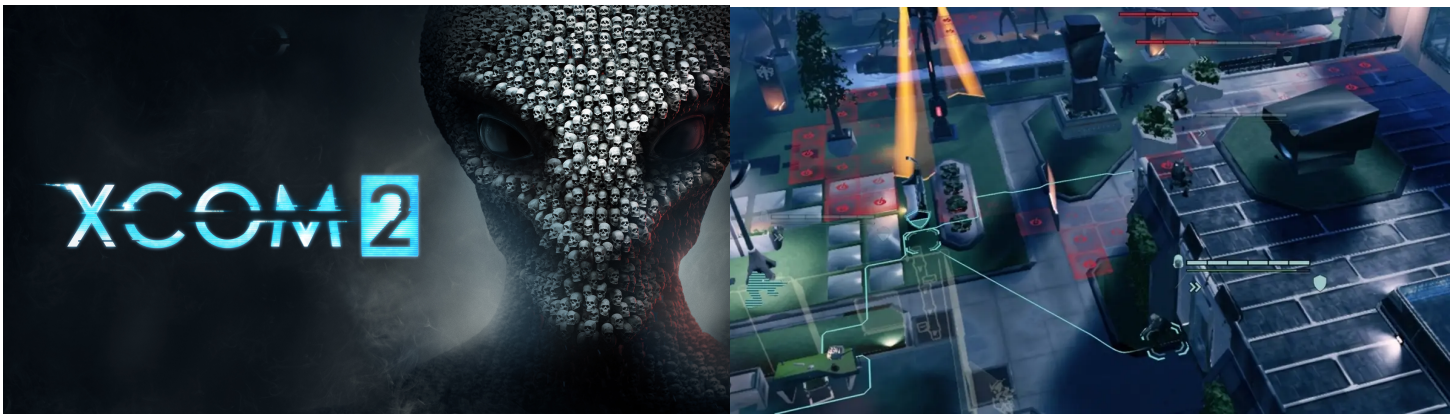


Figura 3: A l'esquerra: portada del XCOM 2. A la dreta: captura in-game del joc.

Aquest joc ha estat la meva principal referència en aquest projecte. En aquest cas, el jugador controla a un grup de personatges, cada un és un jugador completament únic, amb un nom, nivell, armes, habilitats i aspecte. Si algun d'aquests personatges acaba morint dins del nivell, se'l tracta com si realment hagués mort i no es pot tornar a jugar.

Respecte als nivells, al jugador se li dona una missió a completar i se'l deixa a un escenari inexplorat. Un cop fa contacte amb enemics, comença el combat per torns, però depèn de la missió es pot fer sense haver de combatre. Tot això junt, fa que sigui un joc que valoro molt i la raó per la qual hagi escollit seguir endavant amb aquest projecte.

2.5.2. Comparació de l'estat de l'art

Una vegada hem vist quin és l'estat actual del mercat en termes de jocs d'estratègia, procedim a comparar i cercar analogies i possibles necessitats a explotar utilitzant els següents camps a classificar.

- Jugabilitat(JUG): 0-10
- Gràfics i efectes visuals(GEV): 0-10
- Inteligencia artificial(IA): 0-10
- Presa de decisions(PD):0-10
- Dificultat(DIF): 0-10
- Sistema de progrès(SP):0-10
- Duració del joc (DJ): 1-10
- Preu de llançament(PL):0-10

	JUG	GEV	IA	PD	DIF	DJ	SP	PL
Fire Emblem	6	5	6	5	8	30h	7	44,95€
Civilization VI	10	9	9	9.5	9.5	25h	10	49,95€
XCOM 2	10	8.5	8	8.5	9	30h	10	49,99€

Veient la taula anterior, podem arribar a diferents conclusions:

- La IA i la presa de decisions del jugador estan lligats a la dificultat del joc. Com més decisions se li permeten al jugador, més probabilitats de no prendre la correcta, i com més complex sigui la IA, més difícil li posarà al jugador.
- Tenir un sistema de progrés dins del joc és necessari, ja que afegeix complexitat i millora l'experiència del jugador.
- Una de les principals raons de la seva popularitat és la jugabilitat d'aquests jocs, i com una acció pot afectar completament a la resta.

Per tant, creiem que, oferint una duració de 10h (el producte final), un preu de 19,99€, una dificultat més accessible que altres productes similars i una varietat de mecàniques acord amb la duració, el nostre joc tindria cabuda a la indústria.

2.5.3. Model de negoci

Actualment, podem trobar diferents tipus de monetitzacions aplicables al nostre joc, però cal escollir la que millor s'adapti al gènere del joc i al públic objectiu:

- Free-to-play: l'usuari pot accedir al joc gratuïtament. Els beneficis econòmics provenen de publicitat integrada a dins del joc o de microtransaccions que donen al jugador recursos o continguts estètics alternatius.

- Tradicional: el joc requereix un pagament únic abans de començar a jugar. Una vegada s'ha efectuat aquest pagament, tot el contingut del joc està disponible.

- Freemium: l'usuari pot accedir gratuïtament a funcionalitats i parts del joc bàsiques, però altres funcionalitats o seccions avançades requereixen un pagament previ.

- Subscripció: l'usuari ha de pagar mensualitats per poder accedir al contingut del joc. En el moment, que es cancel·la o es deixa de pagar la subscripció, el contingut deixa d'estar disponible.

Analitzant les diferents possibilitats i els tipus de monetització emprats per jocs similars, decidim oferir un model tradicional.

3. Planificació

En aquest apartat, establirem l'organització temporal i de recursos que s'utilitzarà durant tot el procés de desenvolupament. Caldrà definir i dividir correctament les tasques a realitzar, categoritzar aquestes tasques, dissenyar un cronograma que servirà de guia durant el desenvolupament i també establir un protocol o marc de treball per aconseguir una gestió eficient del projecte.

3.1. Tasques a realitzar

- Planificació del projecte
- Preparació i redacció de la memòria
- Selecció de l'entorn de treball
- Disseny de mecàniques i accions
- Disseny sistema de nivells dels personatges
- Implementació de les mecàniques
- Implementació de la IA dels enemics
- Implementació dels Spells
- Implementació sistema combat per torns
- Implementació de les interfícies
- Disseny escenaris
- Cerca i implementació dels models 3D personatges
- Cerca i implementació dels VFX dels Spells
- Cerca i implementació models escenaris
- Depuració d'errors
- Testing

3.2. Diagrama de Gantt

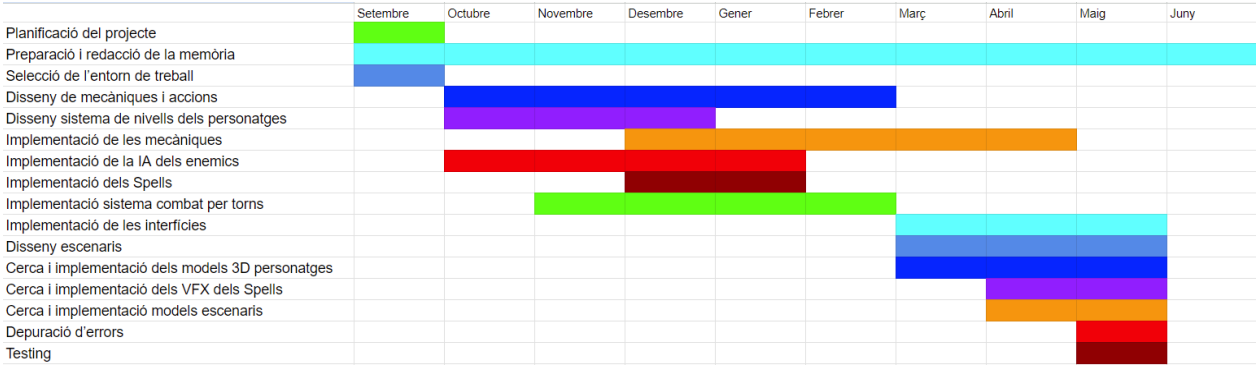


Figura 4: Diagrama de Gantt amb el cronograma estimat durant l'inici del projecte.

3.3. Metodologia de treball

Abans de començar a implementar el prototipus, cal definir una organització eficient del temps i esforços que cal invertir. En concret, ens decidim per fer ús de la metodologia SCRUM:

- En comptes de planificar a molt llarg termini què s'implementarà en els següents mesos i anar desenvolupant petites parts del joc per després ajuntar-les, es fa una versió molt reduïda del joc i es realitza un desenvolupament incremental, on es va augmentant la qualitat del producte.

- Es fa una llista ordenada per prioritat de totes les mecàniques i característiques que es volen afegir al joc. Aquest ordre dictaminarà a cada moment del desenvolupament quin és el següent bloc de treball prioritari.

- Encara que hi hagi apartats amb prioritat per poder oferir un producte mínim acabat, les diferents etapes o seccions del desenvolupament no es realitzen de manera totalment seqüencial, sinó que es van superposant, fent que el producte creixi en qualitat en tots els apartats a la vegada.

- El treball es divideix en Sprints de dues setmanes de duració (el temps entre tutories). Abans de començar el Sprint es decideix què s'implementarà durant aquest, i es remarca quin grau de prioritat té cada apartat.

- A les tutories (similar a les iteracions amb els clients d'un producte de software) es mostren els avenços realitzats durant el Sprint i es prenen les decisions adequades sobre els següents passos.

3.4. Eines de treball

Com s'ha esmentat a l'apartat 2.1.2, a l'hora de desenvolupar un projecte d'aquestes característiques és necessari la utilització de diversos software per generar el contingut necessari, tant artístic com lògic. A continuació s'esmenten els diferents programes i la implicació en el projecte.

3.4.1. Motor del joc

El pilar fonamental del desenvolupament d'un videojoc, sobretot si es tracta d'un projecte independent, és la utilització d'un motor de videojocs capaç de proporcionar-nos eines per l'elaboració d'un joc. En el nostre cas, a causa del cicle realitzat prèviament a accedir a la universitat, el nostre coneixement de Unreal Engine ens va fer decantar per aquest motor, ja que demostrava tenir eines de creació de IA més complexes que els altres motors, a part que el fet de programar de manera més visual era un punt a favor.

Un cop escollit el motor, era necessari escollir la versió, i similar al comentat prèviament, vam continuar amb la versió utilitzada al cicle, la 4.22.3.

3.4.2. Editor d'imatges

Com he comentat prèviament, l'editor d'imatges utilitzat ha estat l'Adobe Photoshop 2020. Ha estat l'eina feta servir per editar les imatges que posteriorment s'han fet servir per a la creació del HUD.

4. Marc de treball

En aquesta secció detallarem quins són els conceptes que cal entendre prèviament a la lectura de la present memòria i quin programari s'ha utilitzat durant la totalitat del projecte.

4.1. Conceptes previs

- Spells: Atacs que utilitzarà el personatge i la IA per derrotar a l'equip contrari. La paraula Spells tindria la traducció Catalana de "encanteri". Com la traducció directa no es fa servir mai al món dels videojocs, he decidit fer servir el terme estàndard "Spell".

- Estat cobert: Estat del personatge o enemic que s'obté quan aquest s'aproxima a una cobertura. Es manté fins que marxa el personatge. En aquest estat, el personatge rep menys mal.

- Estat flanquejat: Estat del personatge o enemic que apareix quan un personatge enemic es troba flanquejat. Per trobar-se flanquejat, el personatge o enemic ha de trobar-se en una cobertura (ex: a dalt) i el personatge de l'altre equip ha de trobar-se en la direcció contrària (ex: a baix).

5. Disseny de videojoc

Al present capítol, desenvoluparem en profunditat la idea inicial i concretarem, d'acord amb l'objectiu del projecte, l'escala, les possibilitats i les limitacions del disseny del prototipus, sigui estèticament, narrativament o mecànicament. En específic, s'explicaran les funcionalitats més importants que més endavant s'implementaran i es detallaran altres elements que responen a les necessitats del joc.

5.1. Mecàniques

L'apartat clau del projecte rau en les mecàniques i la implementació d'aquestes, el que cal decidir són les entitats i els recursos presents al joc, quina és la relació entre aquestes entitats, quins reptes es presentaran al jugador, quines accions pot realitzar per superar aquests reptes, quina informació rep el jugador en cada moment, etc.

5.1.1. Espai de joc

Tenint en compte la naturalesa del projecte (es tracta d'un prototipus per demostrar el potencial del projecte finalitzat), l'espai del joc no necessita ser gaire ambiciós, només cal que serveixi per a una funció concreta: permetre veure l'evolució mecànica del joc.

Aquest prototip consta de 3 nivells diferents i independents. Aquests nivells es poden separar per la dificultat i són els següents;

- Primer nivell. Veure Figura 5.

Figura 5: captura del viewport del primer nivell

El primer nivell, el qual consta d'un espai reduït (16x19) i menor nombre d'enemics (2), aquest es podría considerar el tutorial.

- Segon nivell. Veure Figura 6.

Figura 6: captura del viewport del segon nivell

El segon nivell (nivell normal), es caracteritza per ser més gran que l'anterior (16x23) i amb un major nombre d'enemics (4), però continua ambientat en el mateix bosc.

- Tercer nivell. Veure Figura 7

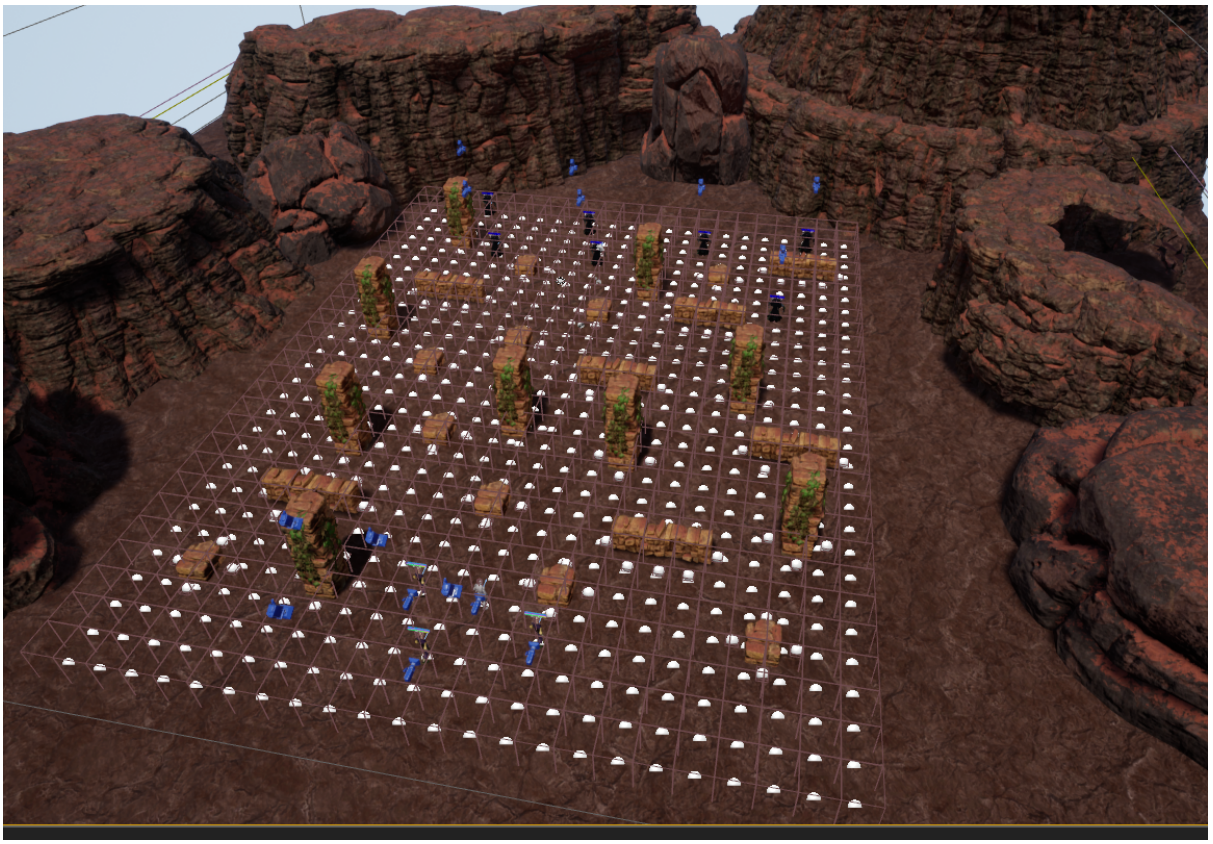


Figura 7: captura del viewport del tercer nivell

El tercer nivell (nivell difícil), també augmenta de mida (29x23) i de nombre d'enemics (7).

Cada nivell limita l'accés del jugador segons el progrés: fins que el jugador no ha completat el nivell 1, no pot accedir al nivell 2, ja que aquest no es pot clicar al menú. Tot i que actualment sí que es permet rejugar un nivell, en un futur s'eliminarà aquesta opció, pel fet que una de les mecàniques que hem pensat implementar és la pujada de nivells dels personatges. Per tant, rejugar nivells serviria per pujar de nivell els personatges en un entorn controlat i facilitaria en gran manera la superació dels propers nivells.

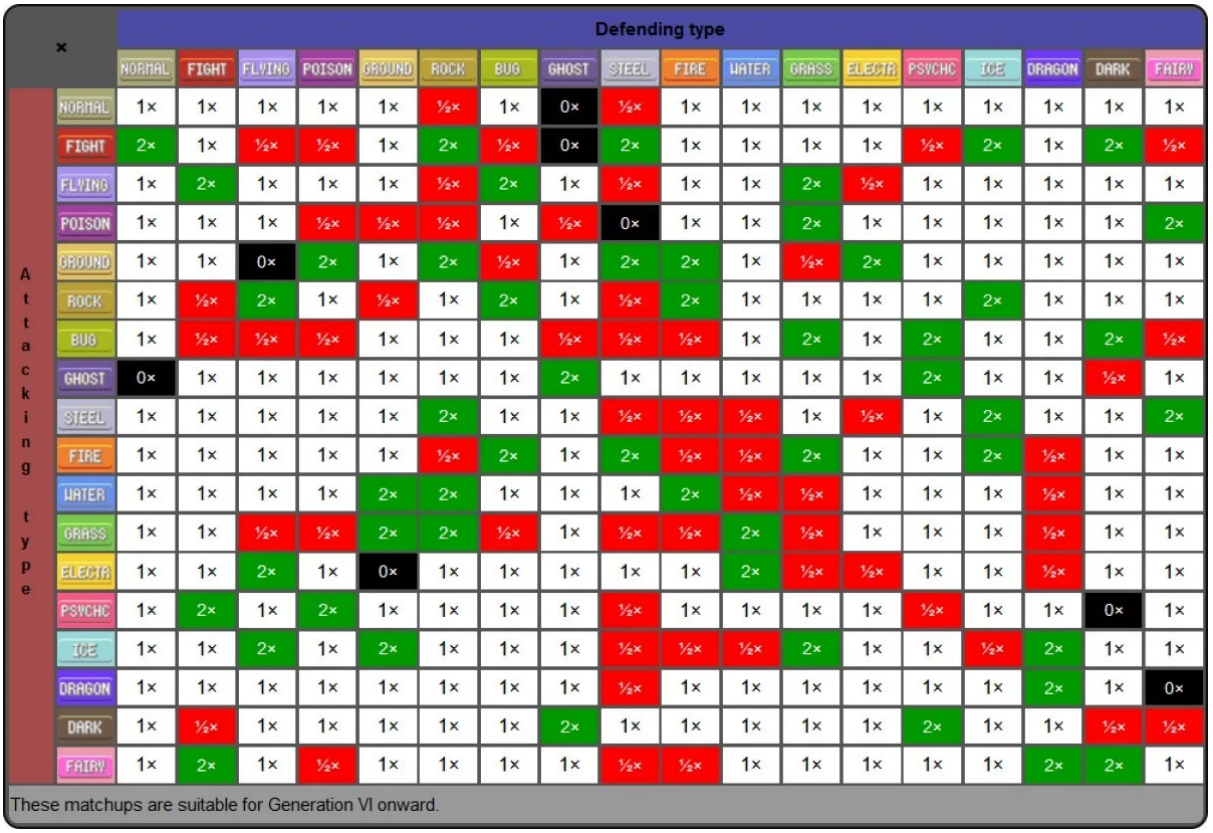
5.1.2. La mecànica del joc, els reptes que ha d'afrontar el jugador i les accions possibles

La definició de les mecàniques és un dels punts més importants a l'hora de donar personalitat al joc, ja que són les regles que regeixen el món i que orientarà l'experiència del jugador cap a un gènere en concret. En aquest cas, i a causa que el gènere pot tenir una gran varietat, podem tenir un gran nombre de mecàniques i accions. Tal com s'esmenta en apartats anteriors, el joc està inspirat en un seguit de mecàniques que, combinades entre si, donen lloc a una fórmula interessant i innovadora.

A l'inici del projecte ens vam plantejar aplicar mecàniques que permetessin al jugador interactuar amb el món i, sobretot, plantejar un entorn que se sentís viu, on el jugador hauria d'interactuar per poder avançar. Tot i que des del principi teníem clar el gènere, van aparèixer diverses idees sobre comportar-lo a la pràctica. Entre aquestes teníem molt clar que volíem que el jugador tingués la capacitat de formar l'equip a partir d'una llista de personatges, i que pogués modificar els atacs de cada un d'ells en funció del nivell. Un altre aspecte que tenia molt clar, era el fet de implementar un sistema de cobertures que augmenta les probabilitats de sobreviure i esquivar dels atacs, i a partir d'aquest sistema de cobertures, implementar també un sistema per flanquejar els enemics utilitzant-les.

5.1.3. Màgia

Per crear un sistema de combat màgic, vaig basar-me en la taula de tipus del Pokemon. Aquesta taula es refereix a la resolució dels atacs i com influeixen depenent dels elements involucrats per acabar mostrant el multiplicador que se li afegirà al poder de l'atac abans de la resolució del mal. Veure Figura 8.



×	Defending type																	
Attacking type	NORMAL	FIGHT	FLYING	POISON	GROUND	ROCK	BUG	GHOST	STEEL	FIRE	WATER	GRASS	ELECTR	PSYCHC	ICE	DRAGON	DARK	FAIRY
NORMAL	1×	1×	1×	1×	1×	½×	1×	0×	½×	1×	1×	1×	1×	1×	1×	1×	1×	1×
FIGHT	2×	1×	½×	½×	1×	2×	½×	0×	2×	1×	1×	1×	1×	½×	2×	1×	2×	½×
FLYING	1×	2×	1×	1×	1×	½×	2×	1×	½×	1×	1×	2×	½×	1×	1×	1×	1×	1×
POISON	1×	1×	1×	½×	½×	½×	1×	½×	0×	1×	1×	2×	1×	1×	1×	1×	1×	2×
GROUND	1×	1×	0×	2×	1×	2×	½×	1×	2×	2×	1×	½×	2×	1×	1×	1×	1×	1×
ROCK	1×	½×	2×	1×	½×	1×	2×	1×	½×	2×	1×	1×	1×	1×	2×	1×	1×	1×
BUG	1×	½×	½×	½×	1×	1×	1×	½×	½×	½×	1×	2×	1×	2×	1×	1×	2×	½×
GHOST	0×	1×	1×	1×	1×	1×	1×	2×	1×	1×	1×	1×	1×	2×	1×	1×	½×	1×
STEEL	1×	1×	1×	1×	1×	2×	1×	1×	½×	½×	½×	1×	½×	1×	2×	1×	1×	2×
FIRE	1×	1×	1×	1×	1×	½×	2×	1×	2×	½×	½×	2×	1×	1×	2×	½×	1×	1×
WATER	1×	1×	1×	1×	2×	2×	1×	1×	1×	2×	½×	½×	1×	1×	1×	½×	1×	1×
GRASS	1×	1×	½×	½×	2×	2×	½×	1×	½×	½×	2×	½×	1×	1×	1×	½×	1×	1×
ELECTR	1×	1×	2×	1×	0×	1×	1×	1×	1×	1×	2×	½×	½×	1×	1×	½×	1×	1×
PSYCHC	1×	2×	1×	2×	1×	1×	1×	1×	½×	1×	1×	1×	1×	½×	1×	1×	0×	1×
ICE	1×	1×	2×	1×	2×	1×	1×	1×	½×	½×	½×	2×	1×	1×	½×	2×	1×	1×
DRAGON	1×	1×	1×	1×	1×	1×	1×	1×	½×	1×	1×	1×	1×	1×	1×	2×	1×	0×
DARK	1×	½×	1×	1×	1×	1×	1×	2×	1×	1×	1×	1×	1×	2×	1×	1×	½×	½×
FAIRY	1×	2×	1×	½×	1×	1×	1×	1×	½×	½×	1×	1×	1×	1×	1×	2×	2×	1×

These matchups are suitable for Generation VI onward.

Figura 8: Taula de tipos Pokemon

Mitjançant aquesta taula, vaig simplificar-la fins a obtenir 5 elements diferents que poguessin tenir cert contrast entre ells, deixant-me amb els següents; Terra, Foc, Aigua, Electricitat i Aire. Veure Figura 9.

Figura 9: Captura treta del menú d'edició del personatge

Un cop definit els elements, vaig crear una relació entre aquests, creant la meva pròpia taula per definir el multiplicador. Acabant per obtenir la següent taula. Veure Figures 10 i 11.

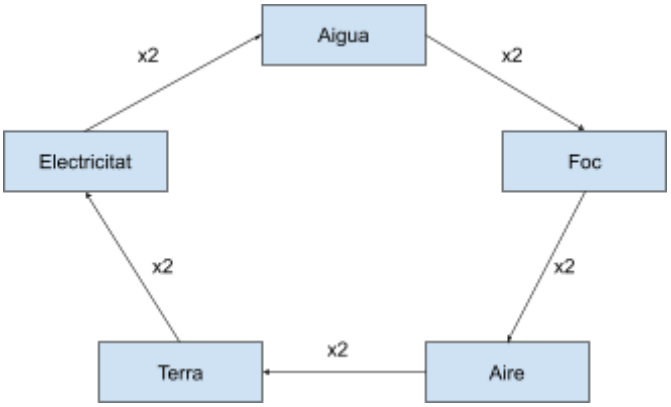


Figura 10: Esquema multiplicadors

		Defensor				
		Terra	Electricitat	Aigua	Foc	Aire
Atacant	Aire	x2	x1	x1	x0.5	x1
	Terra	x1	x2	x1	x1	x0.5
	Electricitat	x0.5	x1	x2	x1	x1
	Aigua	x1	x0.5	x1	x2	x1
	Foc	x1	x1	x0.5	x1	x2

Figura 11: Taula multiplicadors

Un apartat de la màgia que volia agregar era el fet que cada personatge pugues aprendre qualsevol tipus de màgia sense importar l'elemet que eren. D'aquesta manera no es depèn completament de l'element del personatge, però continua sent un punt important a l'hora de crear un equip.

5.1.4. Sistema de combat

En la creació del sistema de combat, una de les principals mecàniques a implementar va ser un sistema de probabilitat d'encertar l'atac. Basant-me en el joc mencionat prèviament, XCOM, un dels elements que més influïen en el combat era la probabilitat d'encertar un atac, ja que et forçava a observar l'escenari a la recerca de millorar les

posicions de les tropes per facilitar que l'atac doni a l'enemic alhora que es vigila de reduir la probabilitat de rebre atacs. Aquesta probabilitat es calcula a partir de la distància dels dos personatges (atacant i defensor) i de la detecció d'objectes entre ells.

A part d'aquesta probabilitat, en el combat també es modifica el dany final comprovant si el defensor està sent flanquejat o si no es troba en una cobertura. D'aquesta manera també es penalitza descuidar les tropes i es beneficia el fet de crear una estratègia adequada a la situació.

5.1.5. Accions

Les accions del jugador es poden simplificar en:

- Caminar
- Atacar

Però, depenent de l'acció que el jugador faci; amb el personatge que ho faci, pot obtenir un resultat o un altre. Per exemple; si dos personatges poden atacar, però un dels dos està flanquejant a l'enemic, el mal que pot inflingir serà potencialment més elevat que en el cas que l'ataqui l'altre. Això també succeeix en el cas d'atacar amb un Spell contrari l'element de l'enemic; es fa més mal que en el cas d'atacar amb un altre; o inclús amb l'acció de caminar, es pot moure per flanquejar a un altre enemic, o evitar que flanquegin al jugador. Per tant, tot i que el llistat d'accions és reduït, dona una gran varietat de resultats diferents.

5.1.4. Enemics del joc

Els enemics són una part fonamental del videojoc; sense una IA que actuï de forma coherent davant de determinats estímuls, l'experiència del jugador pot perdre qualitat i pot baixar la immersió. Per aquesta raó, la IA va ser una de les parts fonamentals en el desenvolupament d'aquest projecte. Per a aquesta versió del projecte, hi ha implementat diferents versions de l'enemic.

- Enemic de foc. Veure Figura 12.

Figura 12: Captura del viewport de l'actor de l'enemic de foc

- Enemic d'aigua. Veure Figura 13.

Figura 13: Captura del viewport de l'actor de l'enemic d'aigua

- Enemic de terra. Veure Figura 14.

Figura 14: Captura del viewport de l'actor de l'enemic de terra

5.1.4.1. Tipologia dels enemics

Els enemics que volem implementar el joc s'han d'adaptar a l'objectiu del projecte i al perfil de jugador:

- Els enemics han de respectar el sistema de torns i les accions per torn que poden realitzar
- Els enemics han de decidir quin objectiu és millor per atacar.
- Els enemics han de decidir a quina posició s'han de moure amb l'objectiu d'apropar-se o de flanquejar a un personatge.

5.1.4.2. Comportament dels enemics: relació amb les accions del jugador

El comportament dels enemics s'implementarà amb un Behaviour Tree, on cada estat o "branca" necessita, conceptualment, una prioritat diferent de la resta de branques.

En qualsevol cas, té sentit decidir que atacar als personatges del jugador té prioritat respecte a canviar de posició, si fos al contrari els enemics només es mourien pel mapa sense interactuar amb el jugador.

5.2. Interfícies

El joc necessita informar al jugador dels estats de diverses variables importants. Més endavant veurem quina és l'aparença i posició d'aquestes interfícies, però abans cal mencionar-les, explicar-les i classificar-les.

- Menú principal: per començar la partida.

- Menú pausa: ha de permetre tornar al menú i continuar el joc.

- Pantalla mort del jugador: ha de permetre tornar al menú.

- Pantalla victòria: ha de permetre tornar al menú.

- Menú elecció nivell: per escollir el nivell que es vol jugar.

- Menú editor equip: per agregar o treure membres de l'equip, a més, cada membre té un botó per editar els Spells.

- HUD ingame: per mostrar els Spells i la probabilitat d'encertar.

- Menú elecció de personatge: per escollir un personatge per agregar a l'equip.

- Menú seleccionar Spells: per modificar els Spells que tindrà el personatge escollit ingame.

Aquestes interfícies les podem classificar segons l'existència a la història del joc i al món físic del joc:

- Interfície no-diegètica: les entitats del joc desconeixen l'existència i no existeix al món físic del joc.

- Interfície diegètica: les entitats del joc poden veure la interfície i existeixen al món físic del joc

- Interfície meta: el protagonista és conscient de l'existència d'aquesta interfície, però no ocupa espai al món físic.

- Interfície espacials: les entitats no poden veure la interfície, però aquestes ocupen un espai tridimensional a dins del joc.

Amb aquesta classificació, podem categoritzar les interfícies del joc de la següent manera:

- Interfícies no-diegètiques
    - Menú principal
    - Menú pausa
    - Pantalla mort del jugador
    - Pantalla victòria
    - Menú elecció nivell
    - Menú editor equip
    - Menú elecció de personatge
    - Menú seleccionar Spells

- Interfícies meta
    - HUD ingame

Per tal d'entendre millor i visualitzar la funció al joc, seguidament es mostraran les captures on podem veure les interfícies principals ja implementades al joc.

- Menu principal. Veure Figura 15.

Figura 15: Captura del viewport del menú principal.

- Menú pausa. Veure Figura 16.

Figura 16: Captura del viewport del menú pausa.

- Pantalla mort del jugador. Veure Figura 17.

Figura 17: Captura del widget de la pantalla de mort del jugador.

- Pantalla victòria. Veure Figura 18.

Figura 18: Captura del viewport de la pantalla de victòria.

- Menú elecció nivell. Veure Figura 19.

Figura 19: Captura del viewport del menú d'elecció de nivell.

- Menú editor equip. Veure Figura 20.

Figura 20: Captura del viewport del menú d'editor de l'equip.

- Menú elecció personatge. Veure Figura 21.

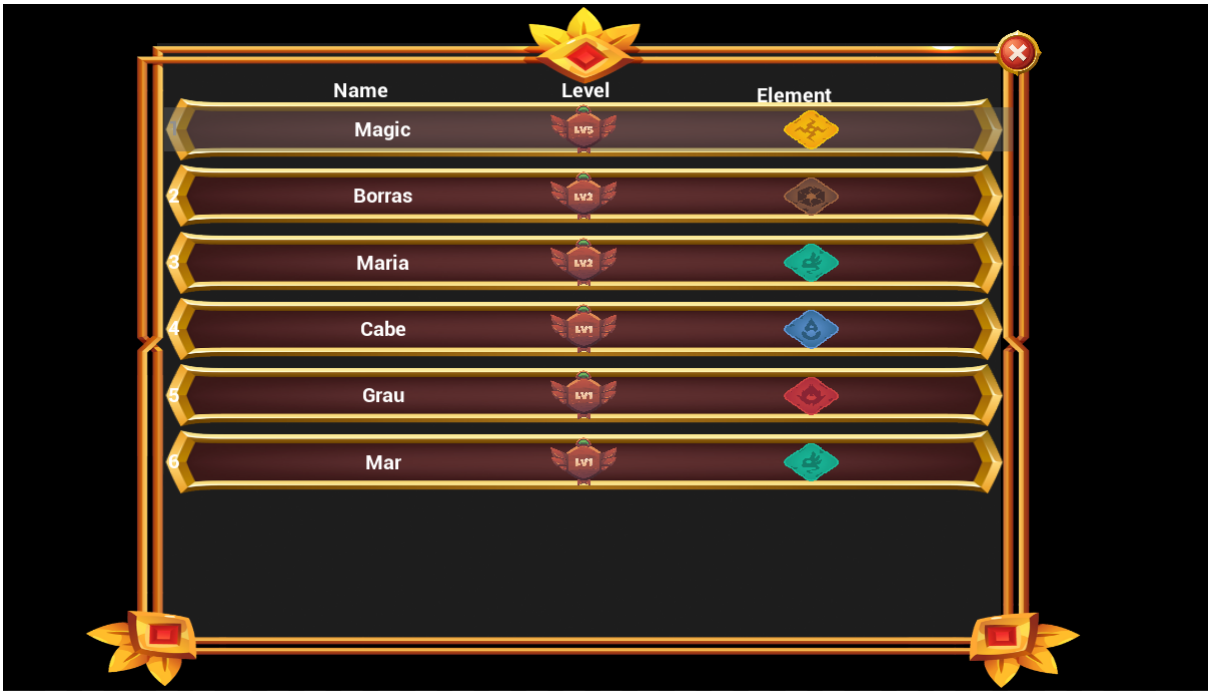


Figura 21: Captura del viewport del menú d'elecció de personatge.

- Menú seleccionar Spells. Veure Figura 22.

Figura 22: Captura del viewport del menú de seleccionar Spells.

5.3. Elements de feedback

El jugador necessita rebre certes respostes instantànies per part del joc per sentir que les accions tenen conseqüències i que és ell qui té el control. Per aquest motiu és molt important dissenyat alguns elements bàsics de feedback que supleixin aquesta necessitat:

- Modificació elements 3D: quan el jugador estigui jugant una partida, a l'atacar, la barra de vida de l'enemic que rebi l'atac es reduirà si l'ha encertat. Aquesta reducció serà relativa al dany infligit. Veure Figura 23.

Figura 23: Imatge de la barra de vida dels personatges

- Modificació d'imatges 2D: quan el jugador estigui modificant els Spells d'un personatge, les imatges dels Spells seleccionats variaran depenent d'on cliqui. Veure Figura 24 i Figura 25.

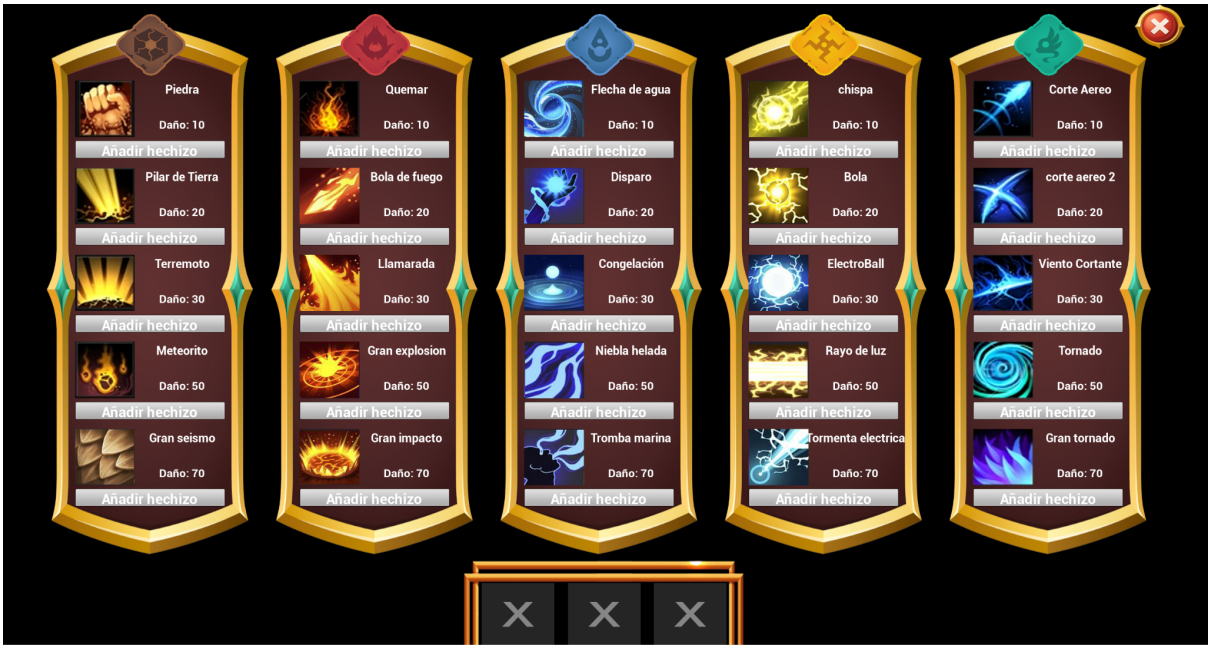


Figura 24: Imatge dels Spells d'un personatge buit

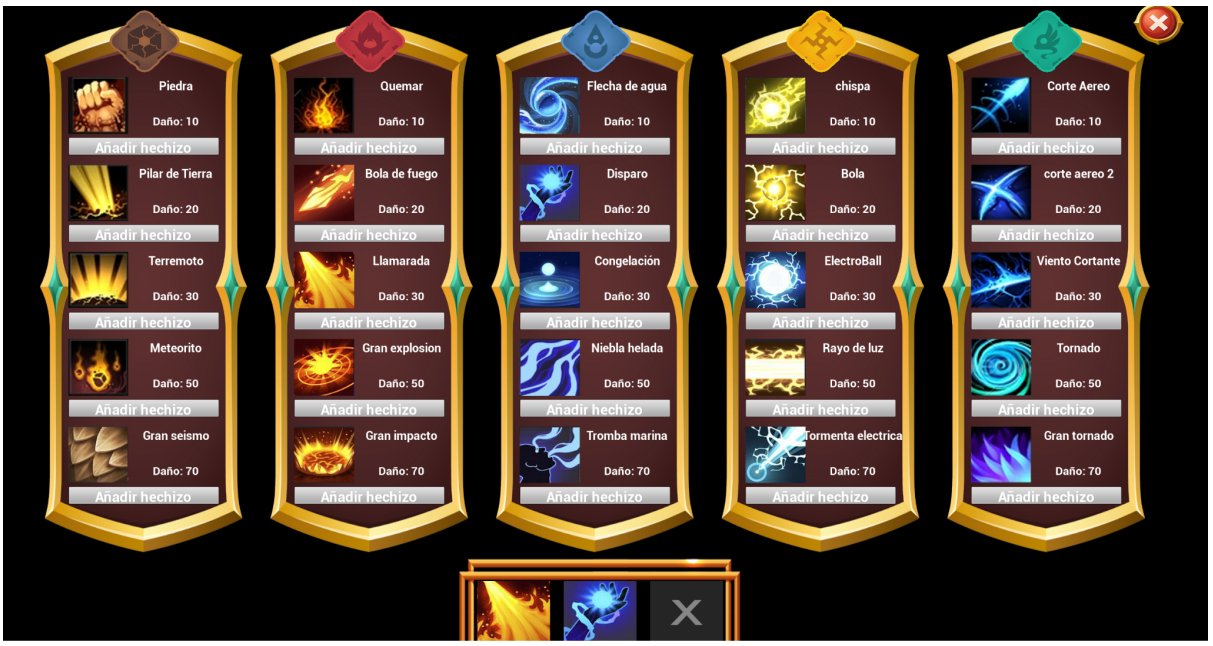


Figura 25: Imatge multiples Spells seleccionats

5.4. Escenaris, models, animacions i interfícies

Fent ús de l'Unreal Marketplace i dels seus productes gratuïts per temps limitat, hem tingut accés a una àmplia llibreria de paquets que han estat de gran ajuda per crear tota la part artística del projecte. Tots els paquets que es mostraran a continuació han estat agregats de forma gratuïta al projecte mitjançant el "Free for the month" d'Unreal Marketplace.

5.4.1. Escenaris

Els escenaris dels nivells implementats, han estat extrets de diferents paquets del Unreal Marketplace.

- Escenari 1 i 2

  Per la creació d'aquests escenaris, he implementat el paquet anomenat "Dreamscape Nature : Meadows - Stylized Open World Environment". Tota la part artística de l'escenari juntament amb els models utilitzats per les cobertures formen part d'aquest paquet. Veure Figures 26 i 27.

Figura 26: Imatge treta del paquet “Dreamscape Nature : Meadows - Stylized Open World Environment” Unreal Marketplace

Figura 26: Mapa showcase del paquet “Dreamscape Nature : Meadows - Stylized Open World Environment”

- Escenari 3

Per la creació d'aquest escenari, he implementat el paquet anomenat "Modular Lost Ruins Kit". D'aquest paquet s'ha tret tota la part artística de l'escenari 3, juntament amb els models utilitzats per les cobertures. Veure Figures 28 i 29.

Figura 28: Imatge treta del paquet "Modular Lost Ruins Kit" Unreal Marketplace

Figura 29: Mapa showcase del paquet "Modular Lost Ruins Kit"

5.4.2. Models

Els models 3D dels personatges han estat importats al projecte a partir de l'Unreal Marketplace o la coneguda pàgina web Mixamo (https://www.mixamo.com/), i es poden separar en 2 principals subapartats:

5.4.3.1. Model dels personatges

Respecte al model 3D del personatge, aquest ha sorgit de la pàgina web comentada prèviament Mixamo. El model es troba a l'apartat de personatges sota el nom de "Abe". Veure Figura 30.

Figura 30: Imatge del personatge "Abe" dins l'entorn de Mixamo

5.4.3.2. Model dels enemics

Pel tema dels enemics i mitjançant Unreal Marketplace, vaig trobar un paquet que mantenia una estètica coherent amb el personatge i el fons. El paquet s'anomenava "Undead Pack" i disposava de diferents personatges no morts per utilitzar. Després d'analitzar els diferents personatges, vaig acabar implementant el monstre anomenat "Lich". Veure Figures 31 i 32.

Figura 31: Imatge treta del paquet de monstres "Undead Pack" d'Unreal Marketplace

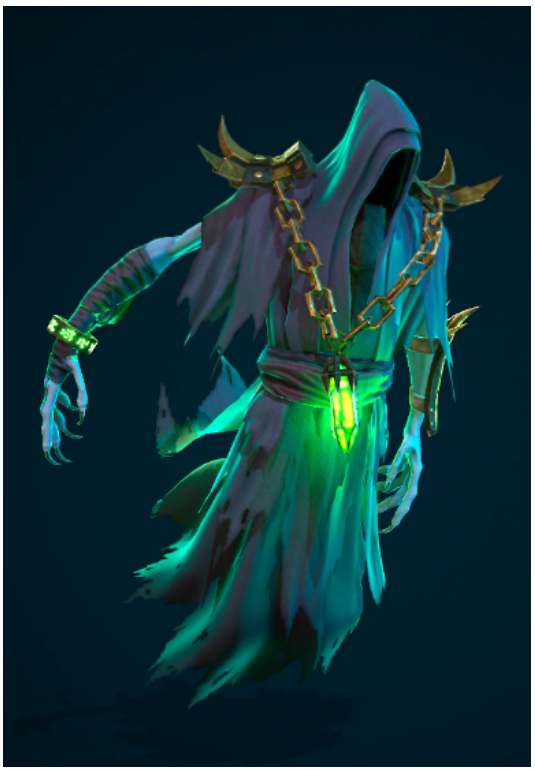


Figura 32: Imatge del Lich del paquet “Undead Pack” d’Unreal Marketplace

5.4.3. Animacions

L’apartat d’animacions es pot dividir en tres principals subapartats, les animacions de personatge, animacions dels enemics i les animacions de VFX.

5.4.3.1. Animacions de personatge

Les animacions, igual que el personatge, han estat extretes de la pàgina web de Mixamo, de l’apartat d’animacions.

5.4.3.2. Animacions dels enemics

Gràcies al paquet implementat pel model del ”Lich”, aquest ja venia amb animacions pròpies, tot i que en el cas d’haver-ne necessitat més, hauria utilitzat la pàgina de Mixamo, la qual té un apartat per pujar el personatge i baixar-lo amb les animacions.

5.4.3.3. VFX

Els VFX utilitzats han estat molts i molt diversos. Per tant he hagut de buscar múltiples paquets d'Unreal Marketplace. Els paquets utilitzats són "Advanced Magic FX 12" (Veure Figura 33), "Advanced Magic FX 13" (Veure Figura 34), "FX Variety Pack" (Veure Figura 35) i "RPG FX Starter Pack" (Veure Figura 36).

Figura 33: Imatge del paquet "Advanced Magic FX 12" d'Unreal Marketplace

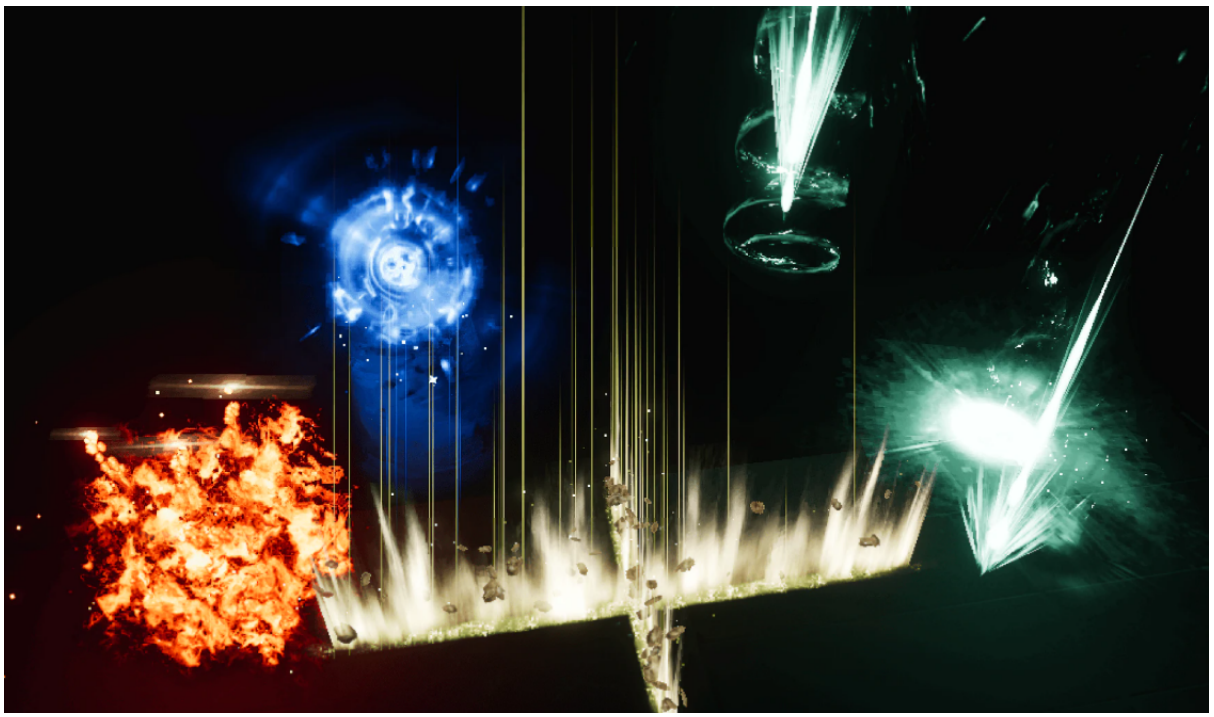


Figura 34: Imatge del paquet “Advanced Magic FX 12” d’Unreal Marketplace

Figura 35: Imatge del paquet “FX Variety Pack” d’Unreal Marketplace

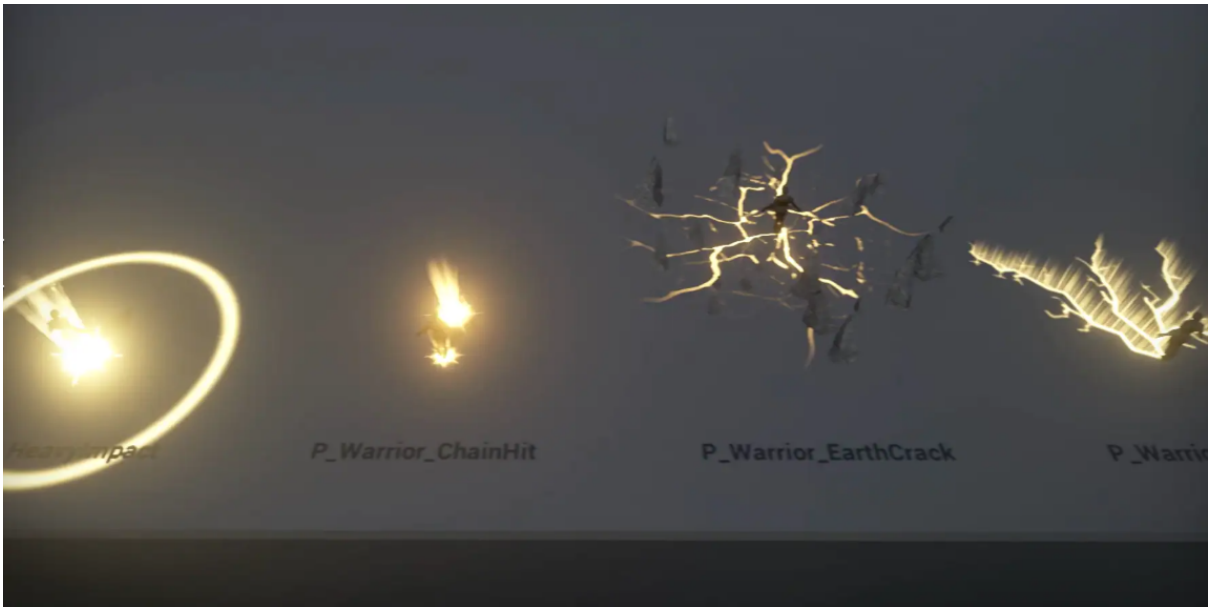


Figura 36: Imatge del paquet “RPG FX Starter Pack” d’Unreal Marketplace

5.4.4. Interfícies

En el cas de les interfícies, ha estat més complex que en els altres casos, ja que no hi havien paquets que s’ajustessin a les necessitats. A causa d'això, vaig necessitar buscar per internet pàgines a on poder descarregar-me els menús. La pàgina que vaig acabar trobant es diu “freepik” (https://www.freepik.com/) i permet descarregar de forma gratuïta molt del contingut. Vaig acabar per instal·lar diferents paquets per cobrir les necessitats, acabant amb els següents paquets:

5.4.4.1. Sprite elements

Gràcies al freepik, vaig trobar un post amb els elements amb una estètica similar al que jo buscava i accés a la descàrrega. Veure Figura 37.

Figura 37: Imatge dels logotips dels elements extreta de la pàgina de descàrrega (https://www.freepik.com/free-vector/beautiful-colorful-square-buttons-with-light-border-vector-assets-game-decorative-gui-elements-isolated_13466774.htm#query=beautiful-colorful-square-buttons-with-light-border-vector-assets-game-decorative-gui-elements-isolated&position=0&from_view=search&track=sph)

5.4.4.2. Sprite nivells

Ja que no volia mostrar el nivell dels jugadors amb un simple text, vaig buscar una imatge a on es mostressin els nivells que volia utilitzar, del nivell 1 al 5. Fent servir frepik vaig arribar a trobar uns banners que s'adaptaven a les necessitats. Veure Figura 38.

Figura 38: Imatge dels logotips del nivell extreta de la pàgina de descàrrega (https://www.freepik.com/free-vector/game-level-ui-icons-wooden-shields-banners_29086895.htm#query=game-level-ui-icons-wooden-shields-banners&position=12&from_view=search&track=sph)

5.4.4.3. Sprite frames menú

A l'hora de llistar els personatges jugables, era necessari trobar un frame a on poder mostrar la taula amb els resultats. Després de certa recerca, vaig trobar aquesta imatge, la qual després d'uns quants retocs vaig acabar utilitzant. Veure Figura 39.

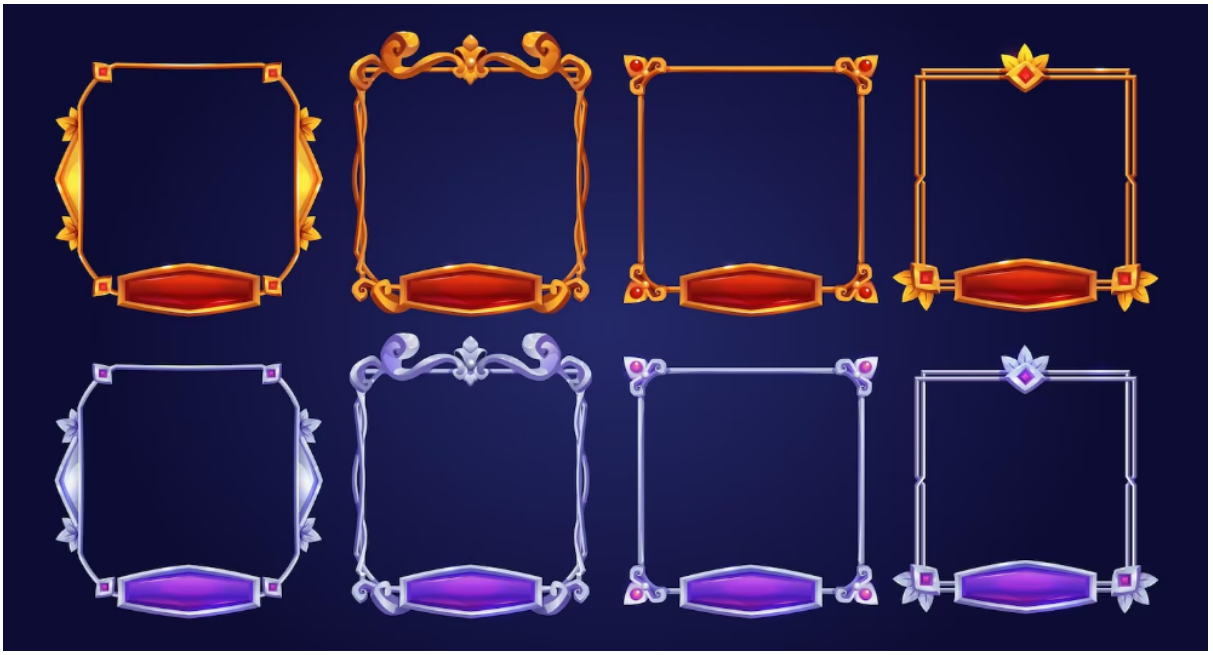


Figura 39: Imatge dels frames extreta de la pàgina de descàrrega (https://www.freepik.com/free-vector/golden-silver-vintage-game-frames-borders-buttons_30534347.htm#page=5&query=medieval%20game%20menu%20paper%20hud&position=28&from_view=search&track=robertav1_2_sidr)

5.4.4.4. Botons i sprite frames menú 2

A l'hora de fer els botons, necessitava una imatge per la qual substituir el botó default, a part que una altra referència per crear nous menús diferents als anteriors. Gràcies a freepik, vaig acabar trobant-me amb aquest resultat. Veure Figura 40.

Figura 40: Imatge extreta de la pàgina de descàrrega (https://www.freepik.com/free-vector/set-game-frames-menu-buttons-cartoon-interface-empty-borders-with-royal-crown-banners-ui-gui-design-elements-slider-red-green-keys-user-panel-sliders-settings-isolated-vector-set_20903007.htm#query=medieval%20game%20ui%20element&position=27&from_view=search&track=robertav1_2_sidr)

5.4.4.5. Sprites dels Spells

Per la cerca d'aquests Sprites, la pàgina de freepik no va ser de gaire utilitat, així que vaig optar per cercar a pinterest (https://www.pinterest.es), una de les pàgines més importants en el sector. Gràcies a aquesta vaig trobar diferents imatges de les quals vaig acabar traient els sprites finals. Veure Figures 41 i 42.

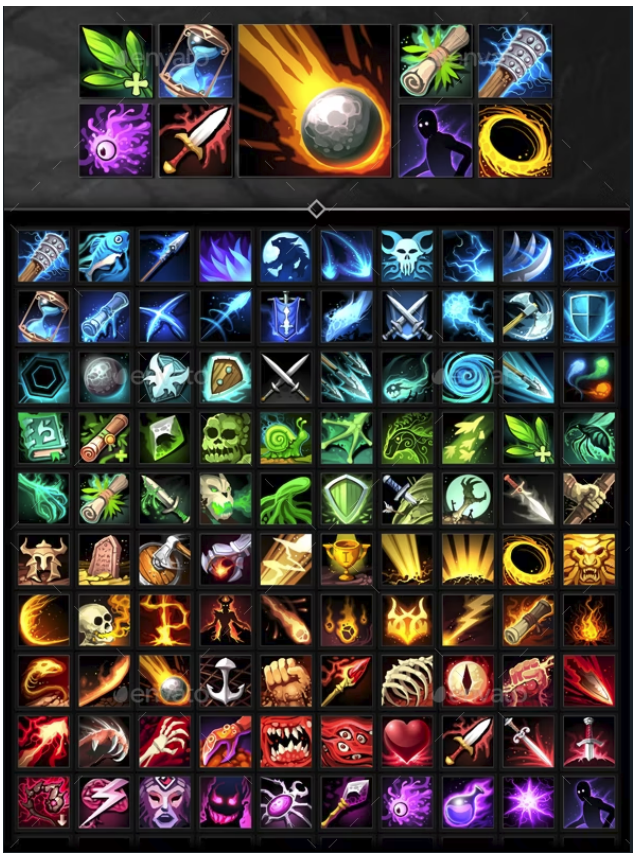


Figura 41: Imatge dels sprites dels Spells extreta de la pàgina de descàrrega (https://www.pinterest.es/pin/841469511638492856/)

Figura 42: Imatge dels sprites dels Spells extreta de la pàgina de descàrrega (https://www.pinterest.es/pin/567242515569586086/)

5.5. Estructura i decisions de disseny del projecte

El projecte d'Unreal tindrà certes característiques que neixen fruit de certes decisions de disseny i necessitats del prototipus

5.5.1. Variables globals

El singleton o instància única és un patró de disseny de software que proveeix el projecte d'una única instància global que ens permetrà tenir variables d'accés global (variables de la partida) que no dependran de l'escena a la qual es troba el jugador.

Algunes de les variables globals més importants són: la llista de personatges, la llista de Spells, la llista de personatges seleccionats, la llista dels elements, etc.

5.5.2. Dispositius d'entrada

Donat que és un prototipus, el joc únicament es podrà jugar amb teclat i ratolí, ja que és necessari detectar els clics del jugador.

6. Implementació i proves

A l'apartat actual s'expliquen detalladament els mètodes més tècnics empleats a Unreal Engine per portar a terme la totalitat de la implementació del prototipus.

6.1. Glossari de termes rellevants d'Unreal Engine

A continuació expliquem alguns conceptes interns d'Unreal que guanyaran rellevància en els apartats següents:

- Blueprint: es tracta d'un script visual basat en grafs de nodes. Tots els Actors d'un projecte d'Unreal tenen associat un Blueprint que permet determinar el comportament programàtic. Conceptualment parlant, un Blueprint és a Unreal el que a C++ són els arxius ".cpp" i ".h".

- Event graph: és la part de cada Blueprint que conté Events.

- Events: són les accions (void() en C++) de cada Blueprint del projecte. Hi ha certs Events predeterminats d'Unreal i també es poden crear Events personalitzats. Aquests Events poden tenir variables d'entrada (paràmetres), però no retornen un valor, simplement executen tasques. Poden contenir tasques asíncrones.

- Functions: són les funcions (que poden actuar com accions i no retornar cap valor) de cada Blueprint. Aquestes funcions no poden contenir tasques asíncrones (com les Timelines). Poden ser privades o públiques (accessibles per altres Actors i Blueprints, o no).

- Tick Event: es tracta d'un event predeterminat d'Unreal que es crida a cada frame. En altres motors de jocs es coneix com Update() de C#.

- BeginPlay Event: és l'Event predeterminat del Blueprint de cada Actor que es crida al començament de l'execució. Quan l'Actor entri en escena al joc, és a dir, l'escena a la qual apareix carregui, es cridarà a l'Event BeginPlay.

- Material: són un tipus específic de Blueprint que no van associats a cap Actor predeterminat, i serveixen per crear materials amb textures i shaders.

- Material Instance: de cada Material es poden fer instàncies diferents que tinguin com a paràmetres valors diferents. És a dir, si un Material utilitza una variable numèrica com a paràmetre per formar el material final, si es crea una instància d'aquest material, es podrà canviar aquest paràmetre numèric a un valor específic i únic d'aquella instància.

- Widget: es tracta d'un element 2D en pantalla, és a dir, un element de la interfície. Cadascuna de les interfícies del joc estarà conformada per un Widget, el qual també incorpora un Blueprint.

- Actor: és la classe de tots aquells objectes que tenen un comportament associat (Blueprint) i una jerarquia d'objectes fills (Mesh, Area, Child Actor, etc.).

- Character: és un tipus d'Actor específic que té una Mesh i un Animation Blueprint per gestionar les animacions d'aquesta Mesh. Està pensat per ser un personatge del joc, el qual té una velocitat, una àrea de col·lisió i altres paràmetres específics dels personatges d'un joc.

- Animation Blueprint: és un tipus de Blueprint que té la funció específica de controlar i gestionar les animacions d'un Character amb un esquelet específic.

- Data Table: es tracta d'una col·lecció lineal estàtica d'un tipus concret de variables. Conceptualment, funciona com una taula estàtica de tipus personalitzable (Structure). Serveix per importar informació d'arxius ".json", ".csv" o similars.

- Structure: és l'estructura que es pot assignar a una Data Table . Per exemple, es pot crear una Structure conformada per un enter i un string, per després crear una Data.

- Game Instance: Blueprint que funciona com un Singleton i permet l'ús de variables globals.

6.2. Structures del joc

Per crear diferents variables globals, ha sigut necessari crear vàries Structures, aquestes han estat:

6.2.1. PlayerData

La Structure anomenada PlayerData serveix per guardar les variables dels personatges que siguin úniques per cada personatge (Veure Figures 43 i 44). Aquestes variables són:

- Player (Actor): Aquesta variable serveix per guardar-se una referència al mateix actor. Actualment no s'utilitza.

- Name (string): Nom del personatge jugable.

- ElementPred(string): Element predilecte del jugador, serveix per calcular el dany que rep dels atacs.

- Level(int): Variable que es guarda el nivell del personatge de cara als Spells que pot utilitzar.

- Spells(int array): Lista de 3 ints que representa la id dels Spells que pot fer servir el personatge. S'inicialitza com a [-1, -1, -1].

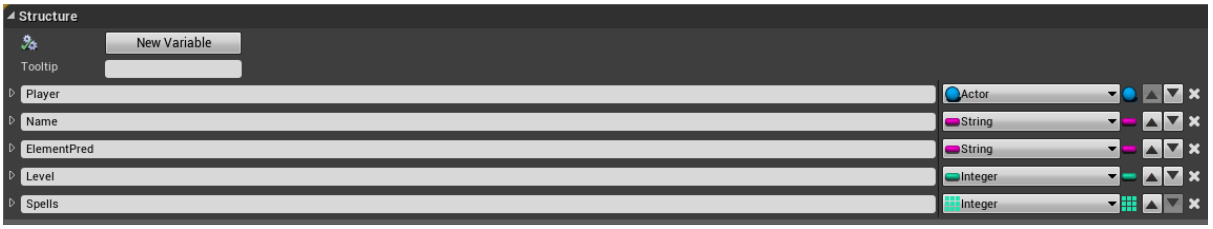


Figura 43: Imatge de les variables de la Structure PlayerData

Figura 44: Imatge de les variables inicialitzades de la Structure PlayerData

6.2.2. SpellData

La structure anomenada SpellData, s'utilitza per guardar la llista de tots els Spells que conté el joc i les variables que es guarden són les següents (Veure Figures 45 i 46):

- Name(string): Nom del Spell.

- Description(string): Descripció del Spell. Actualment no s'utilitza.

- Element(string): Element del Spell

- Damage_heal(int): dany que fa el Spell. Al ser un prototip no he implementat curació, però en un futur hi hauran habilitats curatives.

- MP(int): Cost per usar el Spell. Al ser un prototip no ha estat implementat.

- Level(int): Nivell que necessita el personatge per poder seleccionar el Spell.

- Type(string): Tipus de Spell del que es tracta (Impacte a terra, projectil, etc.).

- Image(Texture 2D): Imatge que representa el Spell en els menús i HUDs.

- Particula(Particle System): FX que es crea a l'utilitzar el Spell ingame.

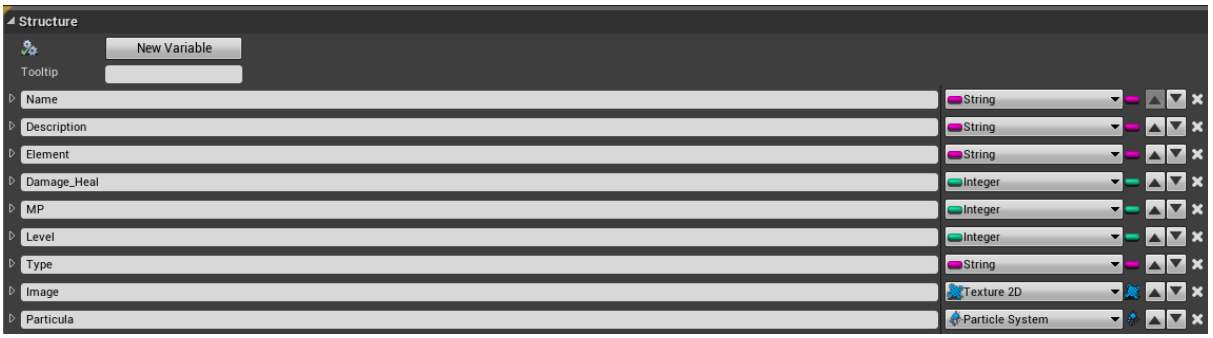


Figura 45: Imatge de les variables de la Structure SpellData

Figura 46: Imatge de les variables inicialitzades de la Structure SpellData

6.3. Variables globals i instància de joc

Per gestionar les variables de la partida, prenem la decisió d'utilitzar una nova Game Instance que cal crear i assignar al projecte (veure Figura 47).

Figura 47: Captura de les opcions del projecte d'Unreal on es pot veure la classe d'instància de joc utilitzada

Aquesta instància de joc, anomenada Stats_GameInstance, conté diverses variables llistades a continuació (veure Figura 48):

- NumPlayers(int array): Llista dels personatges seleccionats, guardats per id.

- ToralPlayers(int): nombre total de jugadors que es poden tenir.

- ActualPlayers(int): nombre de jugadors guardats a la Structure.

- SpellsData(SpellData array): llista de tots els Spells que es poden utilitzar en el projecte.

- Players(PlayerData array): llista de tots els personatges que hi ha actualment en el joc.

- Elements(string array): llista de tots els elements que hi ha en el joc.

- EditingPlayerId(int): id del personatge seleccionat.

- LevelsImg(Texture 2D array): llista de les possibles imatges que es mostren depenent del nivell.

- ElementsImg(Texture 2D array): llista de les imatges dels elements.

- NoSpellImage(Texture 2D): imatge que apareix quan no hi ha un Spell seleccionat.

Figura 48: Captura de pantalla de les variables de la instància de joc

6.4. Gestor dels personatges

Al tractar-se d'un videojoc de tàctica per torns, el jugador no controla a un únic personatge, sinó que hi ha múltiples. A causa d'això ha estat necessari implementar un Actor que gestionés a quin personatge controla el jugador en cada moment. L'Actor implementat s'anomena "TopDownCharacter" (veure Figura 49).

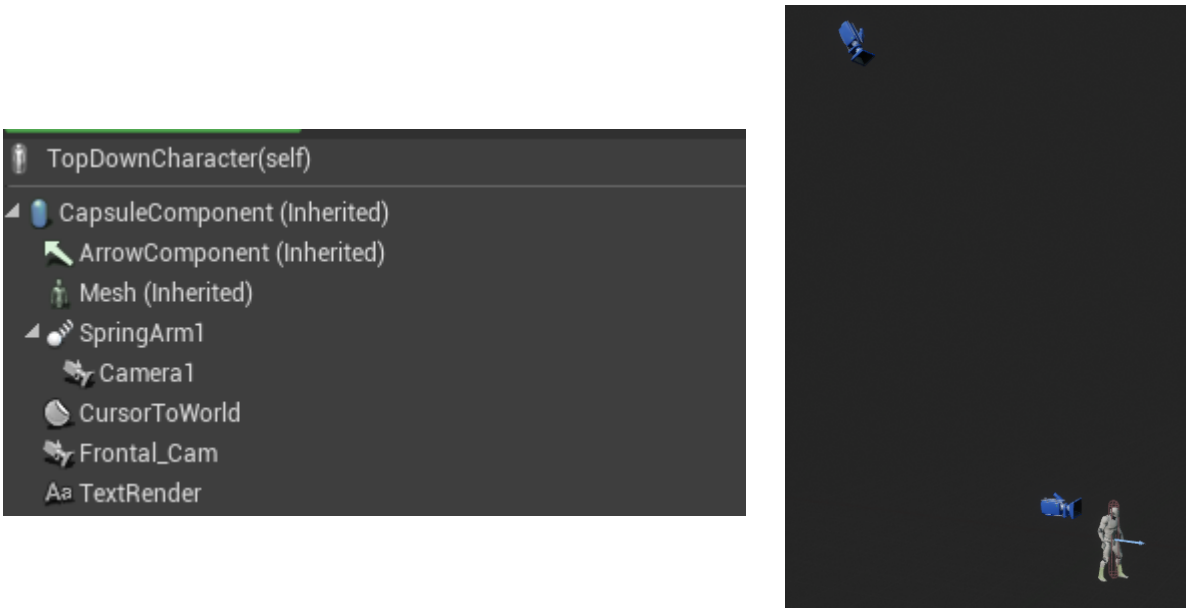


Figura 49: Captura del viewport de l'actor "TopDownCharacter" juntament amb els seus components

Dels components mostrats a les imatges anteriors, cal fixar-se en les dues càmeres, la càmera 1 es tracta de la càmera llunyana, i la càmera Frontal de la càmera que es veu l'apuntar. Algunes de les configuracions més importants canviades en aquest blueprint és l'apartat de Pawn. Veure Figura 50:

- En l'apartat pawn, és necessari que la variable "Auto posses AI" estigui establert com "Placed in world", d'aquesta manera el jugador controlarà a un personatge en iniciar la partida.

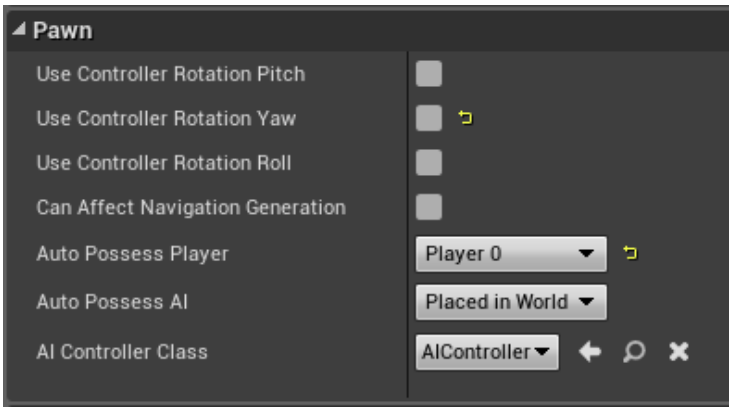


Figura 50: Imatge de la secció de la configuració de l'actor "TopDownCharacter " representant a l'apartat del Pawn

Passant a les funcions d'aquest actor, disposa d'una gran varietat i durant tot aquest capítol, anirem explicant a quines funcionalitats responen tots aquests Events, Functions i variable. Veure Figures 51 i 52.

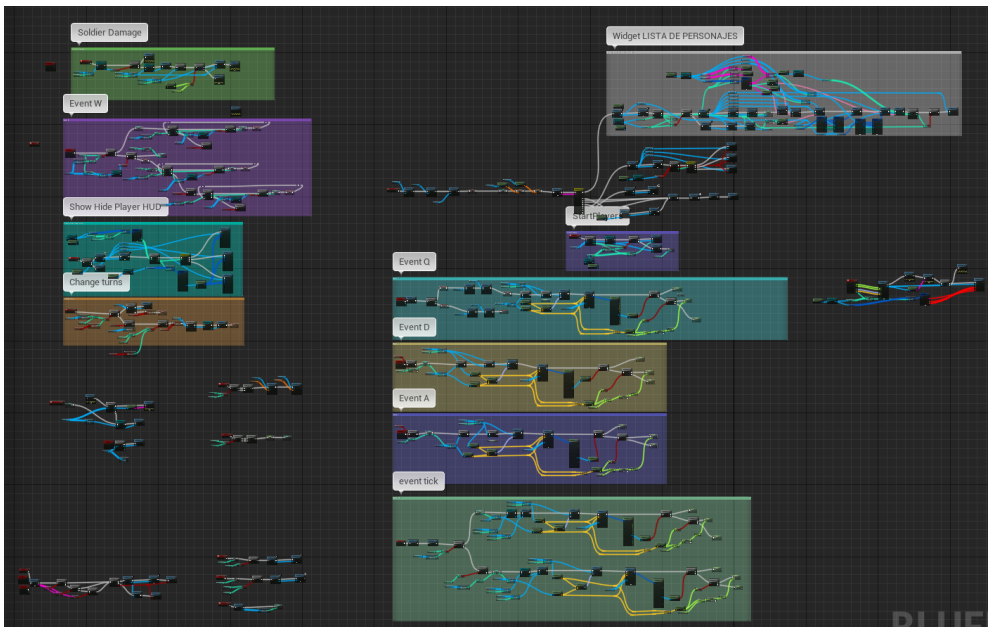


Figura 51: Captura Blueprint TopDownCharacter

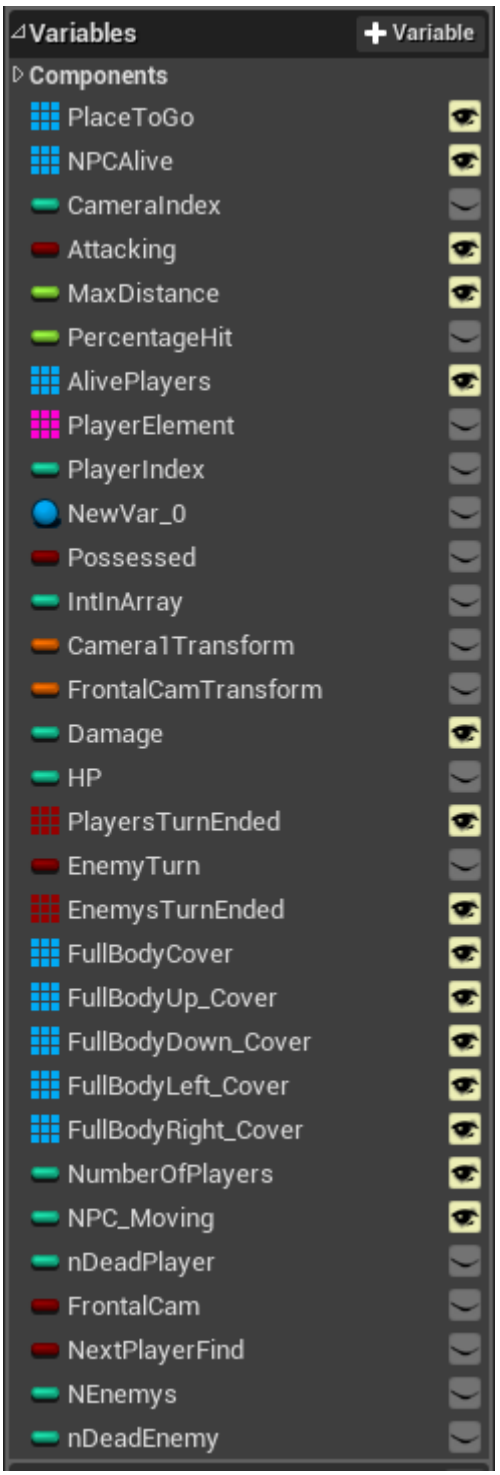


Figura 52: Captura variables TopDownCharacter

Aquestes variables són totes d'aquest actor, tot i que la majoria s'utilitzin en altres Blueprints. Si cal destacar algunes, aquestes serien:

- PlaceToGo(Actor array): llista de tots els actors que són de la classe PlaceToStop.

- NPCAlive(Actor array): llista de tots els enemics vius.

- AlivePlayers(Array actor): llista de tots els personatges.

- FullBodyUp_Cover/FullBodyDown_Cover/FullBodyLeft_Cover/ FullBodyRight_Cover(Actor array): llistes que es guarden els actors que formen part d'una de les quatre cobertures.

- NPC_Moving(int): el seu valor és l'índex de l'array de NPCAlive que representa l'actor que està movent-se.

6.4.1. Inicialització

L'Event BeginPlay (veure Figura 53) s'encarrega d'inicialitzar algunes variables i elements importants:

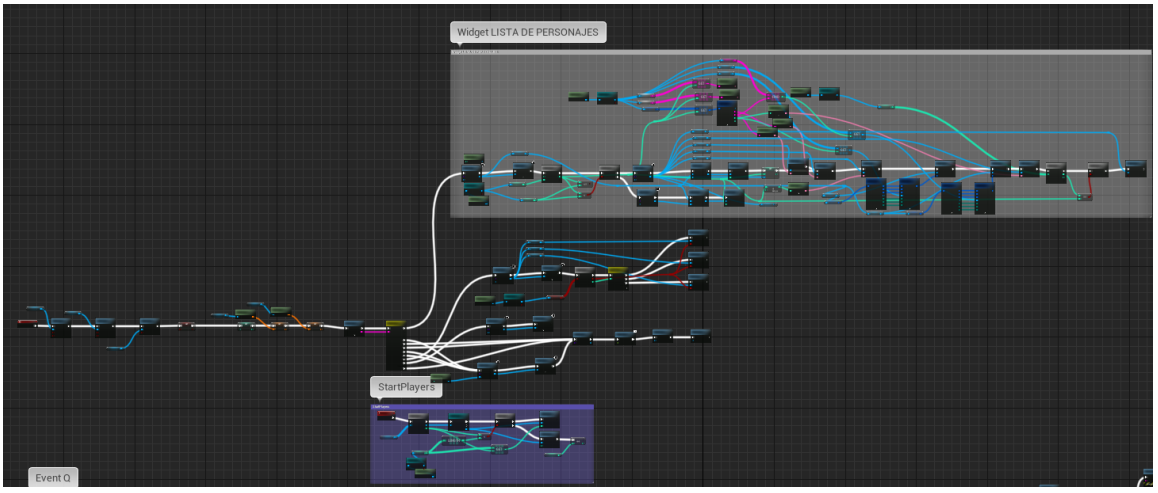


Figura 53: Captura de la funció BeginPlay

- Amaga completament el personatge del TopDownCharacter. Veure Figura 54.

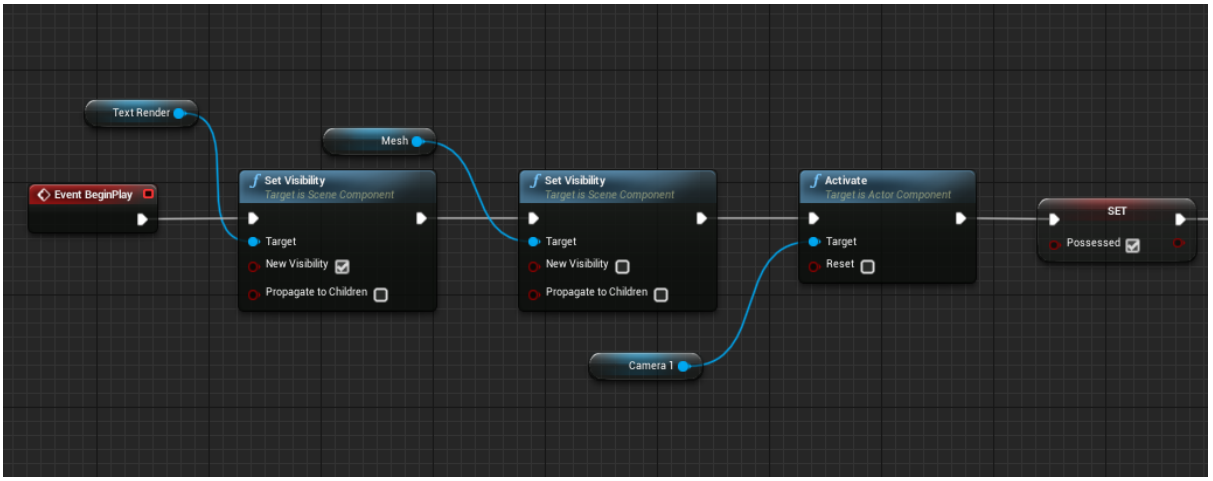


Figura 54: Captura de la funció BeginPlay 1

- Inicialitza les variables necessàries. Veure Figura 55.

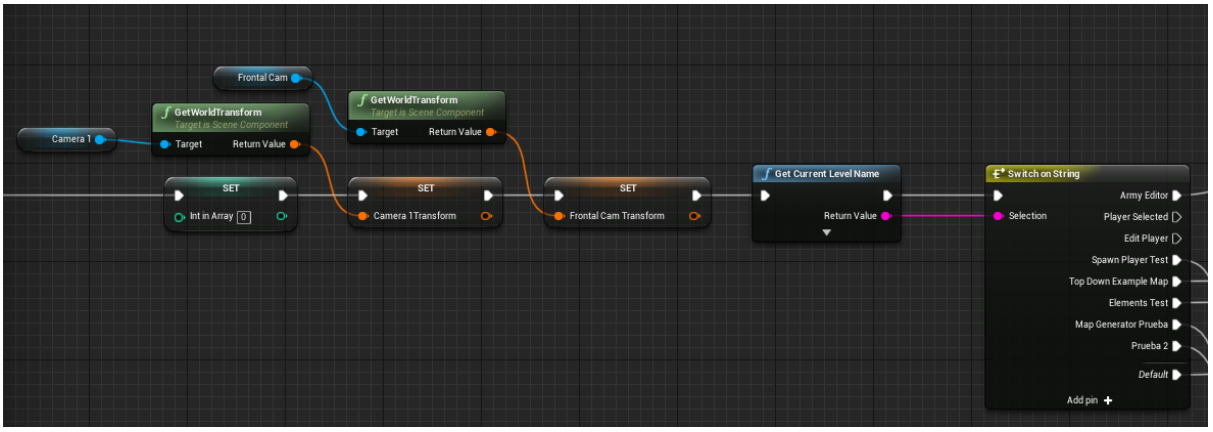


Figura 55: Captura de la funció BeginPlay 2

- Depenent del nivell, mostra un widget o un altre. Veure Figures 56 i 57.

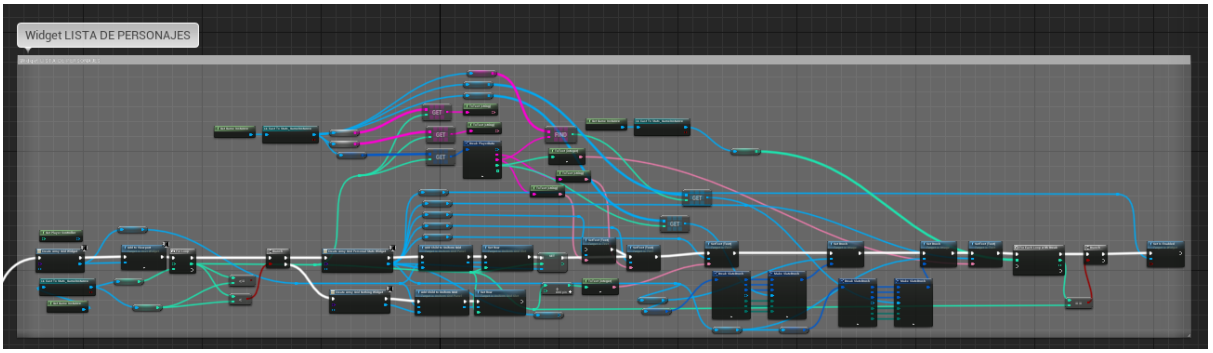


Figura 56: Captura de la funció BeginPlay 4 (Aquesta funció s'explicara a l'apartat de la implementació dels widgets 6.9.4)

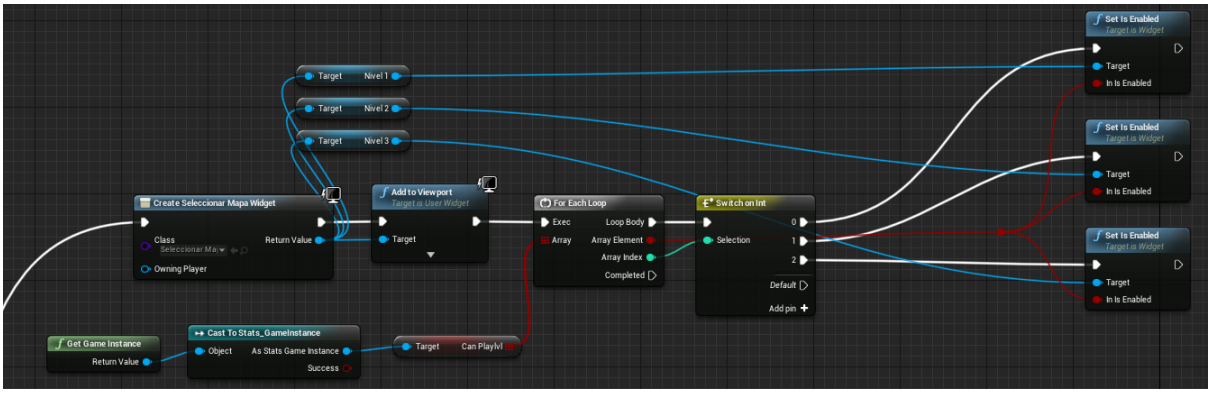


Figura 57: Captura de la funció BeginPlay 5 (Aquesta funció s'explicara a l'apartat de la implementació dels widgets 6.9.6)

- Crea els personatges. Veure Figura 58.

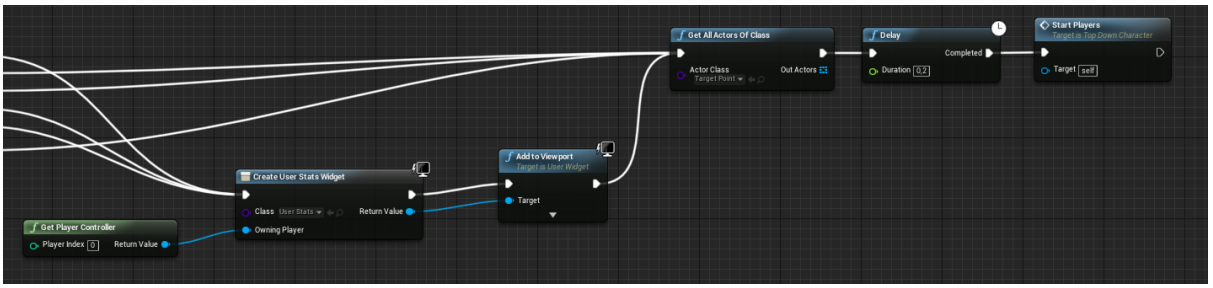


Figura 58: Captura de la funció BeginPlay 3

6.4.2. Event Tick

En Aquest Actor, el Event Tick (veure Figura 59) és un dels events més importants. A causa que és la funció encarregada de gestionar les càmeres mentre apuntes a un enemic. Aquesta funció s'encarrega de:

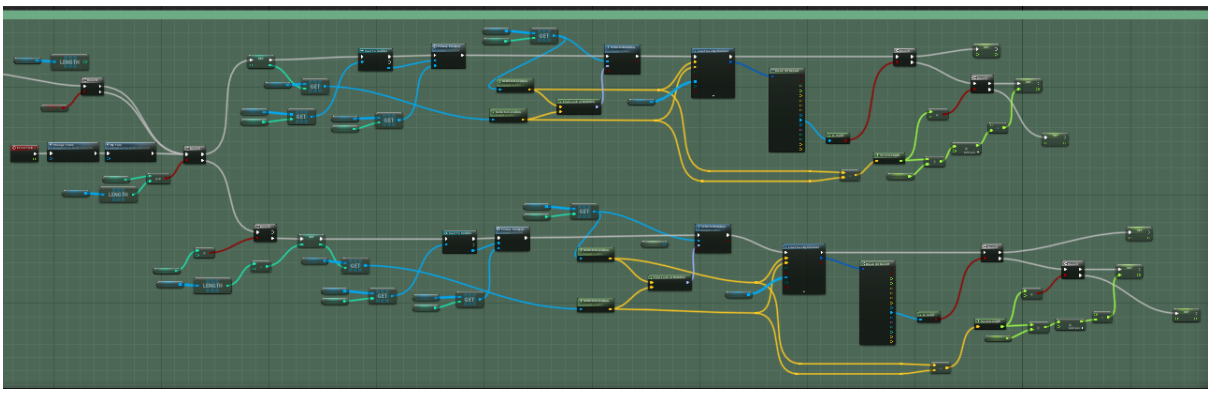


Figura 59: Captura de la funció Tick

- Canviar de torns si tots els personatges d'un equip han realitzat totes les accions. Veure Figura 60.

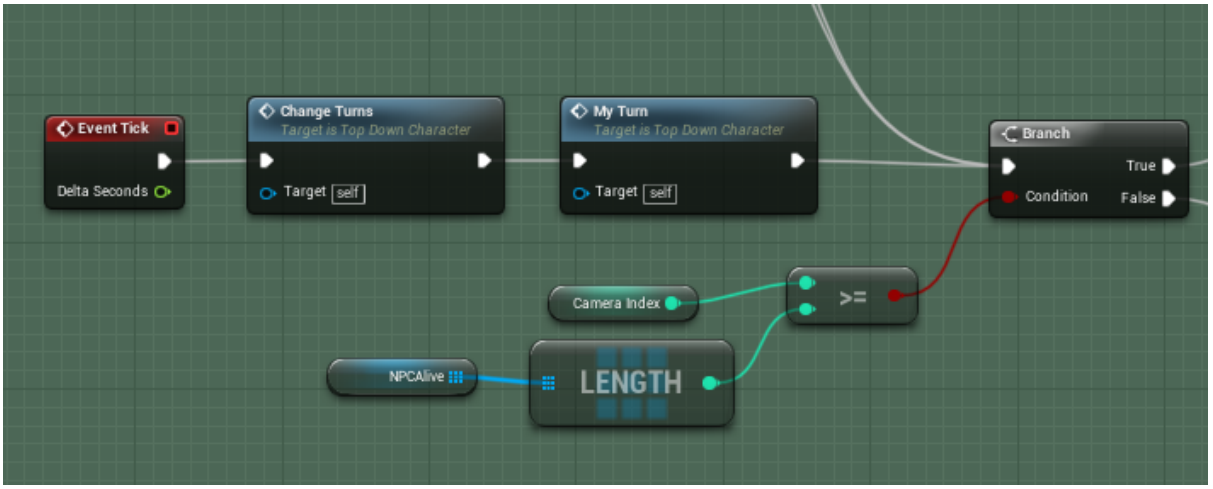


Figura 60: Captura de la funció Tick 1

- Detectar a quin personatge està mirant el jugador i detectar la probabilitat d'encertar un atac. Veure Figures 61-64.

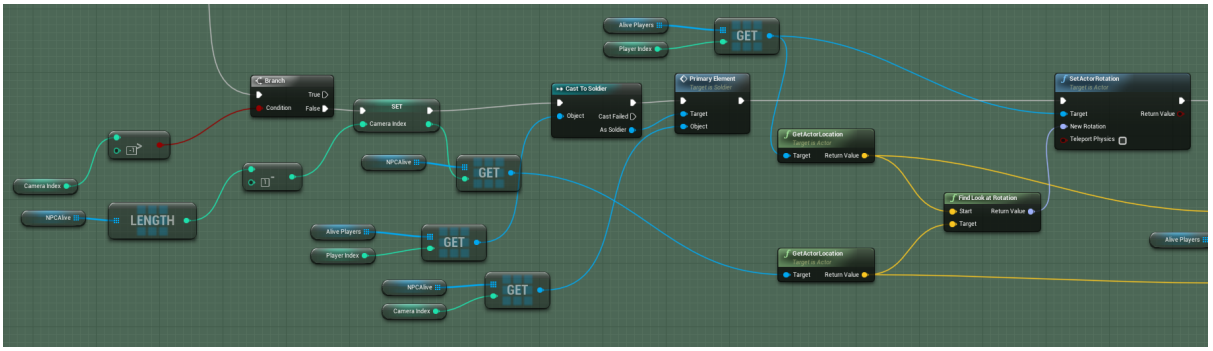


Figura 61: Captura de la funció Tick 2

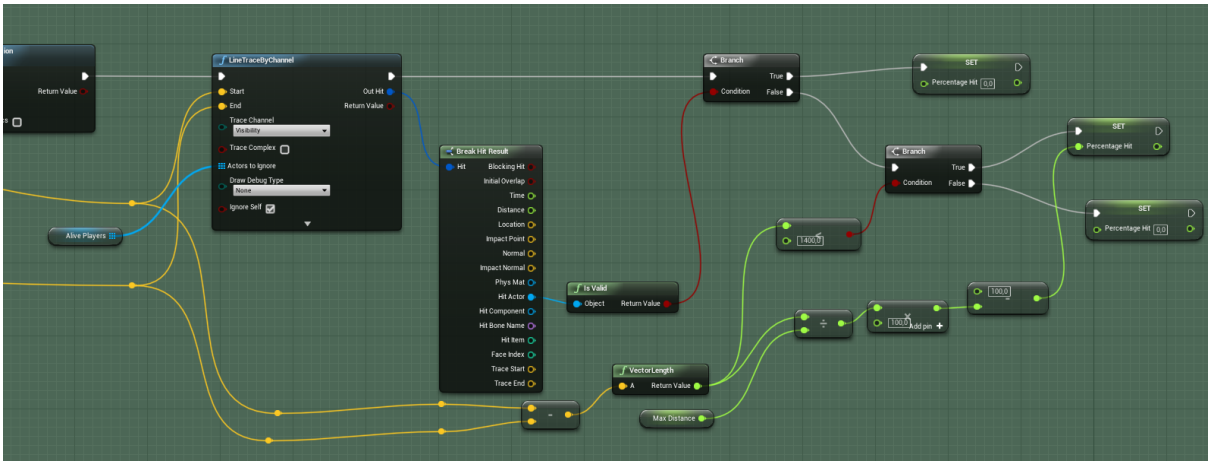


Figura 62: Captura de la funció Tick 3

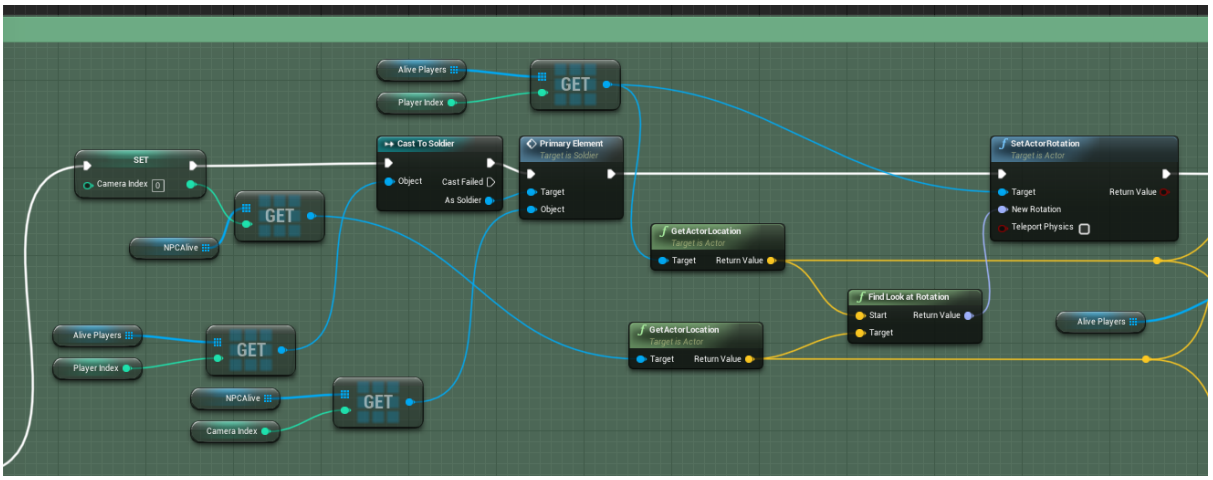


Figura 63: Captura de la funció Tick 4

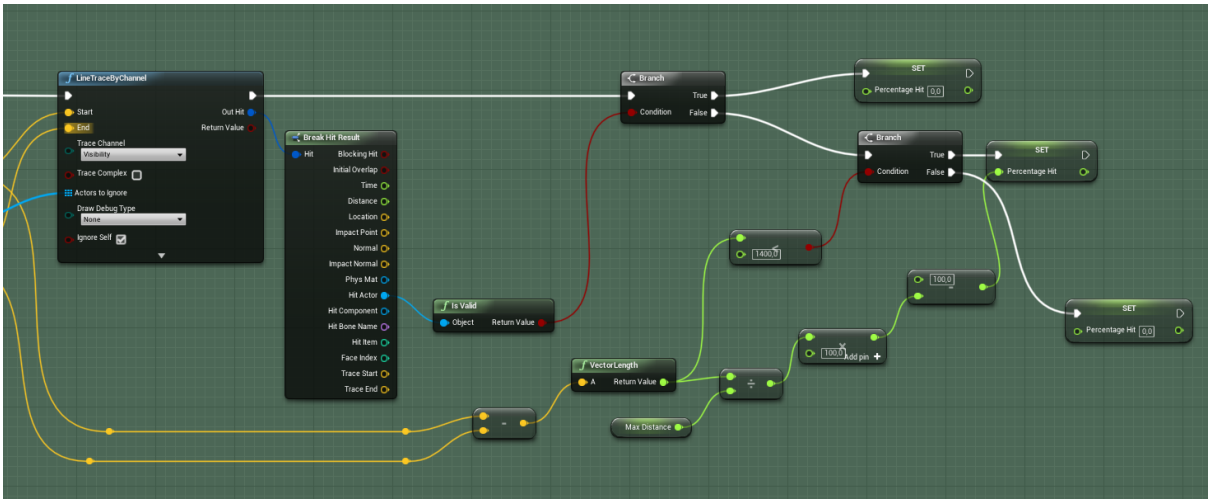


Figura 64: Captura de la funció Tick 5

6.4.3. StartPlayer

Per la creació de personatges, es crida la funció anomenada StartPlayers. Aquesta funció li passa a l'actor del personatge l'id del personatge que són, i en el cas de no haver-hi més, es passa un -1 com a id. Veure Figura 65.

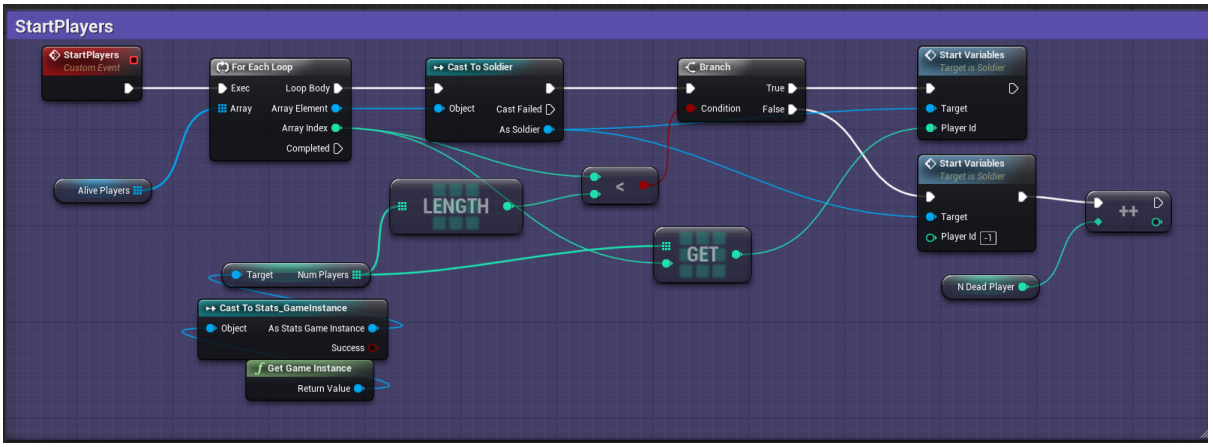


Figura 65: Captura de la funció StartPlayers

6.4.4. Event Q

L'event Q (veure Figura 66), gestiona com actua l'actor en pressionar la lletra Q del teclat. Aquesta funció serveix per:

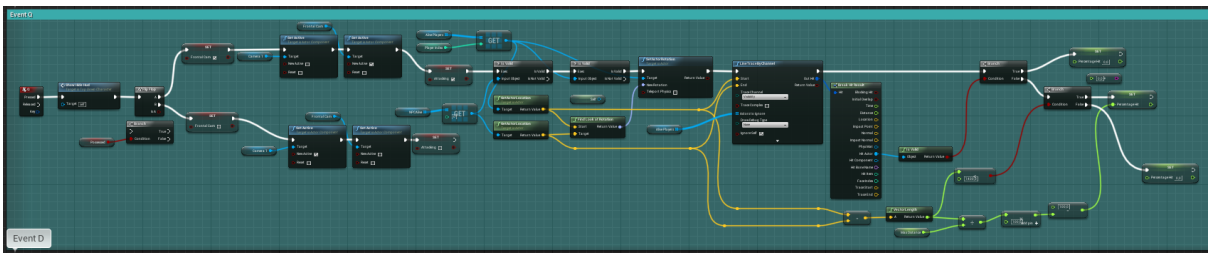


Figura 66: Captura de la funció Event Q

- Apuntar i deixar d'apuntar.

- Mostrar el HUD del personatge i canviar de càmera. Veure Figura 67

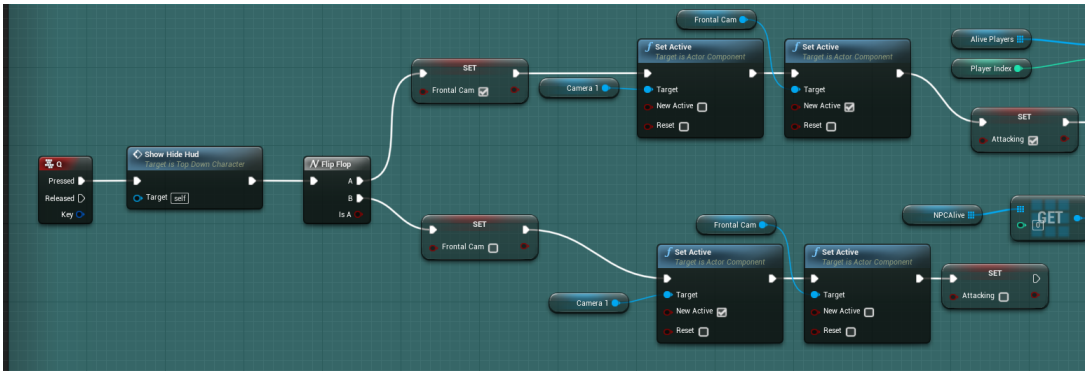


Figura 67: Captura de la funció Event Q 1

- Detectar el percentatge d'encertar. Veure Figures 68 i 69.

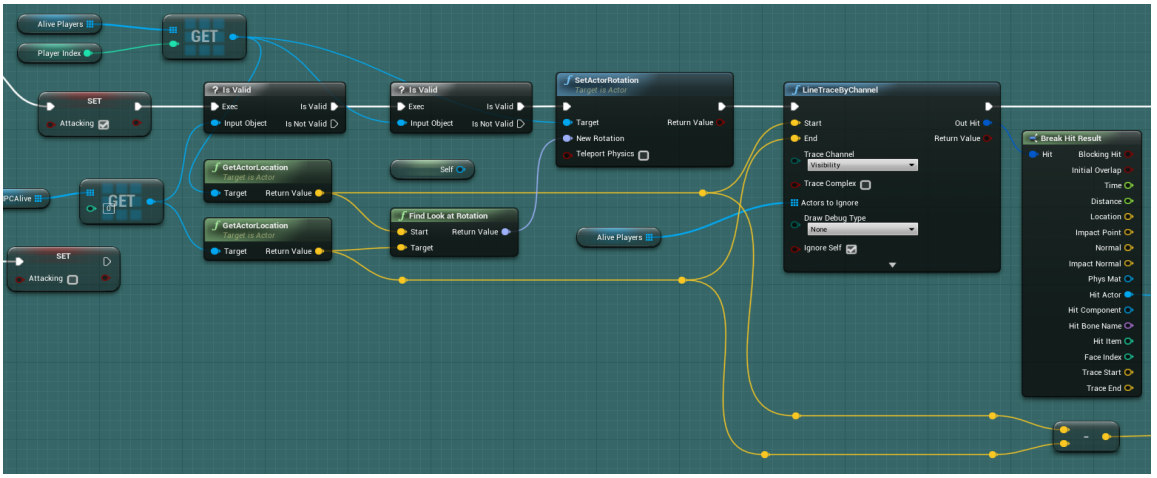


Figura 68: Captura de la funció Event Q 2

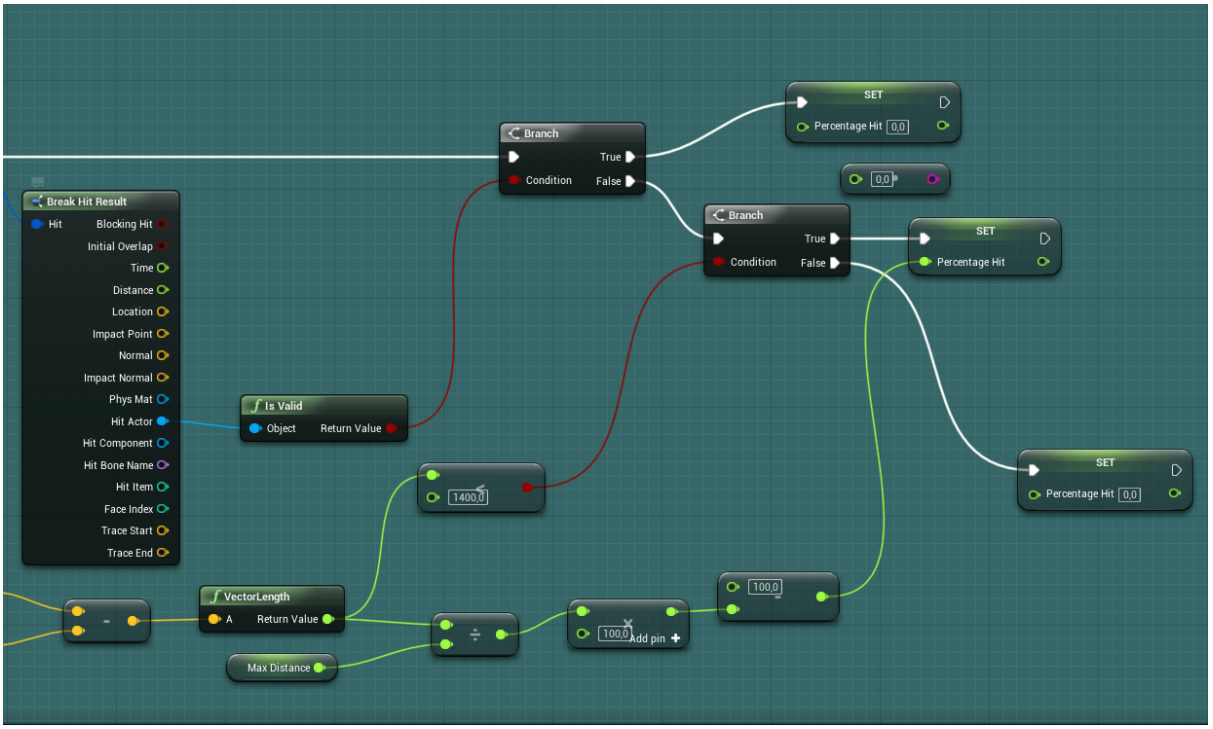


Figura 69: Captura de la funció Event Q 3

6.4.5. Event D / Event A

Aquests events (veure Figura 70 i 71), gestionen com actua l'actor en pressionar la lletra D / A del teclat. Aquesta funció serveix per:

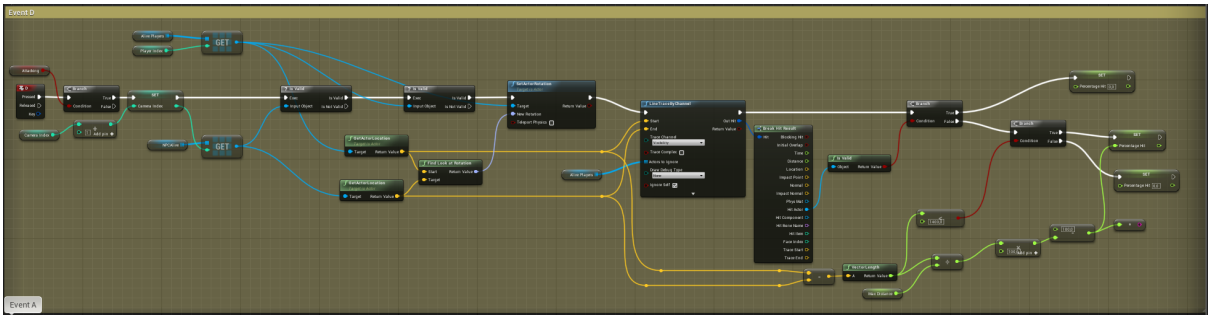


Figura 70: Captura de la funció Event D

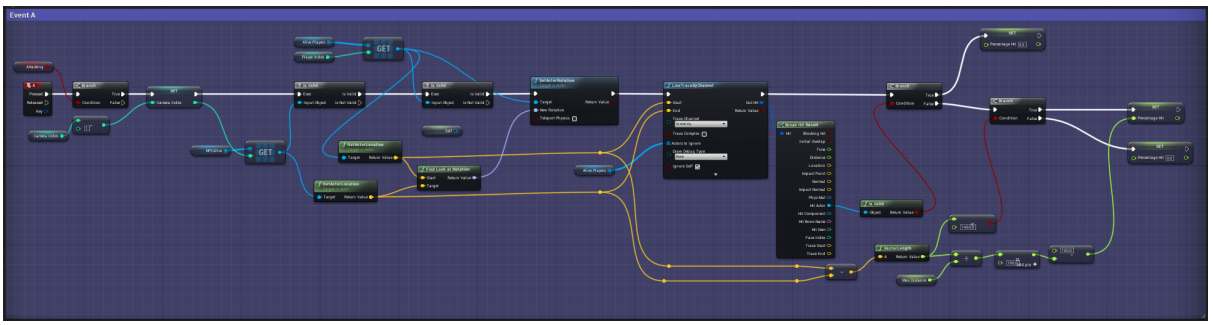


Figura 71: Captura de la funció Event A

- Canviar l'objectiu de l'atac.

- Rotar l'actor per mirar a l'objectiu. Veure Figures 72 i 73.

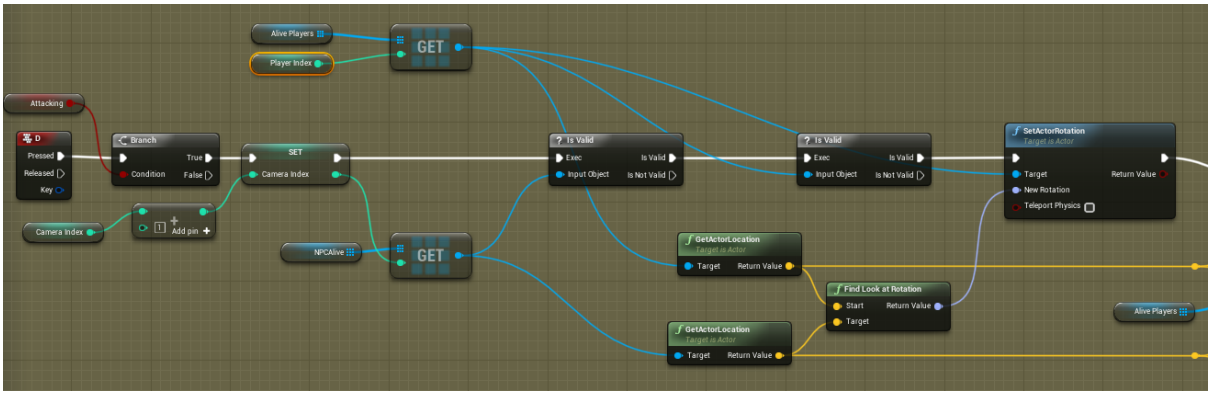


Figura 72: Captura de la funció Event D 1

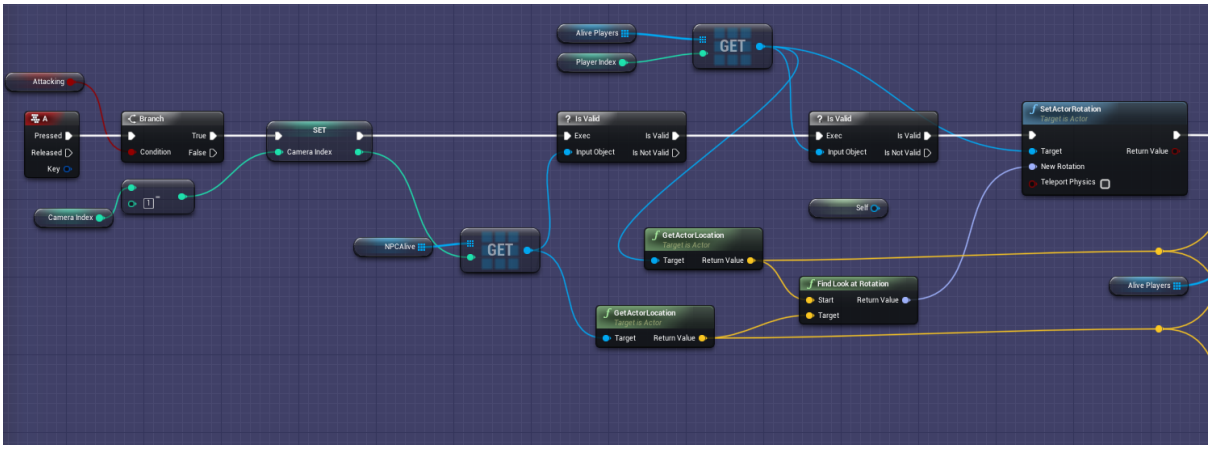


Figura 73: Captura de la funció Event A 1

- Recalcular el percentatge d'encertar l'atac. Veure Figures 74 i 75.

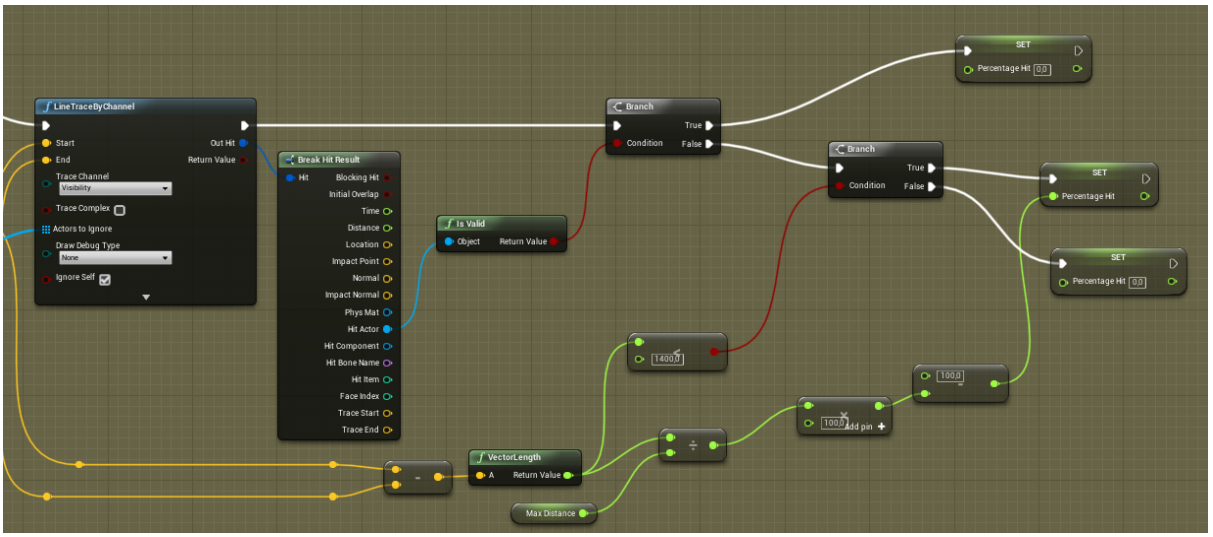


Figura 74: Captura de la funció Event D 2

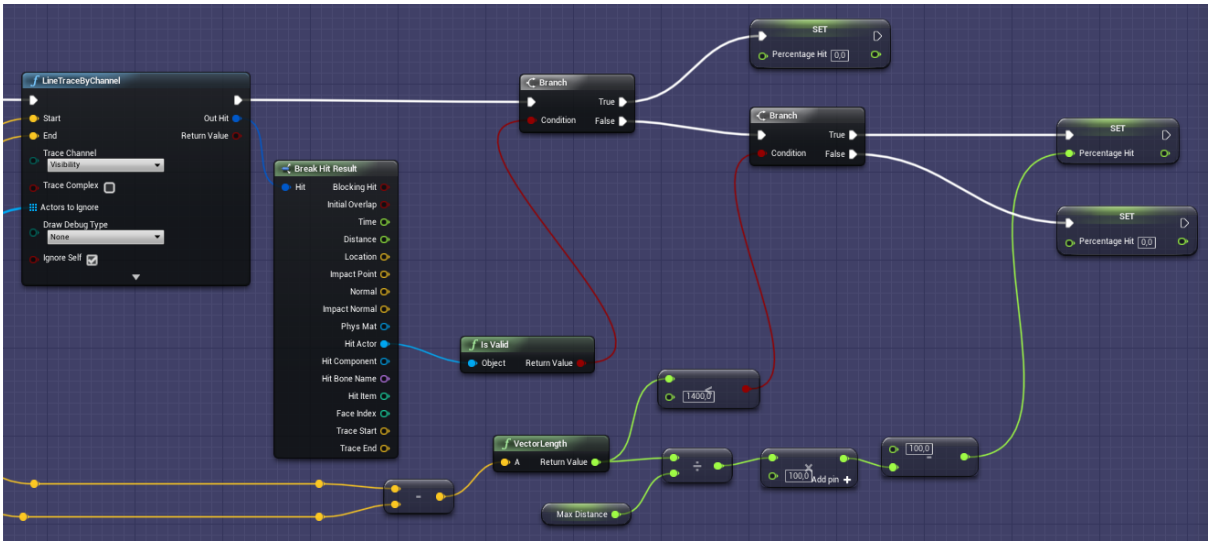


Figura 75: Captura de la funció Event A 2

6.4.6. Event W

L'event W, gestiona com actua l'actor en pressionar la lletra W del teclat. Aquesta funció (veure Figures 76-79) fa el següent:

- Cambiar el personatge que controles al següent personatge viu.

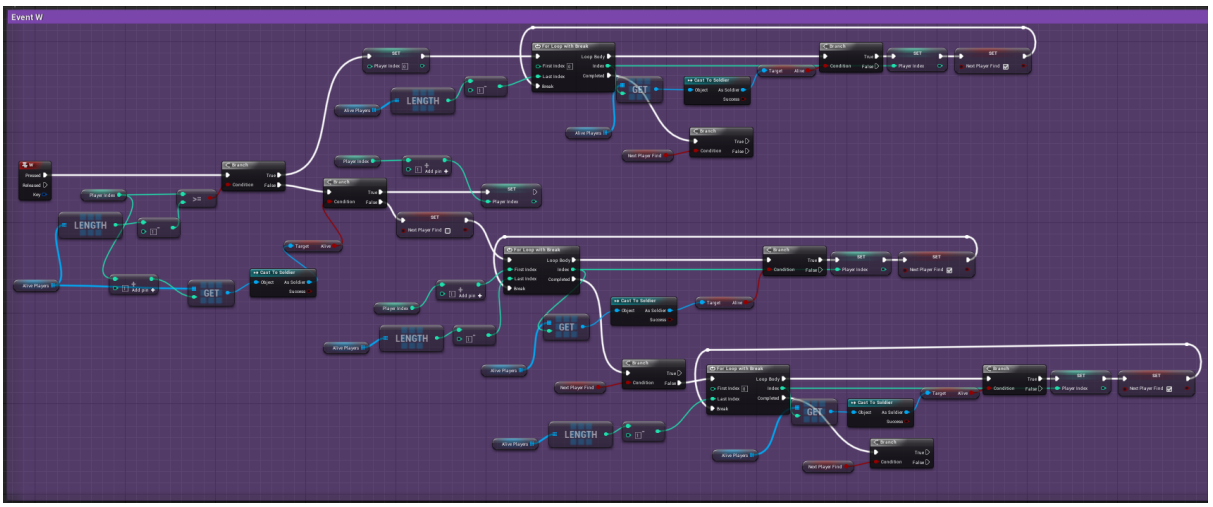


Figura 76: Captura de la funció Event W

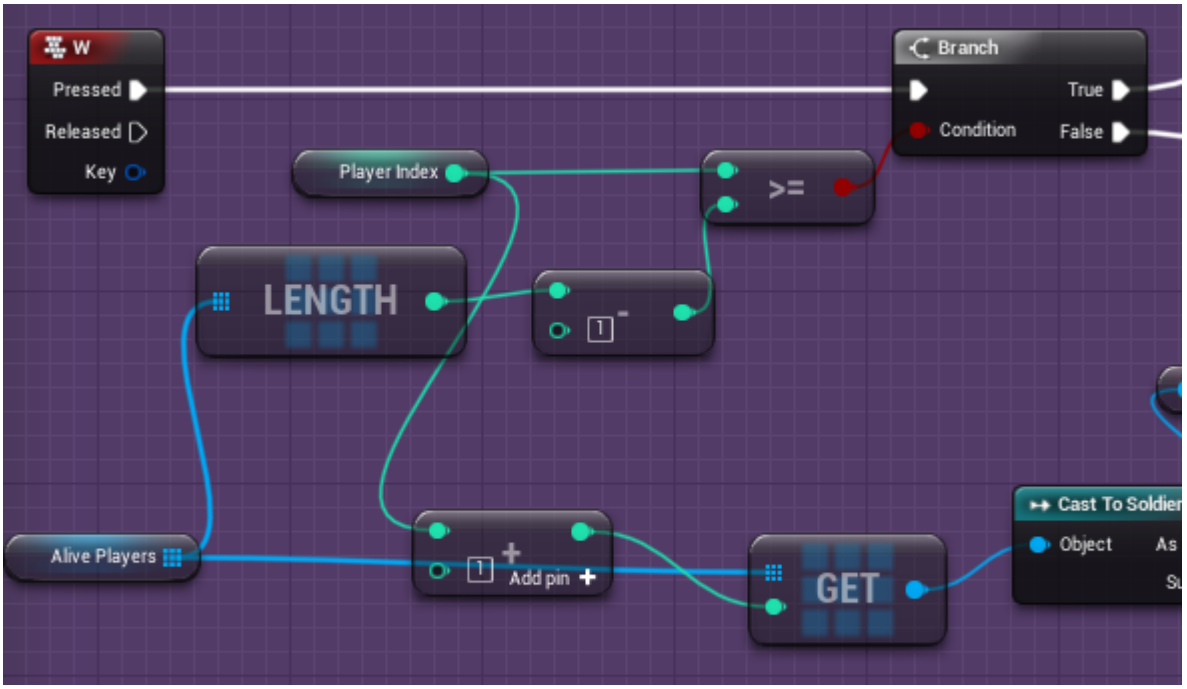


Figura 77: Captura de la funció Event W 1

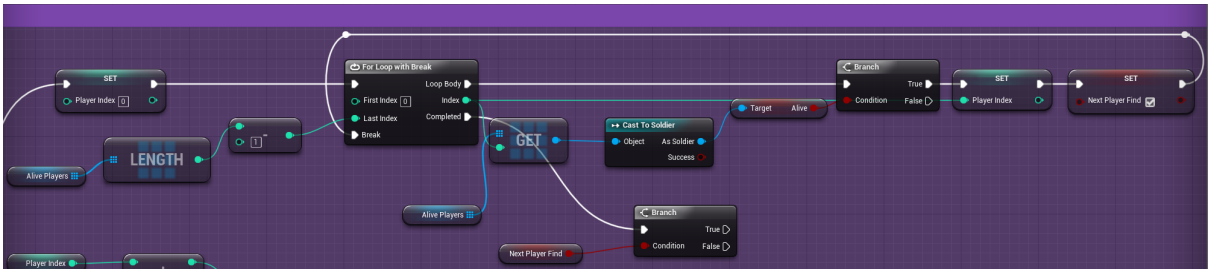


Figura 78: Captura de la funció Event W 2

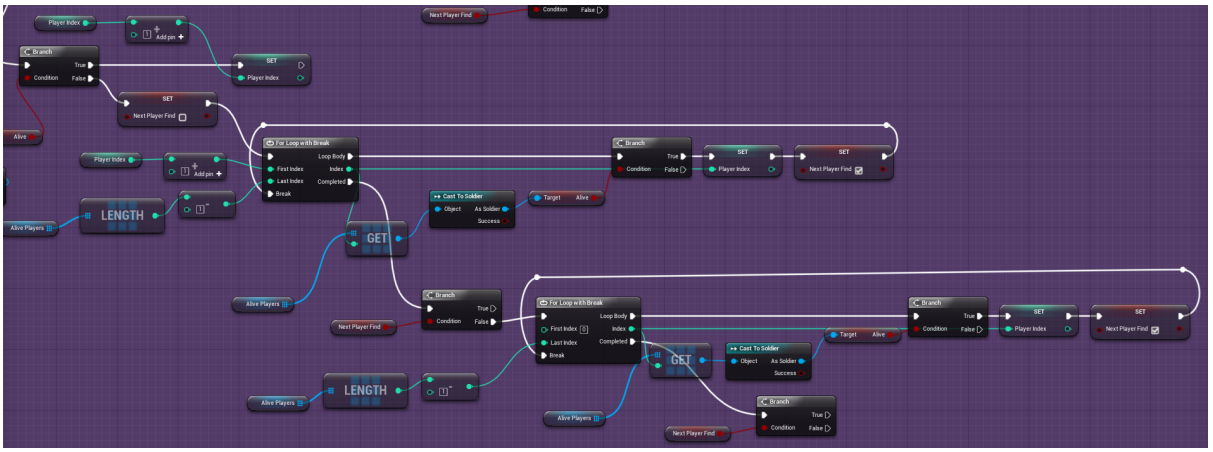


Figura 79: Captura de la funció Event W 3

6.4.7. ShowHideHUD

Aquests event serveix per mostrar o eliminar el HUD que es mostra quan el jugador apunta a un enemic. Aquesta funció (veure Figura 80) fa el següent:

- Crea o elimina el HUD utilitzant un flipflop.
- En cas d'eliminar-lo l'elimina del pare.
- En cas de crear-lo:
    - Crea el widget "UserStats" i el mostra al viewport.

- Per cada botó:

    - Busca els Spells del personatge amb el mateix índex al SpellData i agafa la imatge.

    - Modifica el style del botó d'acord amb la imatge obtinguda.

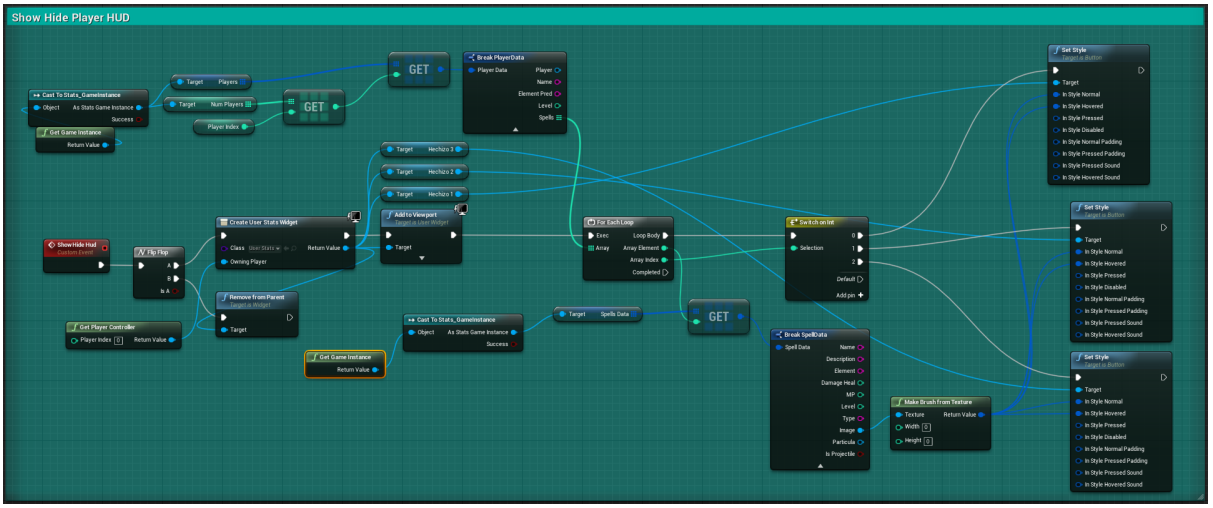


Figura 80: Captura de la funció ShowHideHUD

6.4.8. ChangeTurns

Aquests event (veure Figura 81), serveix per canviar de torn i fa el següent:

- Mira si tots els jugadors han acabat les accions.

    - Si les han acabat, per cada enemic viu, fa un reset a les seves variables que depenen dels torns i buida el boolean array PlayerTurnEnded.

    - Si no ha acabat mira si és el torn de l'enemic.

- Si ho es, buida el boolean array de EnemysTurnEnded i per cada personatge viu, fa un reset a les seves variables que depenen dels torns.

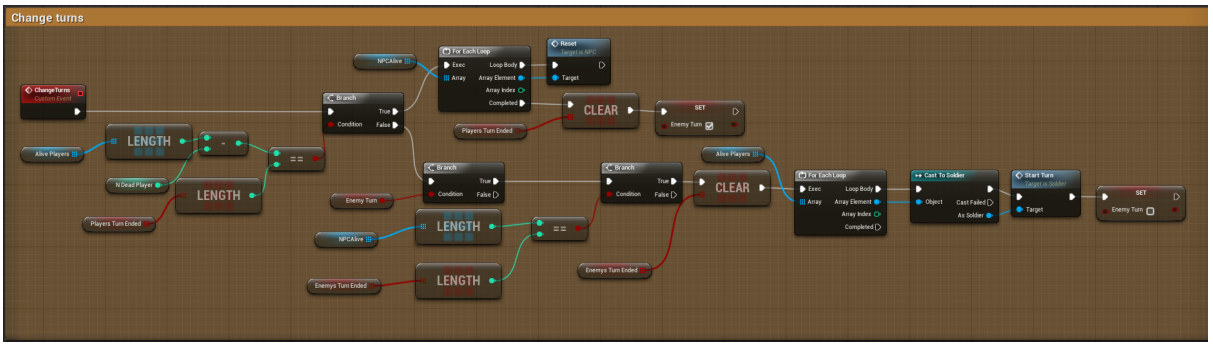


Figura 81: Captura de la funció ChangeTurns

6.4.9. SoldierAttack

Aquest event (veure Figura 82), es crida quan el jugador fa l'acció d'atacar, i fa el següent:

- Revisa si aquest personatge pot fer més accions aquest torn.
- Si encara li queden, li suma una acció.
- Revisa que l'objectiu existeix.
- Genera un float random del 0 al 100.
- Si el percentatge d'encertar és superior al número generat, es crida la funció que gestiona el mal que rep l'enemic.

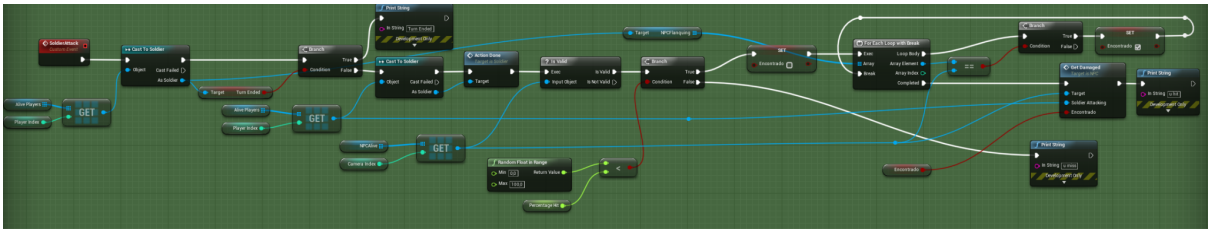


Figura 82: Captura de la funció de la SoldierAttack

6.4.10. PlayerDead / EnemyDead

Aquestes dues funcions (veure Figures 83 i 84) serveixen per gestionar les morts dels personatges / enemics, i fa el següent:

- Incrementar el nombre de morts dels personatges / enemics.
- Comprovar si tots estan morts.
- Si és el cas, es mostra per pantalla el widget pertinent.

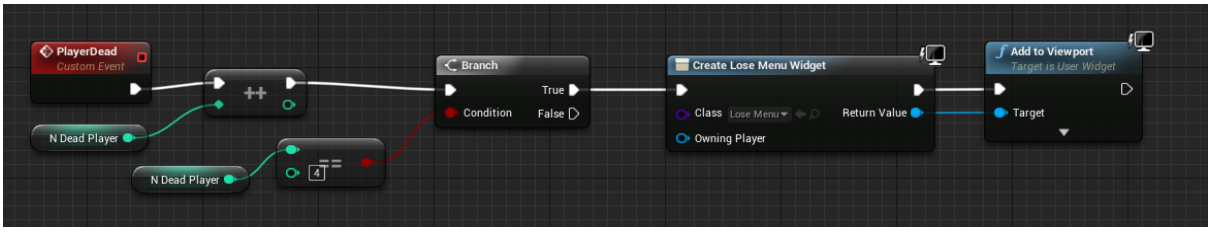


Figura 83: Captura de la funció de la PlayerDead

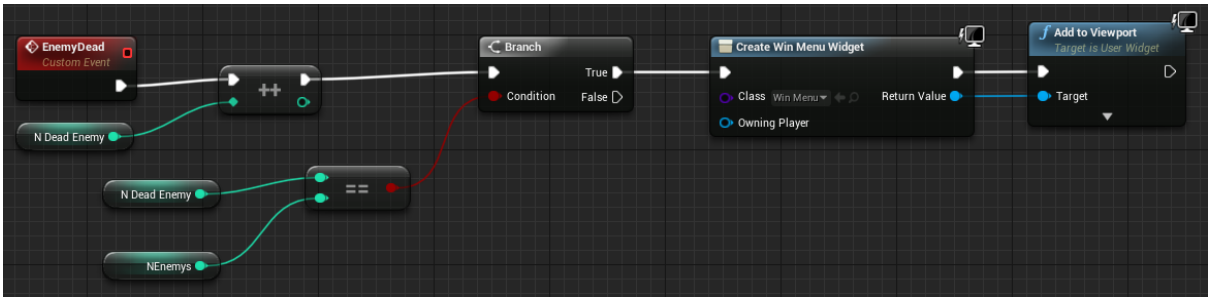


Figura 84: Captura de la funció de la EnemyDead

6.4.11. AddPlayer / QuitPlayer

Aquestes dues funcions (veure Figues 85 i 86) es criden en el menú d'editar l'equip, i afegeixen o eliminen un personatge.

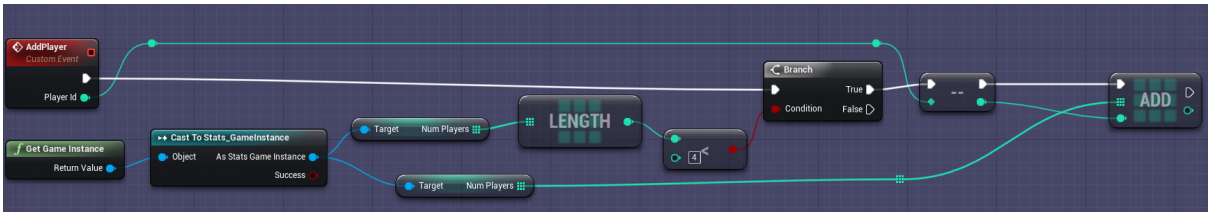


Figura 85: Captura de la funció AddPlayer

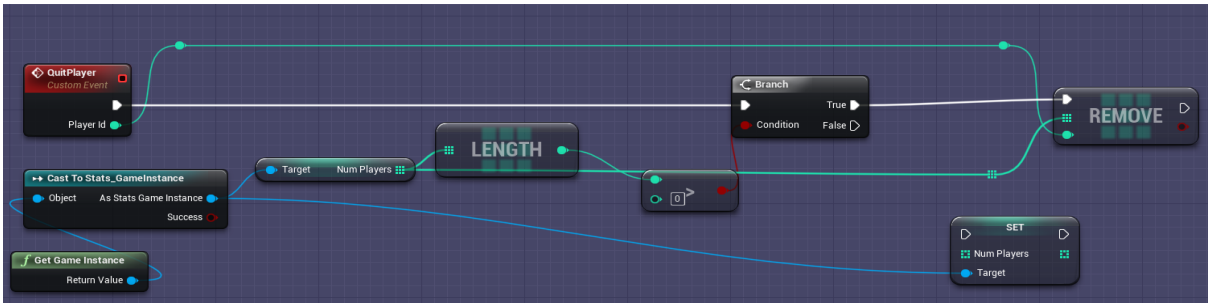


Figura 86: Captura de la funció QuitPlayer

6.4.11. Pause

Aquesta funció (veure Figura 87) serveix per saber quan mostrar el menú de pausa, i fa el següent:

- Detecta l'input del jugador.
- Mira si el nivell és un dels nivells ingame.
- Si ho és, mostra per pantalla / elimina el menú.

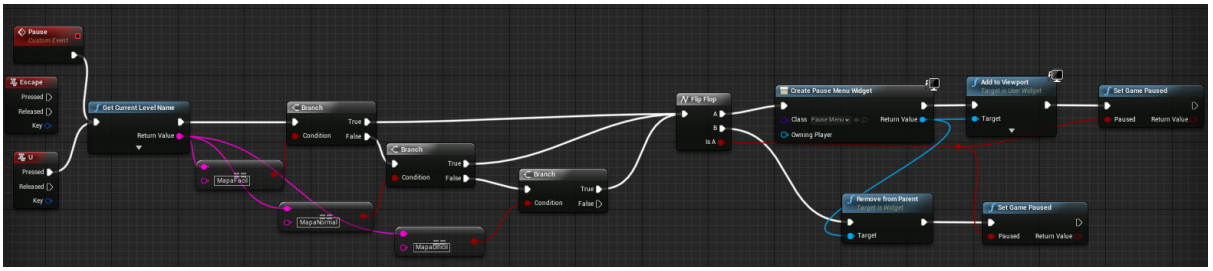


Figura 87: Captura de la funció Pause

6.5. Implementació moviment

A causa del gènere del combat i basant-me en els jocs mencionats en el apartat 2.5.1, el moviment ha estat un sistema quadricular a terra a partir d'un únic Actor (veure Figura 88) que fes el següent:

- Detectar si el jugador està prou a prop per poder clicar la casella.

- Afegir cada Actor d'aquesta classe larray d'àctors PlaceToGo.

- Moure el jugador a la casella clicada si està a prop.

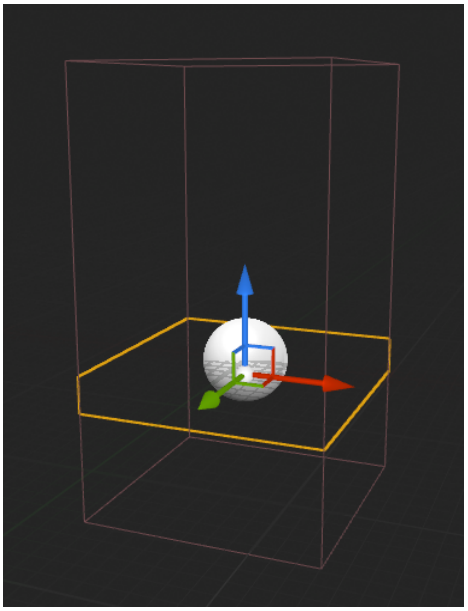


Figura 88: Captura del Actor PlaceToStop en el viewport

6.5.1. Inicialització

Aquesta funció (veure Figura 89) fa el següent:

- Crida a la funció CanGo.

- Afegeix l'actor a l'array d'actors del gestor anomenada PlaceToGo.

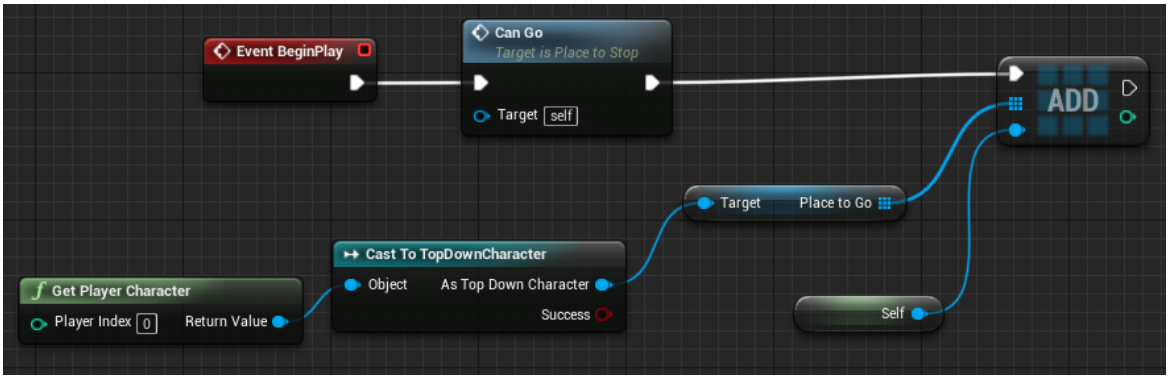


Figura 89: Captura de la funció BeginPlay

6.5.2. Event Tick / CanGo

Les funcions Tick (veure Figura 90) i CanGo (veure Figura 91) s'agrupen en aquest actor, i fan el següent:

- L'event Tick crida a la funció CanGo.

- La funció CanGo mira si existeix l'actor del personatge i si està viu. Després li posa un material o un altre a l'actor actual depenent de la distància.

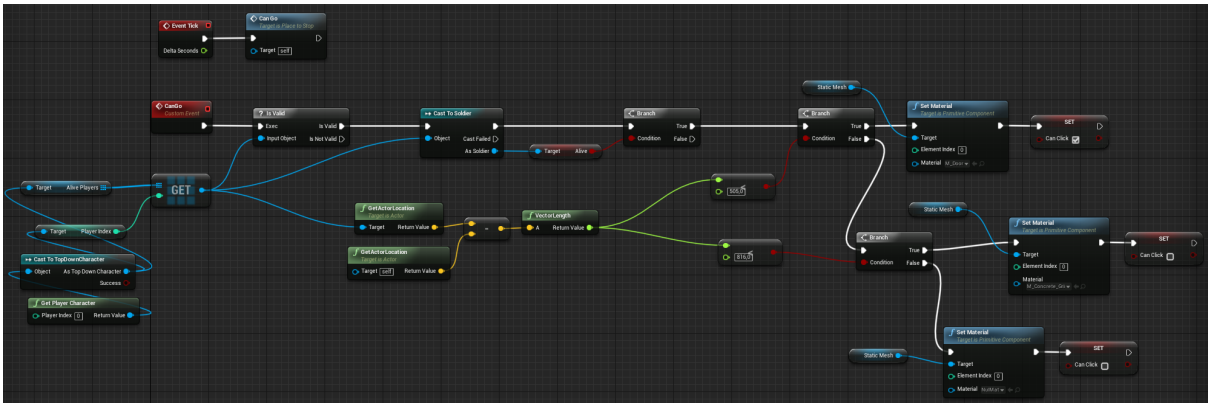


Figura 90: Captura de la funció Tick / CanGo

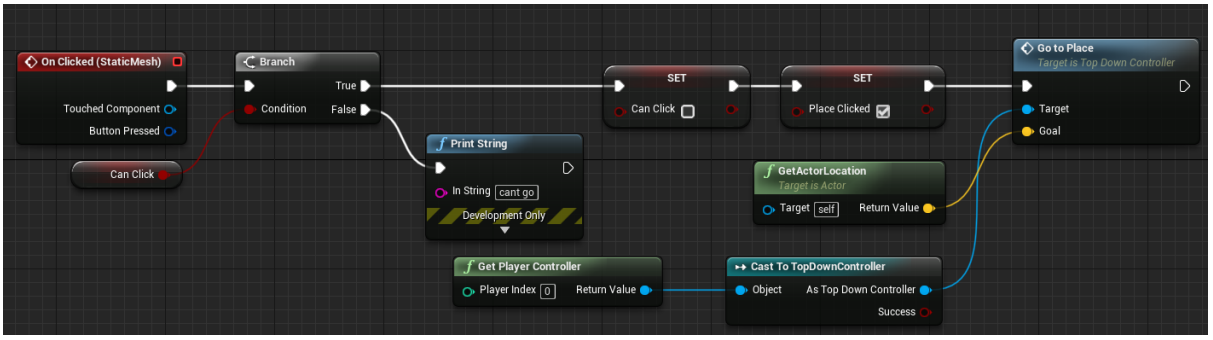


Figura 91: Captura de la funció On Clicked

6.5.3. OnClicked

Com es pot veure en la funció On Clicked (veure Figura 92), la forma trobada per aconseguir que els personatges es moguessin a on el jugador clica, va ser afegint una funció (veure figura 92) en l'actor CharacterController i cridar-la. Aquesta funció crida el node "Simple Move to Location" en el cas que sigui el nostre torn.

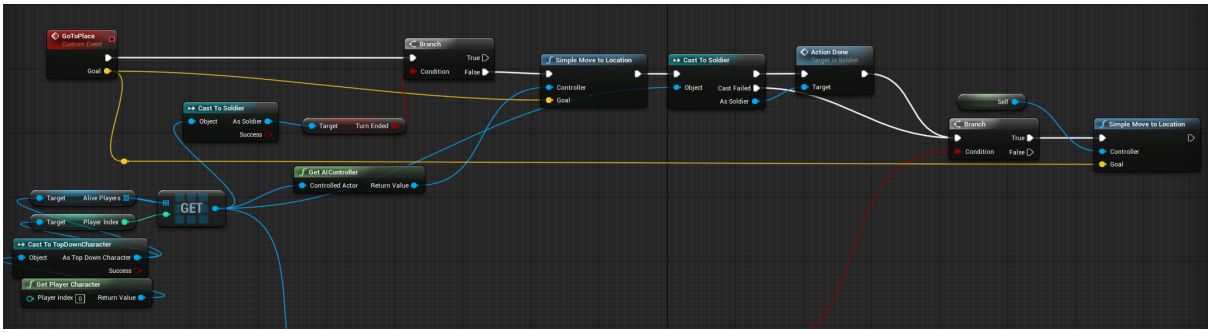


Figura 92: Captura de la funció GoToPlace

6.6. Implementació cobertures

El sistema de cobertures serveix per detectar si algun dels personatges o enemics es troben coberts i en quina de les 4 direccions està cobert (Top/Down/Right/Left). Aquest sistema consta de diferents actors:

6.6.1. Cover Actor

Aquest Actor (veure Figura 93) representa cada cel·la veïna d'una cobertura i té diferents funcions.

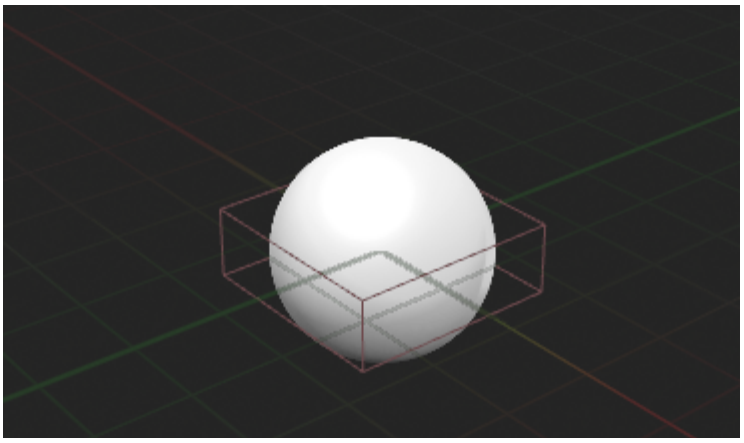


Figura 93: Captura del viewport de l'actor Cover

6.6.1.1. Inicialització

Aquesta funció (veure Figura 94) és la més senzilla, ja que només ens guardem aquest Actor l'array de FullBodyCover del Actor TopDownCharacter.

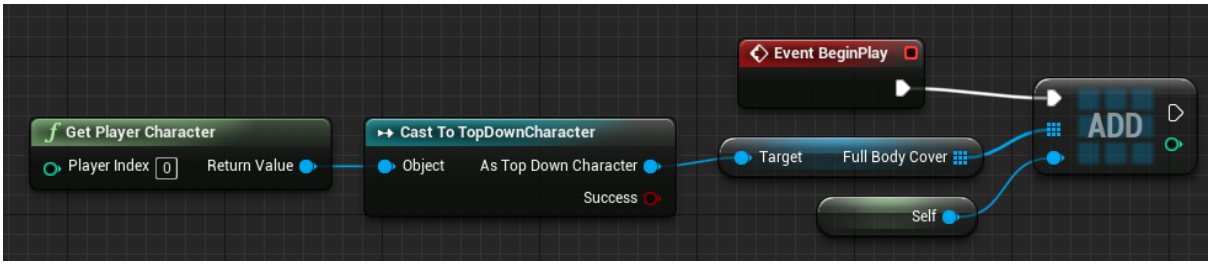


Figura 94: Captura de la funció BeginPlay

6.6.1.2. On Component Begin Overlap

Aquesta funció (veure Figura 95) és l'encarregada de guardar la posició del personatge/enemic quan entra a la cobertura.

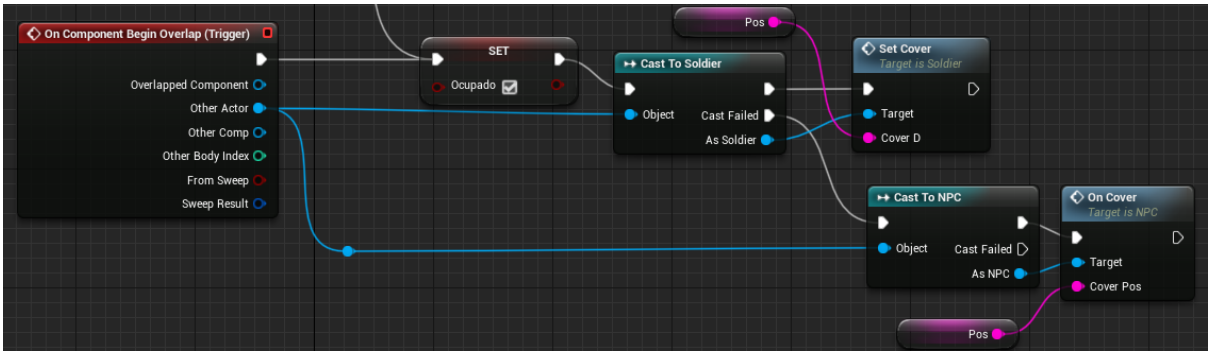


Figura 95: Captura de la funció On Component Begin Overlap

6.6.1.3. On Component End Overlap

Aquesta funció (veure Figura 96) és l'encarregada de modificar la variable del personatge/enemic referent a la cobertura. L'estableix a"NoCover".

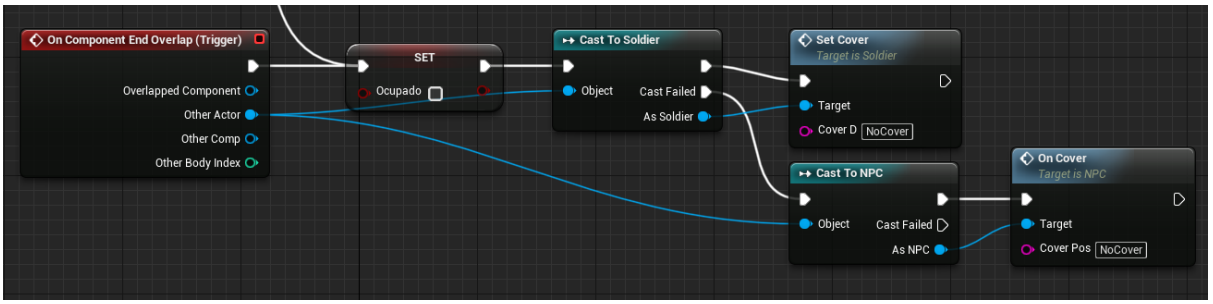


Figura 96: Captura de la funció On Component End Overlap

6.6.2. FullCoverType Actor

Aquest actor (veure Figura 97) és la cobertura ja construïda, en aquest actor, la cobertura es tracta d'un bloc 1x1 més les cel·les veïnes, les quals són "Child Actor Component". Aquest actor té un comportament senzill i poques funcions.

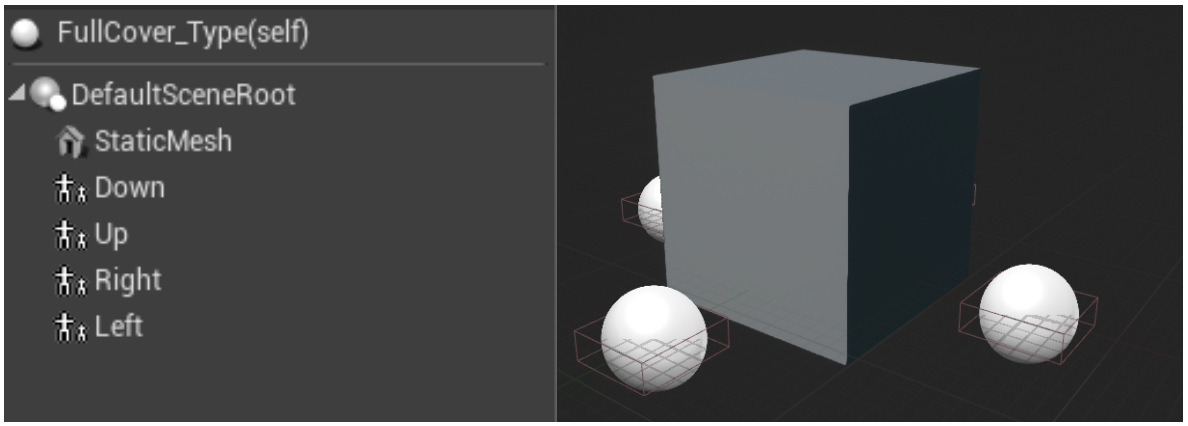


Figura 97: Captura del viewport de l'actor FullCover_Type juntament amb els components

6.6.2.1. Inicialització

Aquesta funció (veure Figura 98) serveix per guardar cada actor al seu array pertinent dins de l'actor TopDownCharacter.

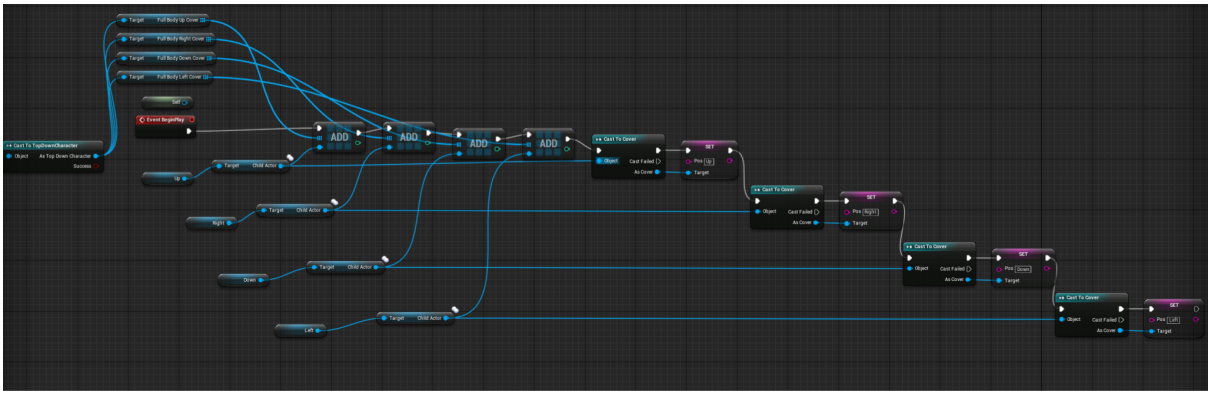


Figura 98: Captura de la funció BeginPlay

6.6.3. Variacions de l'actor FullCoverType

Veient com necessitem el mateix actor a tots els mapes, però amb diferent mesh i material, vam decidir crear actors fills del FullCoverType que servissin per a tots els nivells, acabant amb els següents actors:

6.6.3.1. Variacions pel nivell 1 i 2

Veient el nivell que estava fent, vaig buscar dins dels paquets implementats objectes que s'integressin en l'escenari que estava creant, obtenint dos models diferents, que una vegada implementats als fills, es van obtenir:

- Variació Roca. Veure Figura 99.

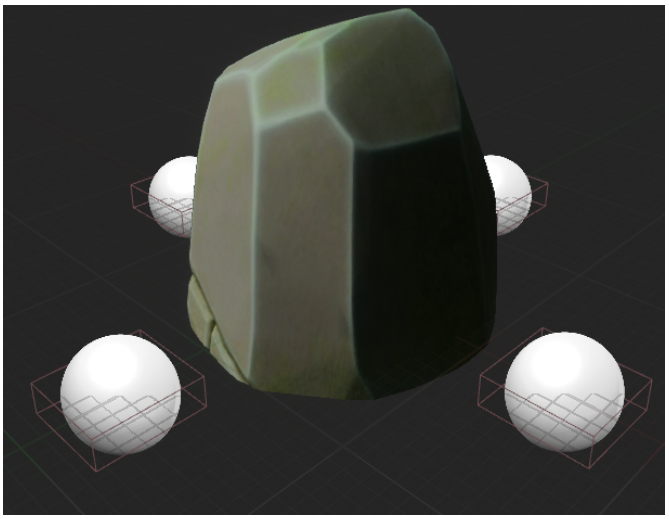


Figura 99: Captura del viewport de l'actor FullCover_Type_Rock

- Variació Arbre. Veure Figura 100.

Figura 100: Captura del viewport de l'actor FullCover_Type_TreeBase

6.6.3.2. Variacions pel nivell 3

Similar a l'anterior cas, vaig trobar els següents models:

- Variació bloc de pedra. Veure figura 101.

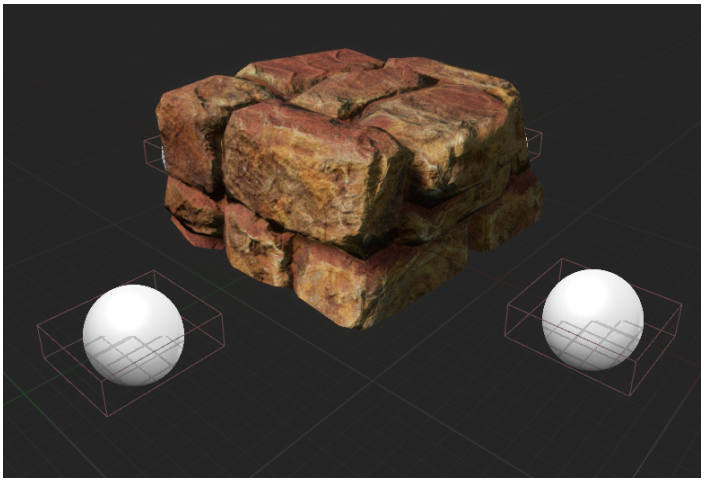


Figura 101: Captura del viewport de l'actor FullCover_Type_CubeRuin

- Variació columna en ruïnes. Veure Figura 102.

Figura 102: Captura del viewport de l'actor FullCover_Type_ColumnRuin

6.6.4. 3x1Cover Actor

Com el nom indica, aquestes cobertures no són com les anteriors, aquestes són rectangulars. D'aquesta manera li donem més joc a la mecànica. Similar a la cobertura FullCoverType, aquest actor (veure Figura 103) no disposa d'altres funcions que no sigui el BeginPlay.

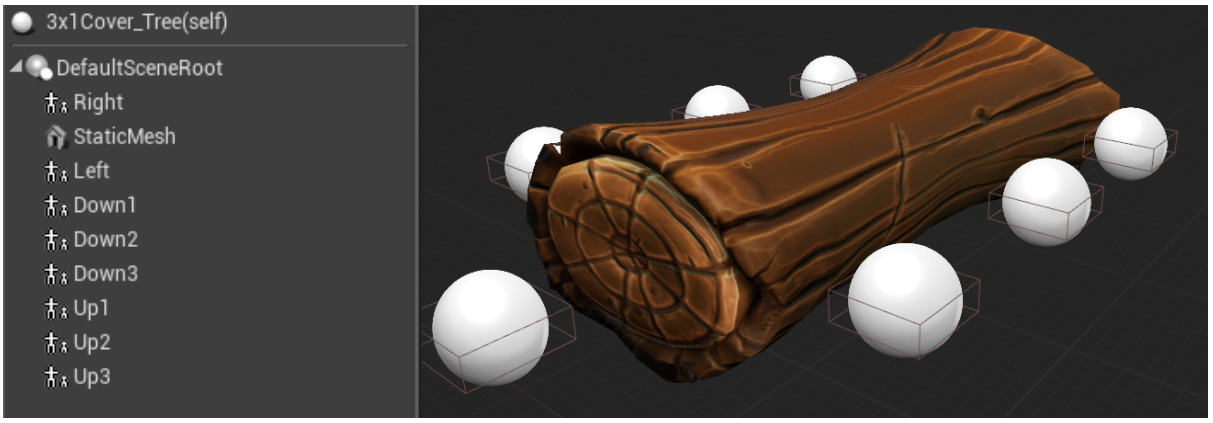


Figura 103: Captura del viewport de l'actor 3x1Cover juntament amb els seus components

6.6.3.1. Inicialització

Aquesta funció (veure Figura 104) serveix per guardar-se cada actor a l'array pertinent dins de l'actor TopDownCharacter.

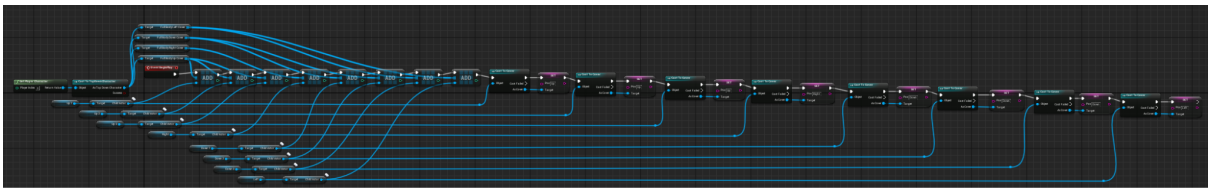


Figura 104: Captura de la funció BeginPlay

6.6.5. Variacions de l'actor 3x1Cover

Veient com necessito el mateix actor a tots els mapes, però amb diferent mesh i material, en aquest cas vaig crear un nou Actor per cada cobertura:

6.6.5.1. Variacions pel nivell 1 i 2

Veient el nivell que estàvem fent, vam buscar dins dels paquets implementats els objectes que s'integressin en l'escenari que estàvem creant, a part de la mesh mostrada prèviament, vam trobar-nos amb un altre, un cop creat, vam obtenir el següent actor:

- Variació Roca. Veure Figura 105.

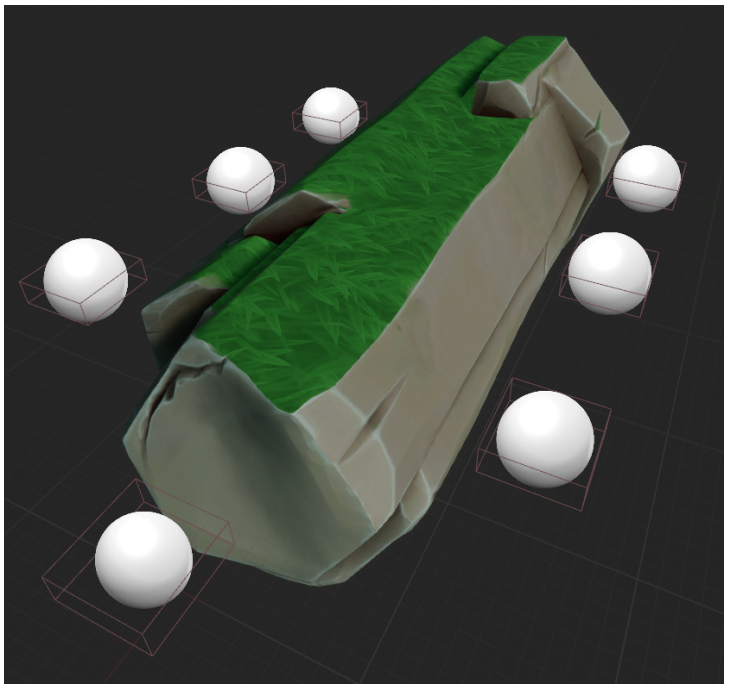


Figura 105: Captura del viewport de l'actor 3x1Cover_Rock

6.6.5.2. Variacions pel nivell 3

En aquest cas, pel nivell 3, només vam trobar un mesh que s'integrés dins de l'escenari, aquest és el següent:

- Variació muralla en ruïnes. Veure Figura 106.

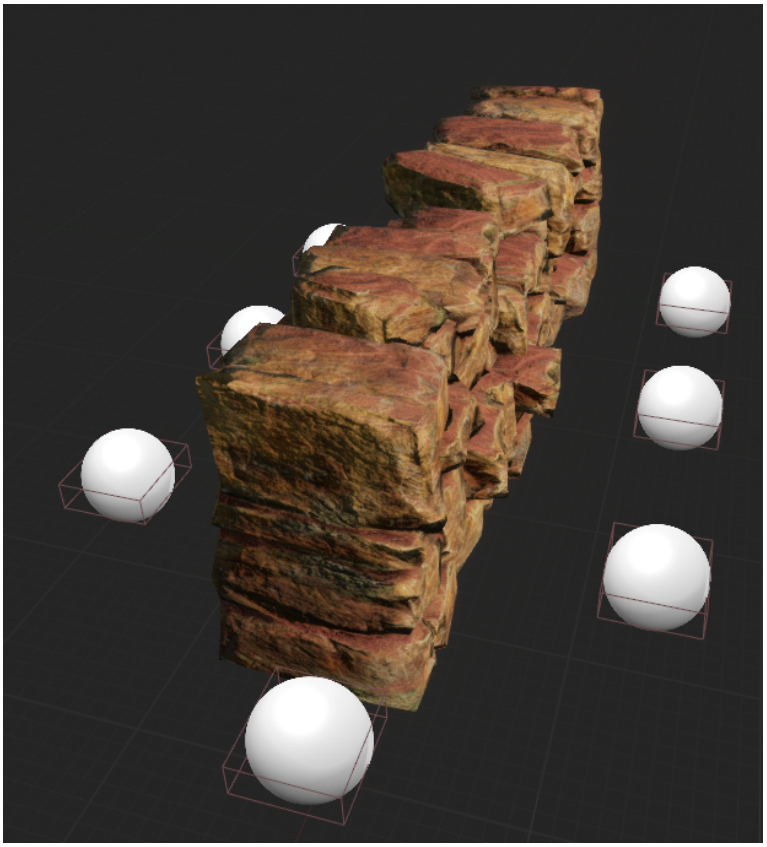


Figura 106: Captura del viewport de l'actor 3x1Cover_WallRuin

6.7. Implementació personatges

Els personatges són les tropes que el jugador controla, cada un d'ells ha de tenir variables úniques. Per aquesta raó vaig crear un Actor (veure Figura 107) específic anomenat "Soldier".

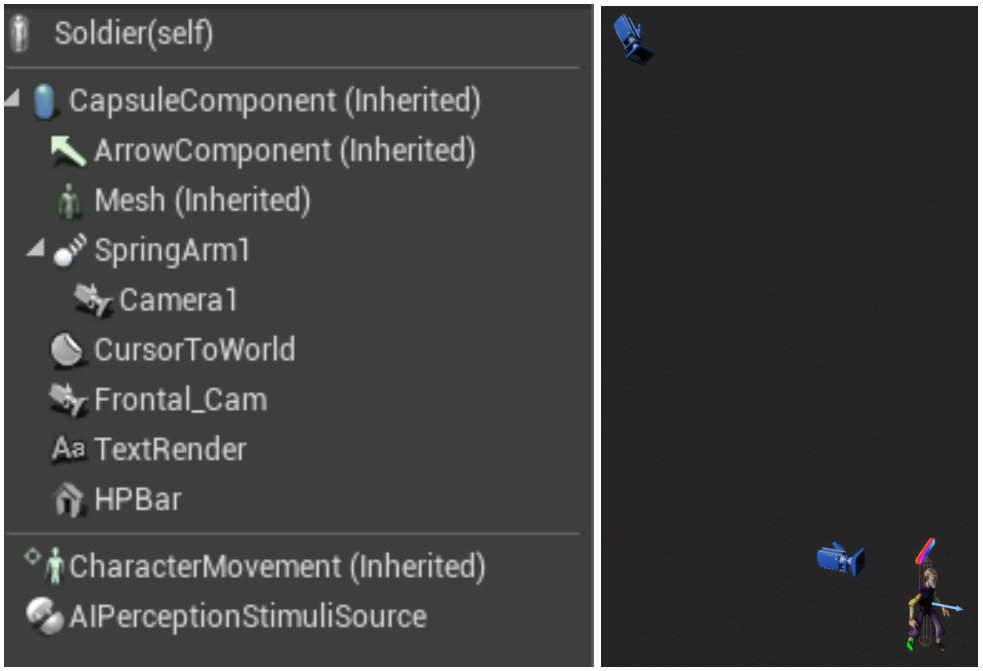


Figura 107: Captura del viewport de l'actor Soldier juntament amb els components

Passant a les funcions d'aquest actor, disposa d'una gran varietat i durant tot aquest capítol, anirem explicant a quines funcionalitats responen tots aquests Events, Functions i variable (veure Figures 108 i 109).

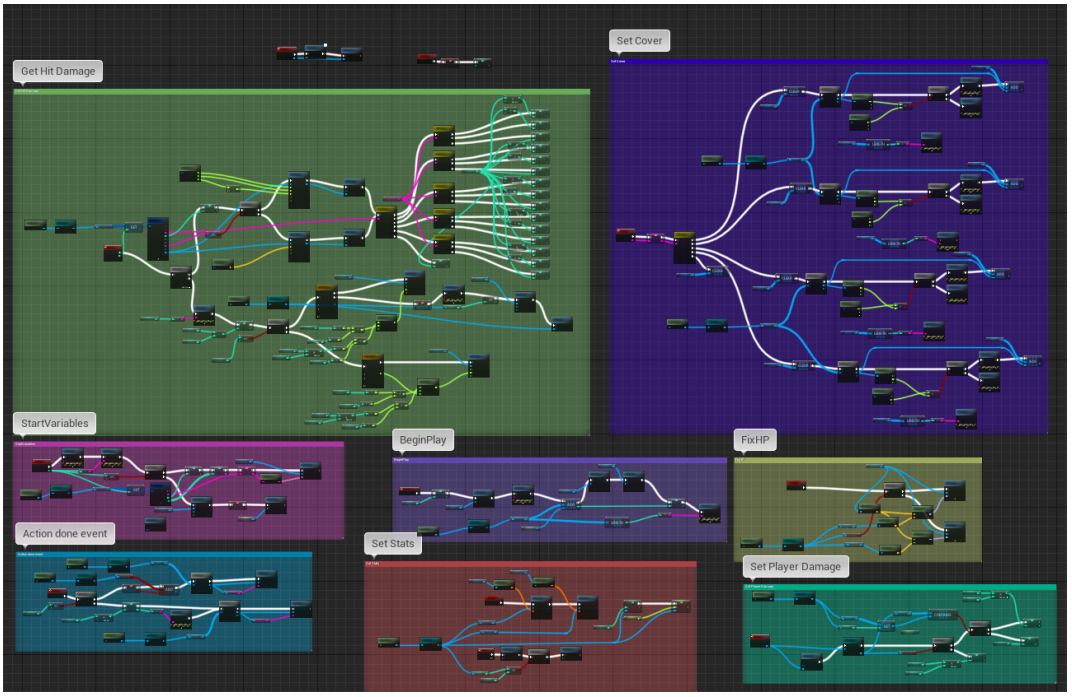


Figura 108: Captura Blueprint Soldier

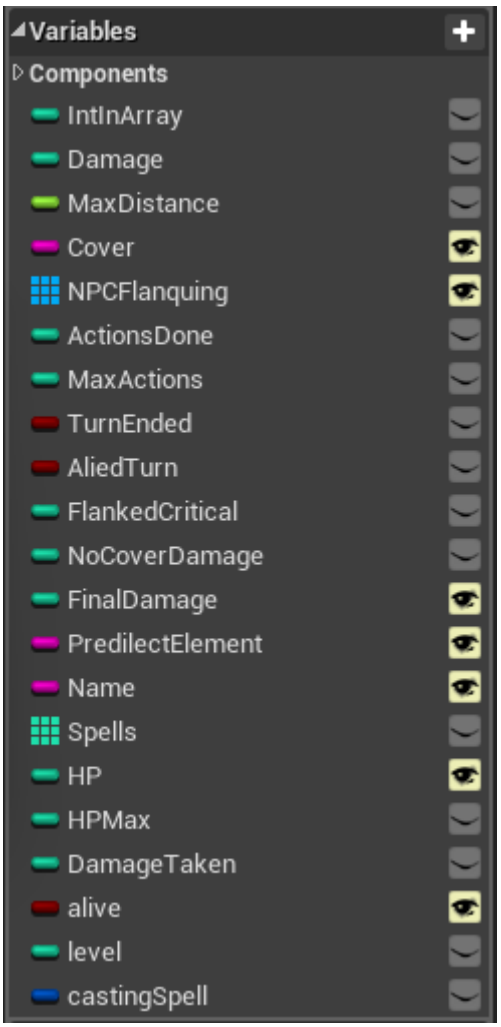


Figura 109: Captura de les variables de l'actor Soldier

Aquestes variables són totes d'aquest actor. Si cal destacar algunes, aquestes serien:

- Cover (string): Posició de la cobertura, es representa amb UP/Down/Right/Left/NoCover.

- NPCFlanquing (Actor array): llista de tots els enemics que estan sent flanquejats.

- ActionsDone(int): nombre d'accions fetes aquest torn.

- MaxActions(int): nombre d'accions que s'han de fer per torn abans que aquest acabi.

- PredilectElement(string): nom de l'element característic del personatge.

- Spells(int array): llista dels Spells del personatge.

6.7.1. Inicialització

L'Event BeginPlay (veure Figura 110) s'encarrega d'inicialitzar algunes variables i elements importants:

- S'especifica la vida del jugador a partir de la vida màxima.

- S'afegeix aquest actor a l'array d'actors del gestor de personatges.

- S'oculten les dues càmeres.

- Ens guardem l'index del personatge de l'array d'actors mencionat abans com a una variable.

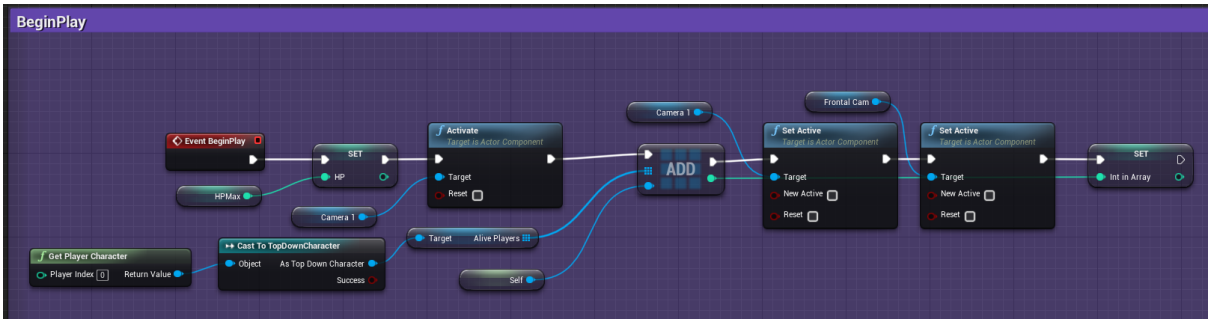


Figura 110: Captura de la funció BeginPlay

6.7.2. Event Tick / MyStats

En Aquest Actor, el Event Tick gestiona les càmeres. Aquesta funció s'encarrega de (veure Figura 111):

- Mira si el personatge en qüestió té el mateix índex que la variable de PlayerIndex del gestor dels personatges

- En cas de ser-ho, crida una funció que mou les càmeres del gestor fins a la posició de les càmeres del personatge.

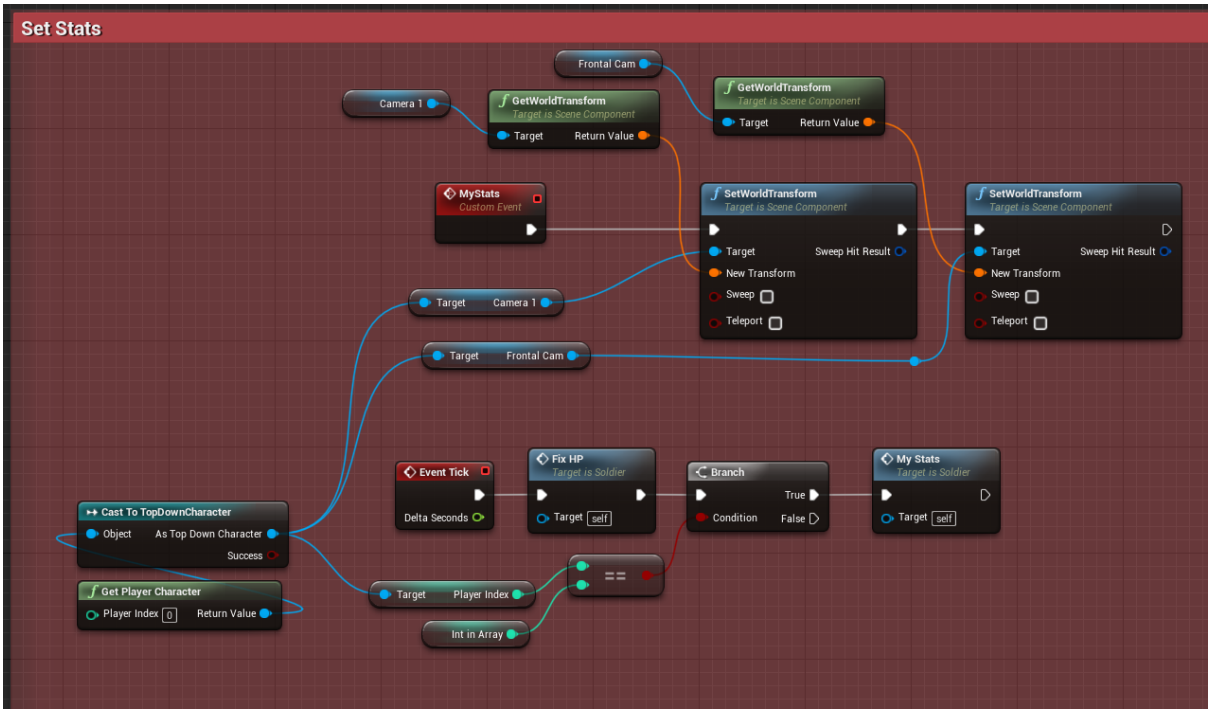


Figura 111: Captura de les funcions Event Tick i MyStats

6.7.3. StartVariables

Aquesta funció (veure Figura 112), és l'encarregada de saber quin personatge és l'actor en concret, i funciona de la següent manera:

- Mira si l'id introduït és igual o superior de 0.

- Si ho hes, busca el jugador amb aquesta id dins de la Structure dels personatges i sobreescriu la informació necessària.

- Si per altra banda no ho és, amaga l'actor dins del joc.

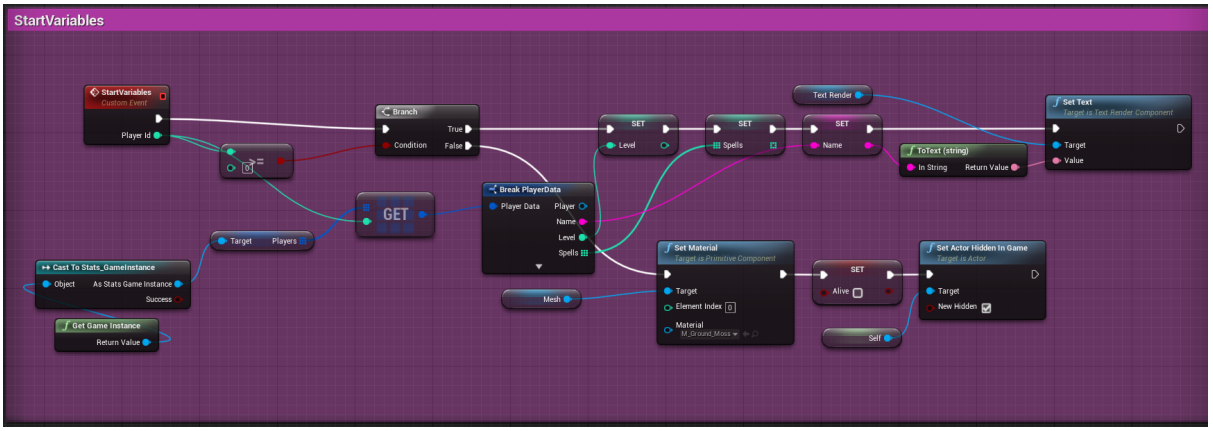


Figura 112: Captura de la funció StartVariables

6.7.4. ActionDone

Aquest actor (veure Figura 113) gestiona les accions preses per aquesta classe. Fa el següent:

- Mira el nombre d'accions preses per aquest personatge aquest torn.

- Si són iguals a 2, es marca com a torn acabat per aquest personatge, s'afegeix a l'array del gestor de personatges i per cada enemic. Es crida la funció OnCover.

- Si no és igual a 2, se suma una acció a la variable i per cada enemic es crida la mateixa funció.

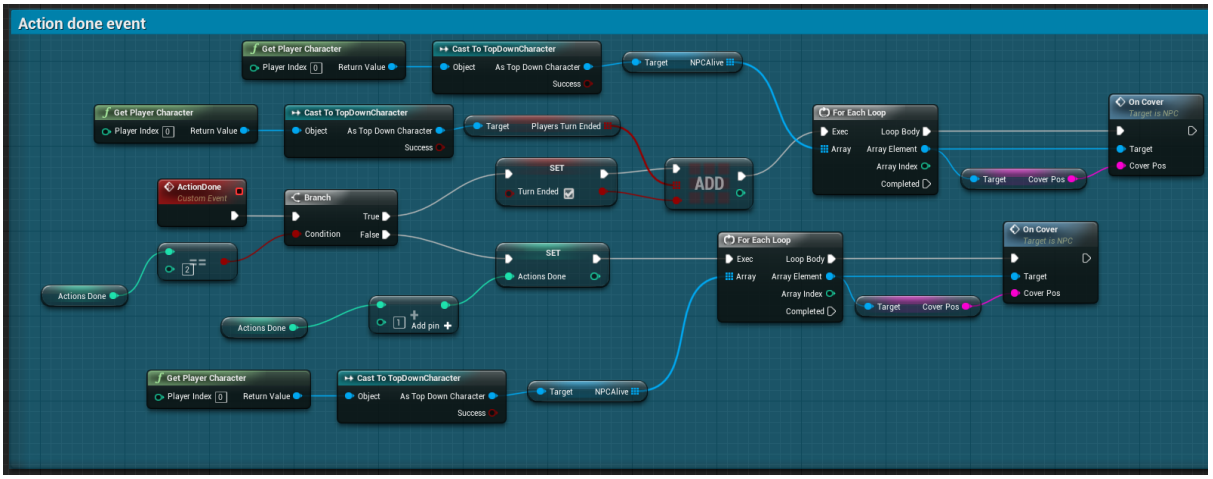


Figura 113: Captura de la funció ActionDone

6.7.5. FixHP

Aquesta funció (veure Figura 114), és una funció auxiliar que ajusta la rotació de la barra de vida de la següent manera:

- Mira quina càmera està en és.
- Crida el node “Find look at rotation”.
- Ajusta la rotació de la barra de vida perquè sempre apunti en direcció a la càmera.

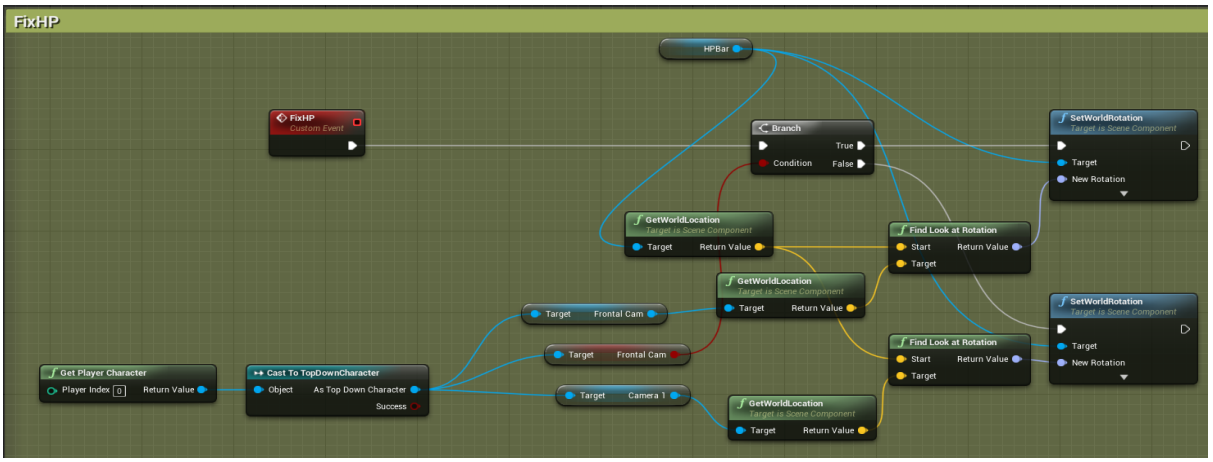


Figura 114: Captura de la funció FixHP

6.7.6. StartTurn

La funcionalitat d'aquesta funció és molt senzilla, ja que únicament s'encarrega de tornar les variables que depenen de les accions del torn l'estat inicial. Veure Figura 115.

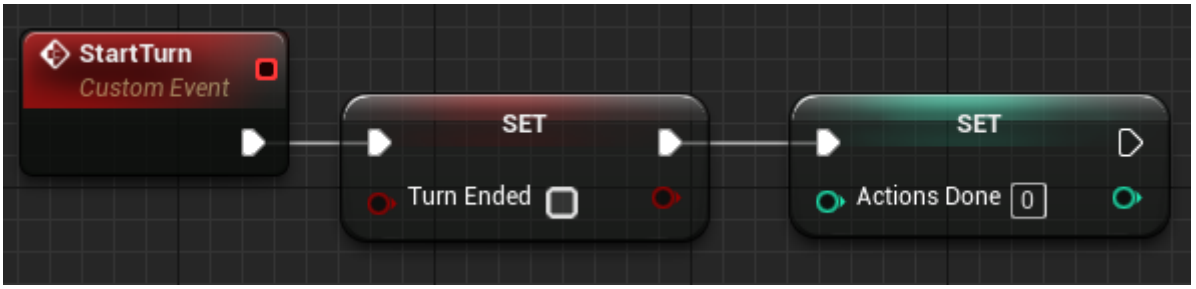


Figura 115: Captura de la funció StartTurn

6.7.7. GetHit / DeleteEmitter

La funció GetHit (veure Figures 116) és l'encarregada de gestionar i calcular el mal que rep el personatge i la funció de DeleteEmitter (veure figura 123) s'encarrega d'eliminar el FX passat cert temps, i fa el següent:

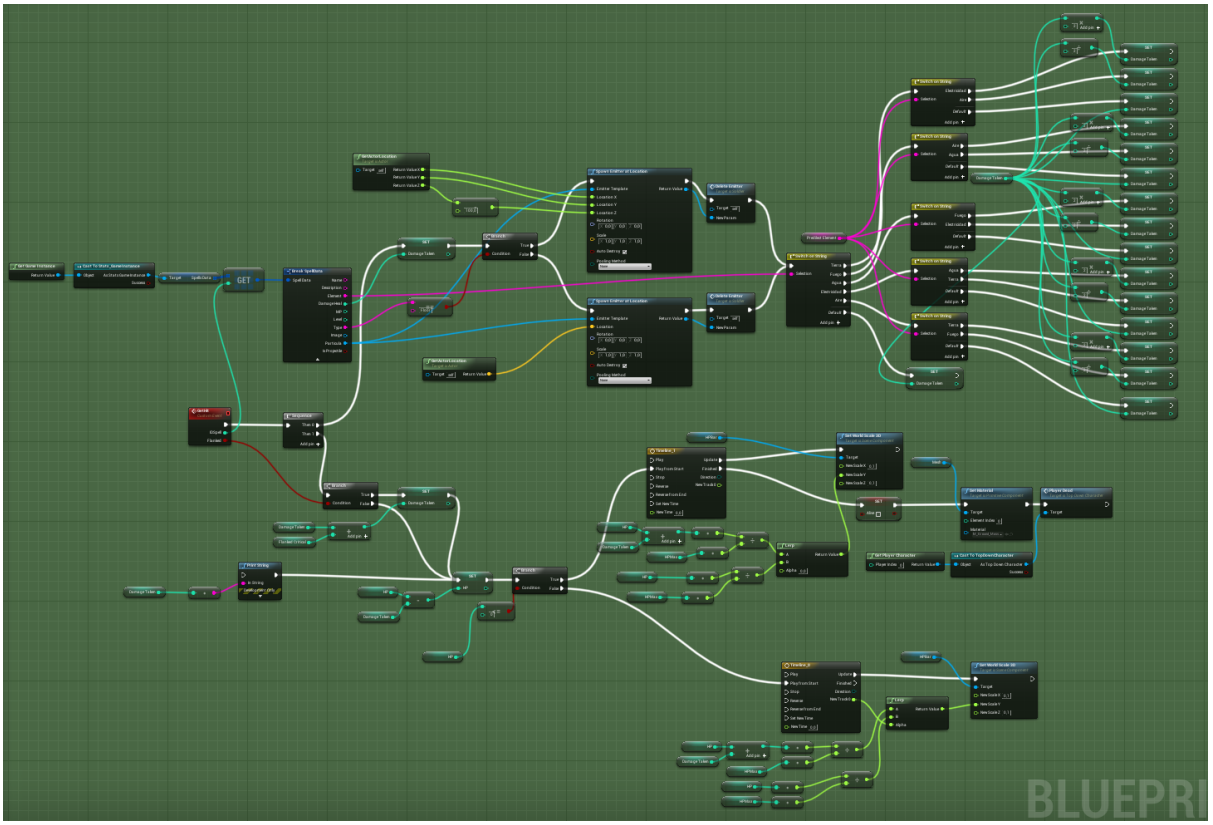


Figura 116: Captura de la funció GetHit

- A partir de l'id del Spell agafada per paràmetre, obtenim el Spell amb aquest id.

- Ens guardem el mal del Spell dins la variable "DamageTaken". Veure Figura 117.

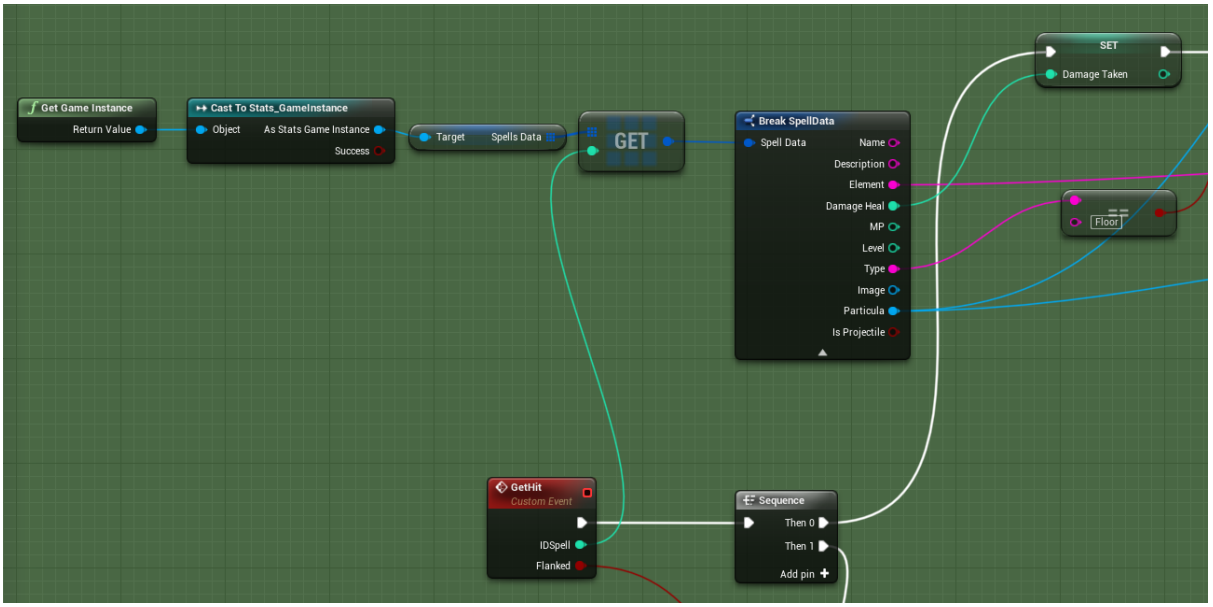


Figura 117: Captura de la funció GetHit 1

- Mitjançat un branch, mirem quin tipus d'atac és el Spell.

- Depenent del tipus de Spell, creem un Emitter a una posició (centre del personatge) o a un altre (als peus del personatge).

- Cridem la funció DeleteEmitter. Veure Figura 118.

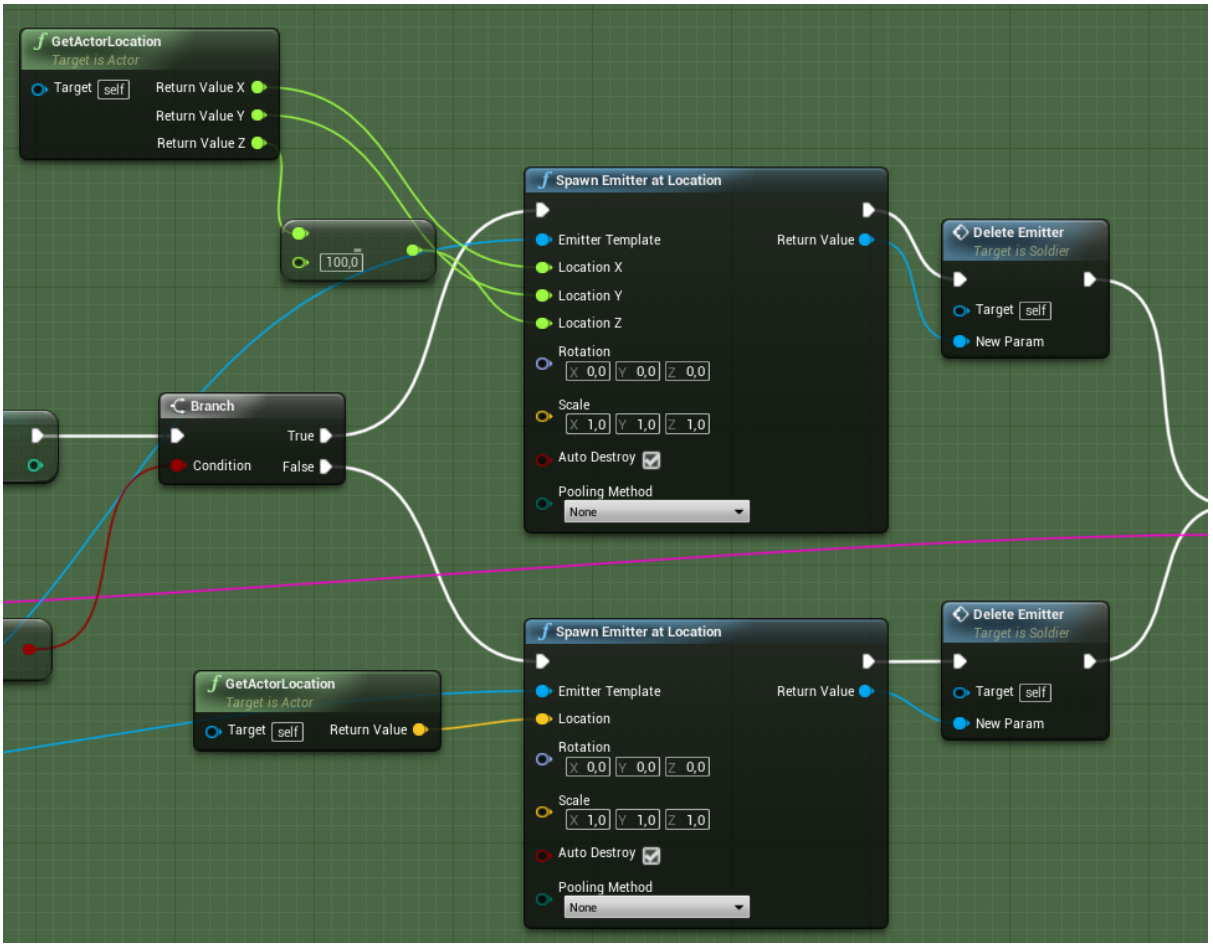


Figura 118: Captura de la funció GetHit 2

- Fem un switch en string a partir de la variable element del Spell.

- Per cada element del switch, fem un altre switch on string, però només dels dos elements que modifiquen el multiplicador del mal utilitzant la variable del personatge anomenada “Predilect Element”.

- Apliquem el multiplicador del mal a la variable “DamageTaken”. Veure Figura 119.

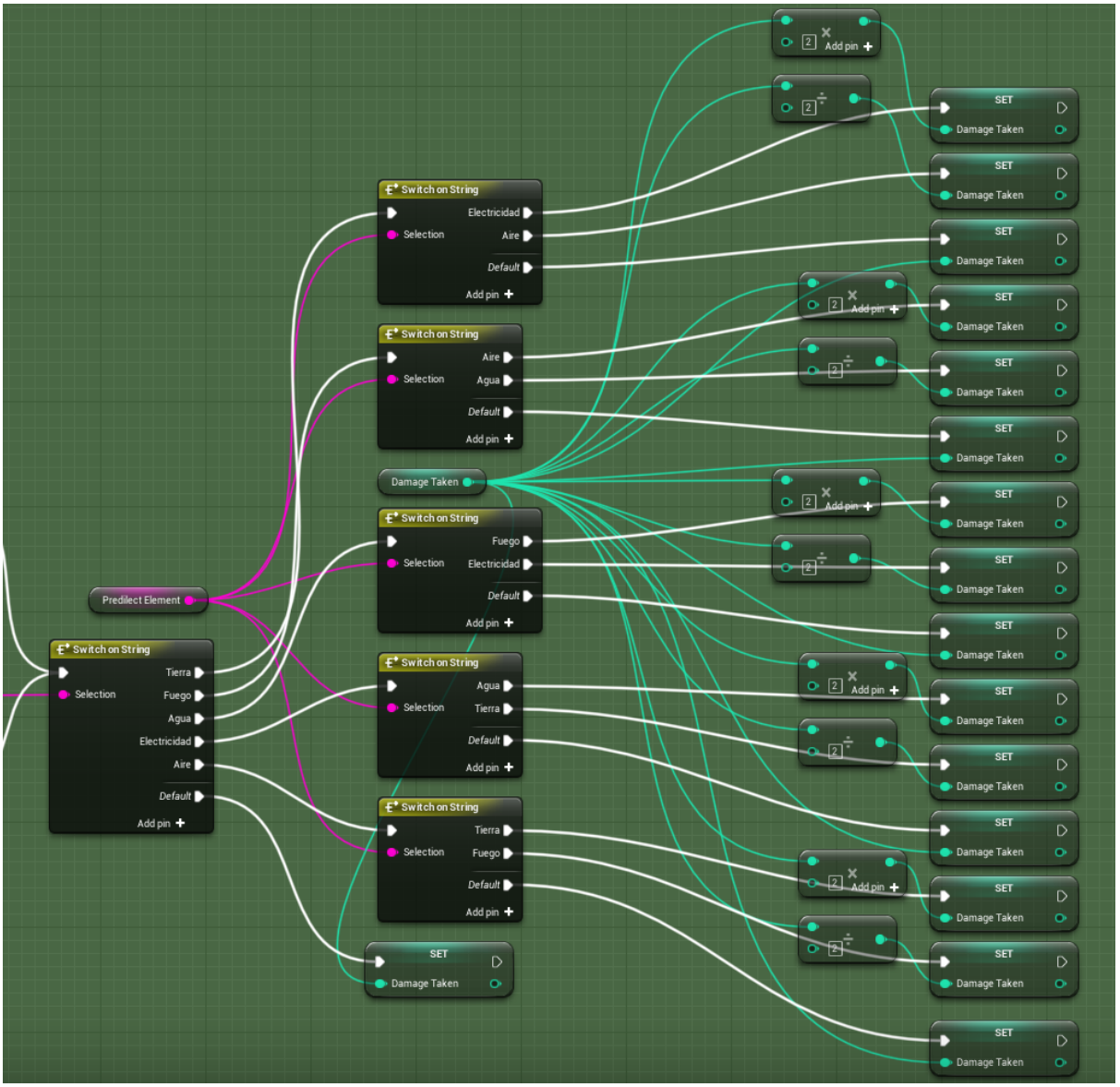


Figura 119: Captura de la funció GetHit 3

- Passem a la part 2 del sequence, on es mira si el personatge ha estat flanquejat o no mitjançant la variable bool passada per referència.

- Sumem la variable “FlankedCritical” a la variable “Damage Taken”.

- Restem la variable “DamageTaken” a la vida actual del personatge. Veure Figura 120.

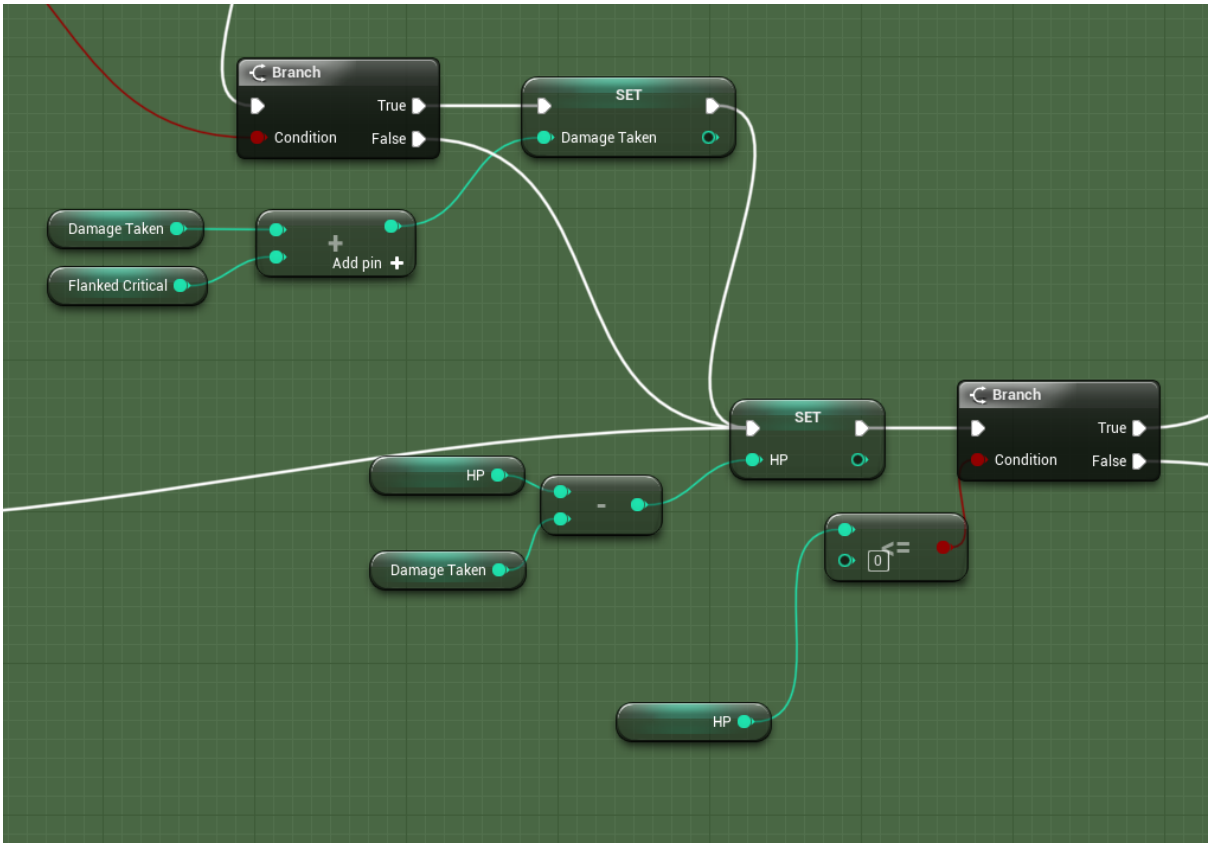


Figura 120: Captura de la funció GetHit 4

- Si la vida és més gran de 0, modifiquem la mesh de la barra de vida utilitzant l'update d'un timeline, amb un Lerp. Al Lerp li hem de posar a la variable "A" vida que es tenia abans del cop dividit la vida màxima, i a la variable "B", la vida actual dividit la vida màxima, i a la variable alpha li hem de posar l'output del timeline. D'aquesta manera la barra de vida fa una animació quan rep mal. Veure Figura 121.

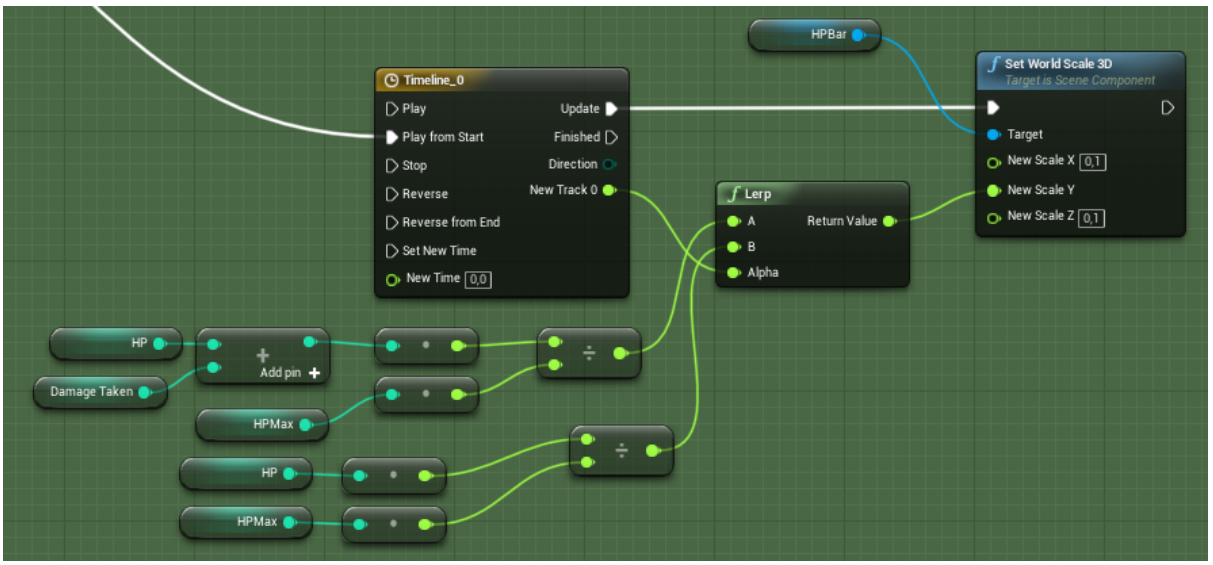


Figura 121: Captura de la funció GetHit 6

- Si la vida és inferior a 0, modifica la mesh de la barra de vida amb una timeline com abans i crida a la funció PlayerDead del gestor. Veure Figura 122.

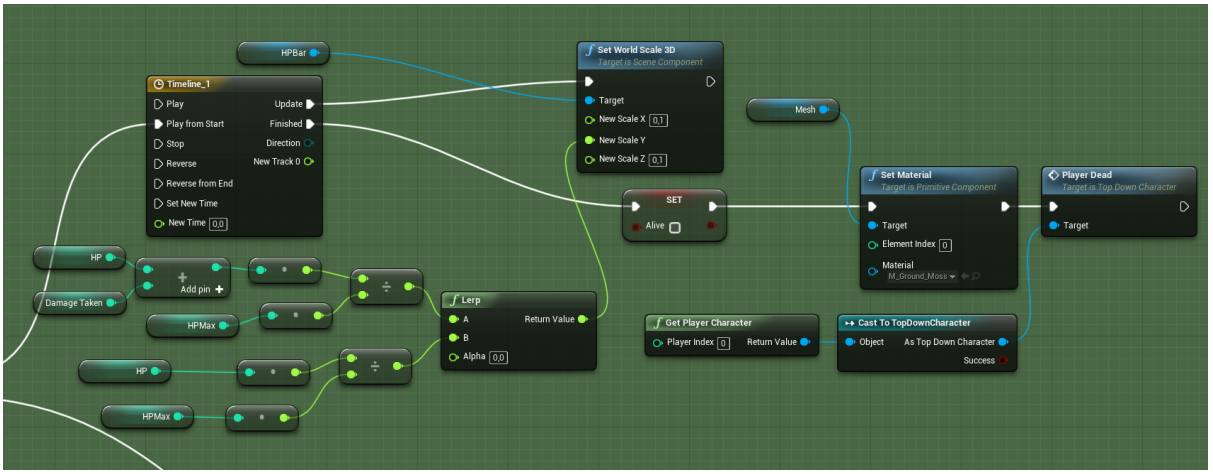


Figura 122: Captura de la funció GetHit 5

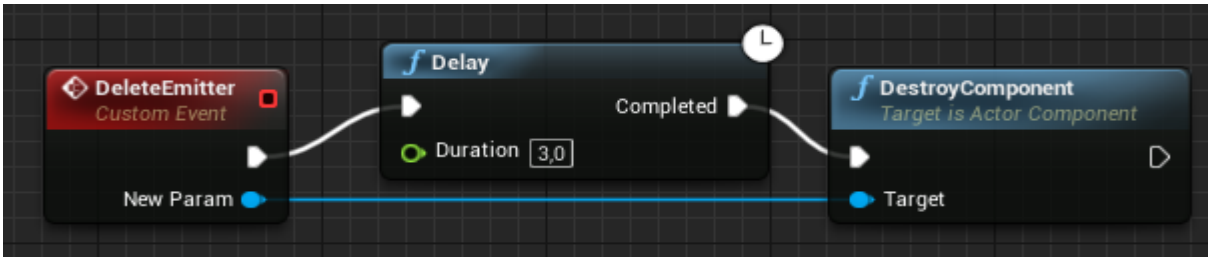


Figura 123: Captura de la funció DeleteEmitter

6.7.8. SetCover

Aquesta funció (veure Figura 124), és l'encarregada de gestionar a quin enemic està flanquejant el jugador, i fa el següent:

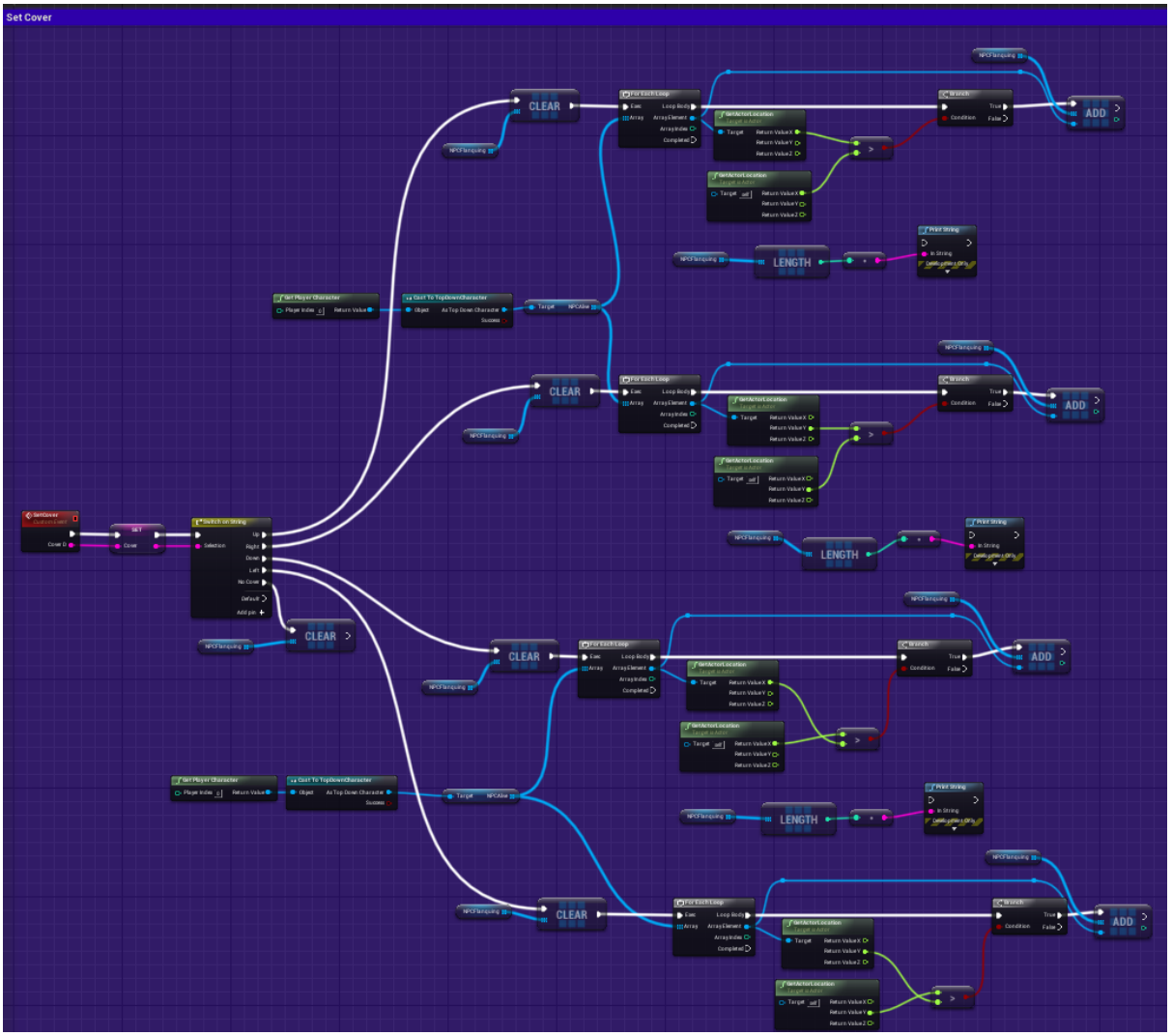


Figura 124: Captura de la funció SetCover

- Es guarda la posició de la cobertura en la qual es troba Up/Down/Right/Left/NoCover. Veure Figura 125.

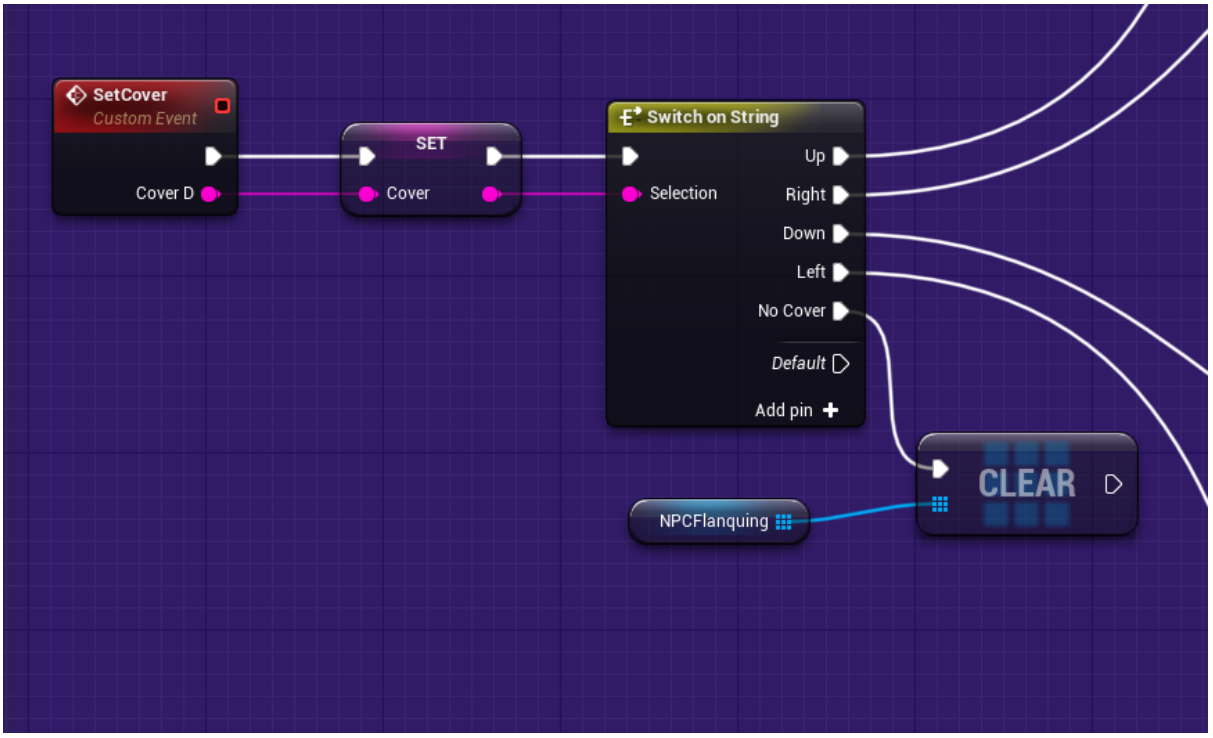


Figura 125: Captura de la funció SetCover 1

- Mitjançant un switch on string, es mira la posició de la cobertura en la qual es troba.

- En el cas de ser Up, es buida l'array de la variable NPCFlanking, es mira l'array de NPCAlive del gestor de personatges i es fa un "For Each Loop" que compara la posició dels enemics amb la dels personatges. Si la X del personatge és més gran que la de l'enemic, està flanquejant-lo, per tant ens guardem l'actor l'array NPCFlanking. Veure Figura 126.

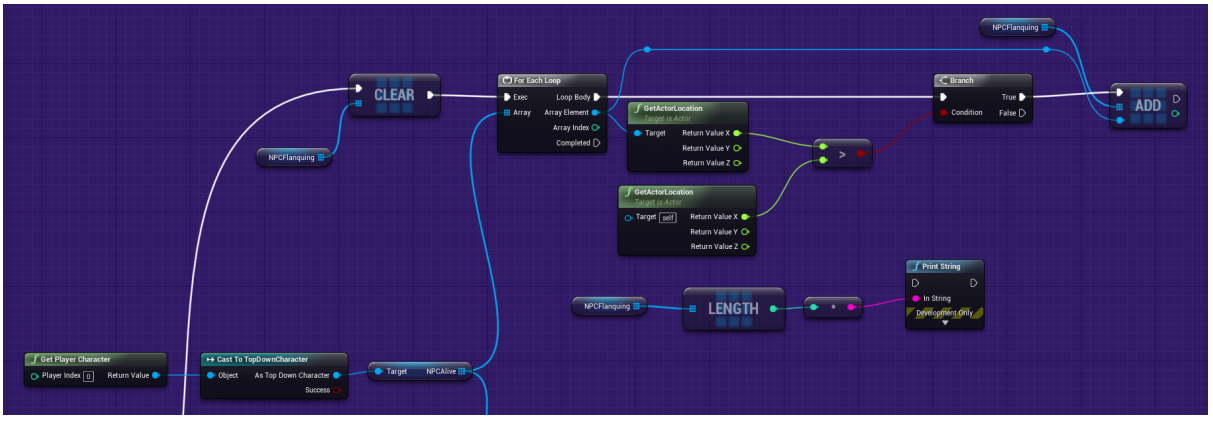


Figura 126: Captura de la funció SetCover 2

- En el cas de ser Right, se segueix el model anterior, però es compara la posició Y. Si la Y del personatge és més gran que la de l'enemic, està flanquejant-lo. Per tant ens guardem l'actor l'array NPCFlanking. Veure Figura 127.

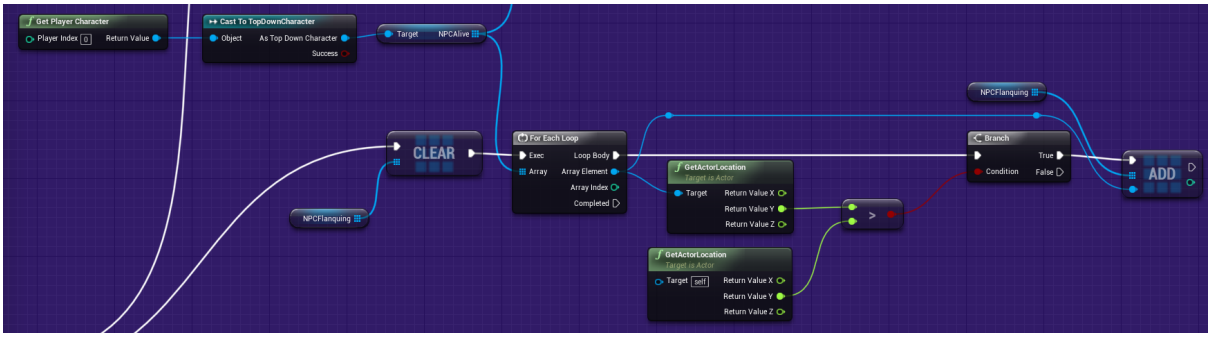


Figura 125: Captura de la funció SetCover 3

- En el cas de ser Down, és igual que en el cas Up, però estarà flanquejant-lo si l'enemic té la X més gran que el personatge. Veure Figura 128.

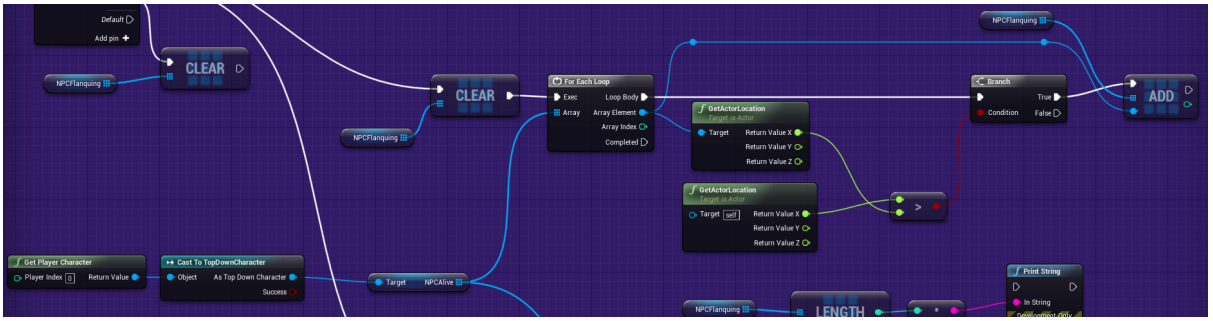


Figura 128: Captura de la funció SetCover 4

- En el cas de ser Left, és igual que en el cas Right, però estarà flanquejant-lo si l'enemic té la Y més gran que el personatge. Veure Figura 129.

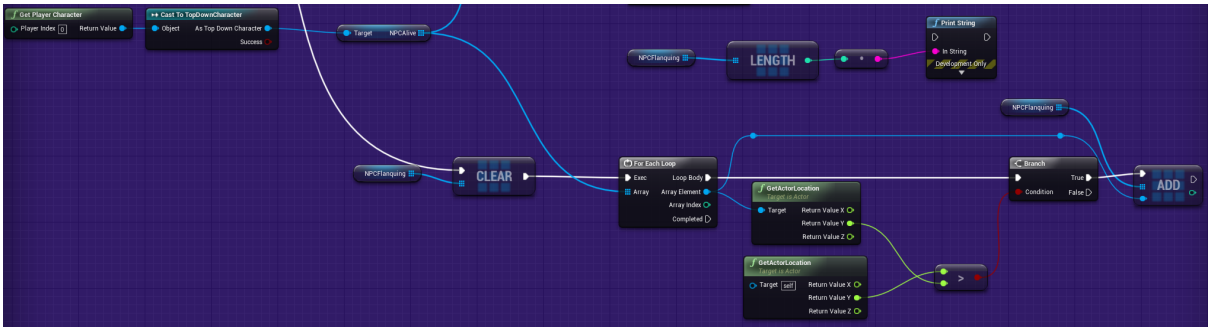


Figura 129: Captura de la funció SetCover 5

- En el cas de ser NoCover, per penalitzar al jugador es buida la variable NPCFlanking.

6.8. Implementació dels enemics

Els enemics són una part fonamental del joc i requereixen un nivell de credibilitat acceptable per no empobrir el resultat final.

La implementació està dividida en diversos elements (veure Figura 128) amb funcions específiques:

- Enemy: és el Character de l'enemic. Gestiona la Mesh, les animacions i les col·lisions del personatge. Té assignat un AI Controller, anomenat "NPC_Controller".

- NPC_Controller: és l'AI Controller de l'enemic.

- NPC_BB: és un diccionari amb totes les variables que el Behaviour Tree utilitzarà per decidir el comportament de l'enemic.

- Tasks: són les tasques que realitzarà l'enemic quan es donin les condicions necessàries (especificades al Behaviour Tree). Alguns exemples de tasques són: Buscar millor cobertura, atacar, acció feta, etc.

- NPC_BT: és un arbre de comportament (Behaviour Tree) que decideix, depenent de l'estat de les variables del Blackboard, el comportament que té l'enemic. El comportament està dividit en branques que fan crides a Tasks o tasques específiques.

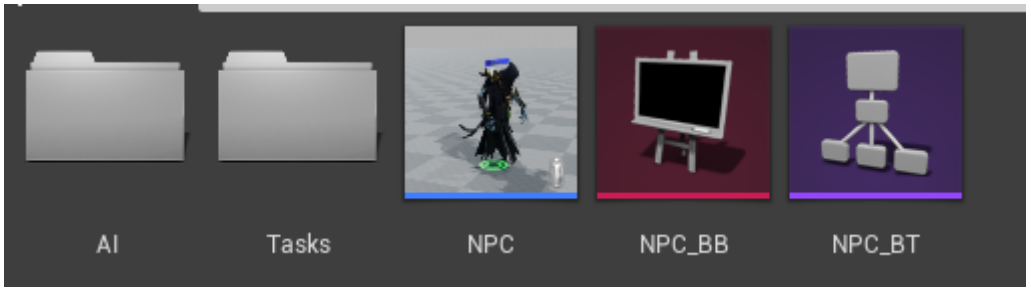


Figura 128: Captura a l'explorador de tots els elements que fan funcionar els enemics

6.8.1. Enemic

Aquest actor és l'encarregat de gestionar els enemics. A causa d'això, és l'actor més gran del projecte. Aquest actor s'anomena "NPC" (veure Figura 129).

Figura 129: Captura del viewport de l'actor NPC juntament amb els seus components

Algunes de les configuracions més importants canviades en aquest Blueprint és l'apartat de Pawn. Veure Figura 130:

- En l'apartat pawn, és necessari que la variable "AI Controller Class" li seleccionem el controllador creat per a aquest actor.

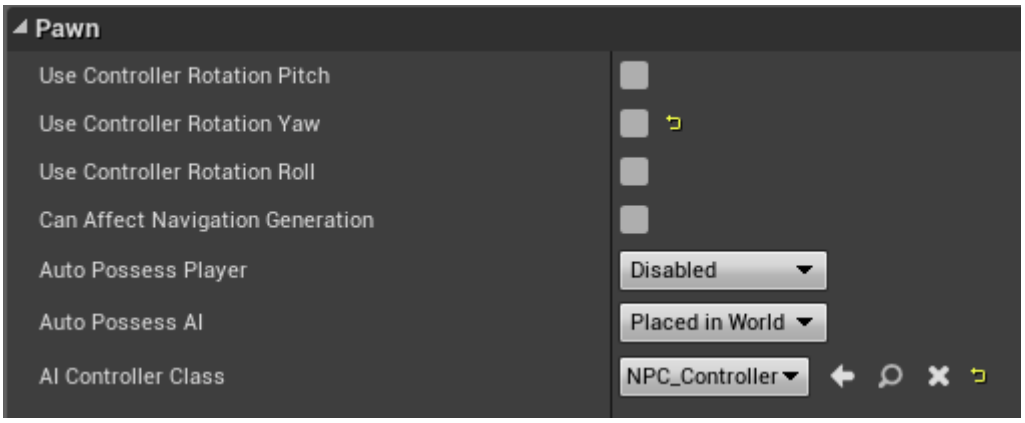


Figura 130: Imatge de la secció de la configuració de l'actor "NPC " representant a l'apartat del Pawn

Passant a les funcions d'aquest actor, disposa d'una gran varietat i durant tot aquest capítol anirem explicant a quines funcionalitats responen tots aquests Events, Functions i variables (veure Figures 131, 132 i 133).

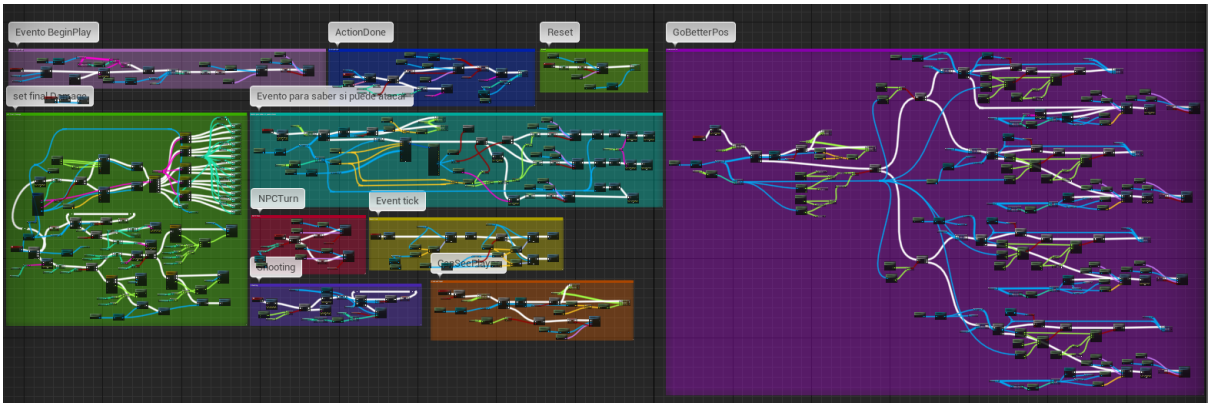


Figura 131: Captura del Blueprint NPC

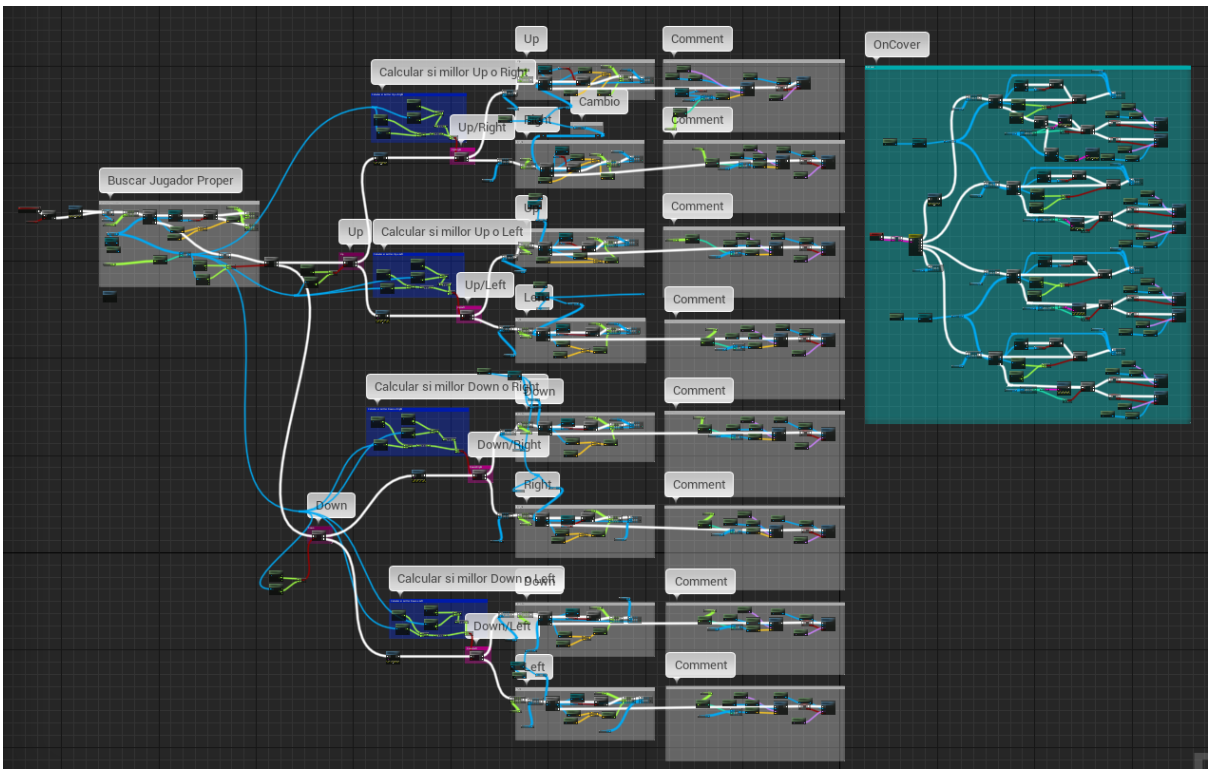


Figura 132: Captura del Blueprint NPC

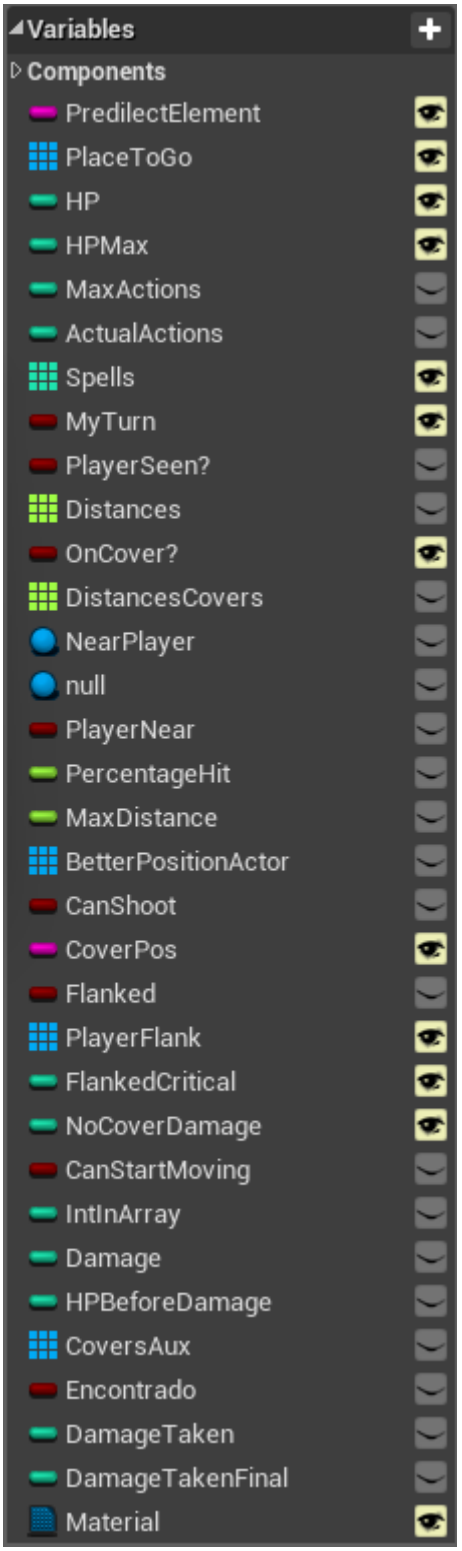


Figura 133: Captura de les variables de l'actor NPC

Aquestes variables són totes d'aquest actor. Si cal destacar algunes d'aquestes, aquestes serien:

- PredilectElement(string): variable que estableix l'element principal de l'enemic.

- MyTurn(boolean): variable que defineix si és el torn de l'enemic.

- Distances(float array): array de distàncies dels personatges.

- DistancesCovers(float array): array de distàncies de les cobertures.

- BetterPositionActor(actor array): array d'actors en el que es guarden les cobertures òptimes a on anar.

- CoverPos(string): posició de la cobertura en la que es troba, o "NoCover".

- PlayerFlank(actor array): array d'actors "Soldier" que estan sent flanquejats.

- CanStartMoving(boolean): variable que permet a l'enemic moure's quan toca.

6.8.1.1. Inicialització

L'Event BeginPlay (veure Figura 134) s'encarrega d'inicialitzar algunes variables i elements importants:

- Modifica el material depenent del material introduït des de l'escena.

- Crida la funció SetSpells.

- Afegeix l'actor a l'array d'actors del gestor i es guarda l'índex en una variable.

- Guarda la vida amb el valor de la vida màxima

- Guarda les variables MyTurn i CanShoot a false i li passa la variable a una key de la blackboard.

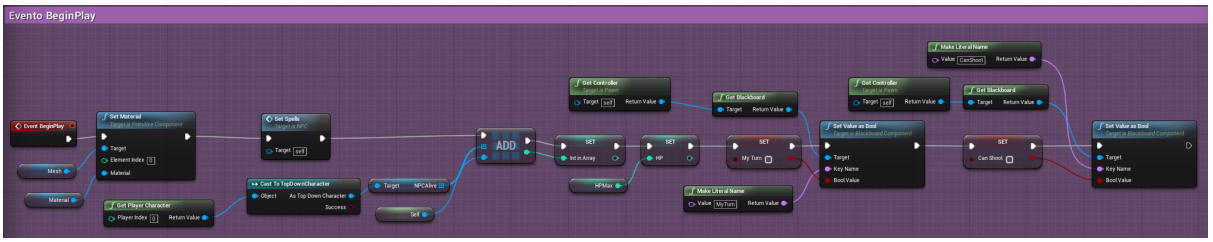


Figura 134: Captura de la funció BeginPlay

6.8.1.2. SetSpells

Aquesta funció (veure Figura 135) s'encarrega d'agafar els Spells necessaris, i ho fa de la següent forma:

- Agafa la llista de Spells de la instància de joc per obtenir la llargada d'aquesta.

- Es fa un for amb la llargada de l'array de Spell de l'actor.

- Per cada element, es busca un número random entre els valors 0 i la llargada de la llista Spell.

- Cada element es guarda l'array de Spell modificant el valor d'aquesta en l'índex pertinent.

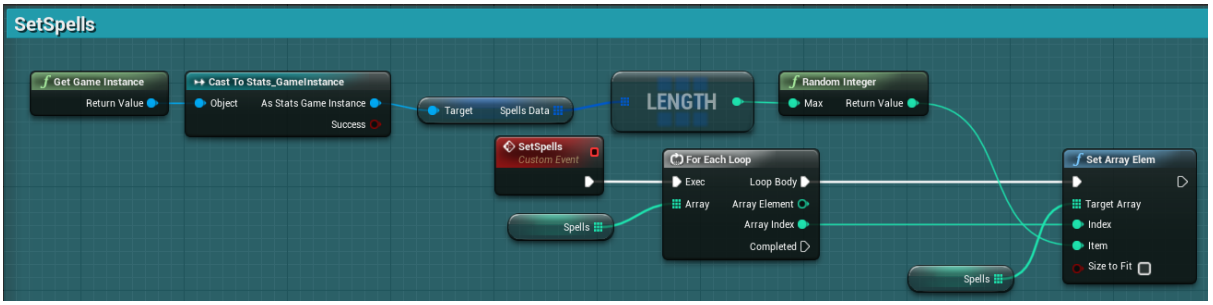


Figura 135: Captura de la funció SetSpells

6.8.1.3. Event Tick

Aquest event (veure Figura 136) disposa de 4 accions que realitza cada tick, i aquestes són:

- Cridar a la funció CanSeePlayer.

- Gestiona el Text (nombre d'accions fetes per aquest enemic) perquè sempre miri cap a la càmera.

- Crida la funció NPCTurn.

- Es gestiona la barra de vida perquè sempre miri cap a la càmera.

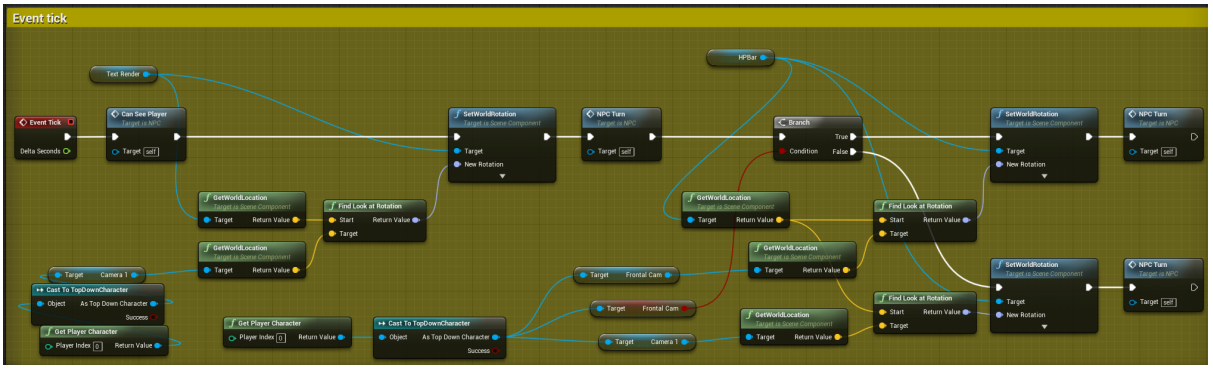


Figura 136: Captura de la funció Event tick

6.8.1.4. CanSeePlayer

Aquest event (veure Figura 137) s'encarrega de saber quan el personatge s'ha apropat molt a un enemic i fa el següent:

- Comprova si hi ha un jugador a prop amb la variable PlayerNear.

- Si no hi ha ningú, buida l'array de floats anomenat Distancies.

- Mitjançant un bucle que recorre la llista d'actors dels personatges, mira si el personatge està viu. En el cas d'estar-ho, es guarda la distància entre l'un i l'altre.

- Un cop recorregut l'array, mitjançant un branch, mira si el valor mínim de l'array de Distancies és menor a x.

- En cas de ser-ho, es marca que l'enemic ha vist el jugador amb la variable PlayerNear.

- Es passa la variable PlayerNear a la Blackboard.

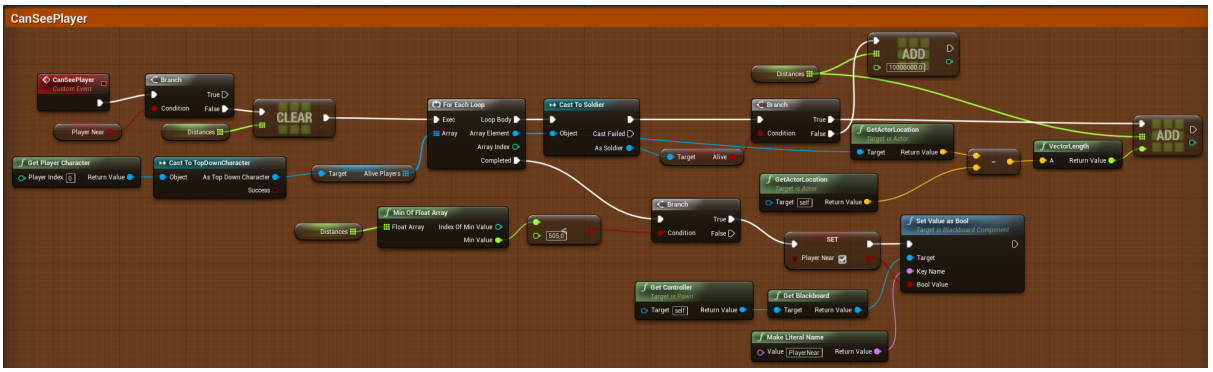


Figura 137: Captura de la funció CanSeePlayer

6.8.1.5. NPCTurn

Aquesta funció (veure Figura 138) mira si és el torn de moure's. Això ho fa de la següent manera:

- Mira si la variable NPC_Moving és igual a la variable que guarda l'índex de l'array NPCAlive.

- En cas de ser-ho, es modifica la variable CanStartMoving i es passa a la Blackboard.

- En cas de no ser-ho, també es modifica la variable CanStartMoving i es passa a la Blackboard.

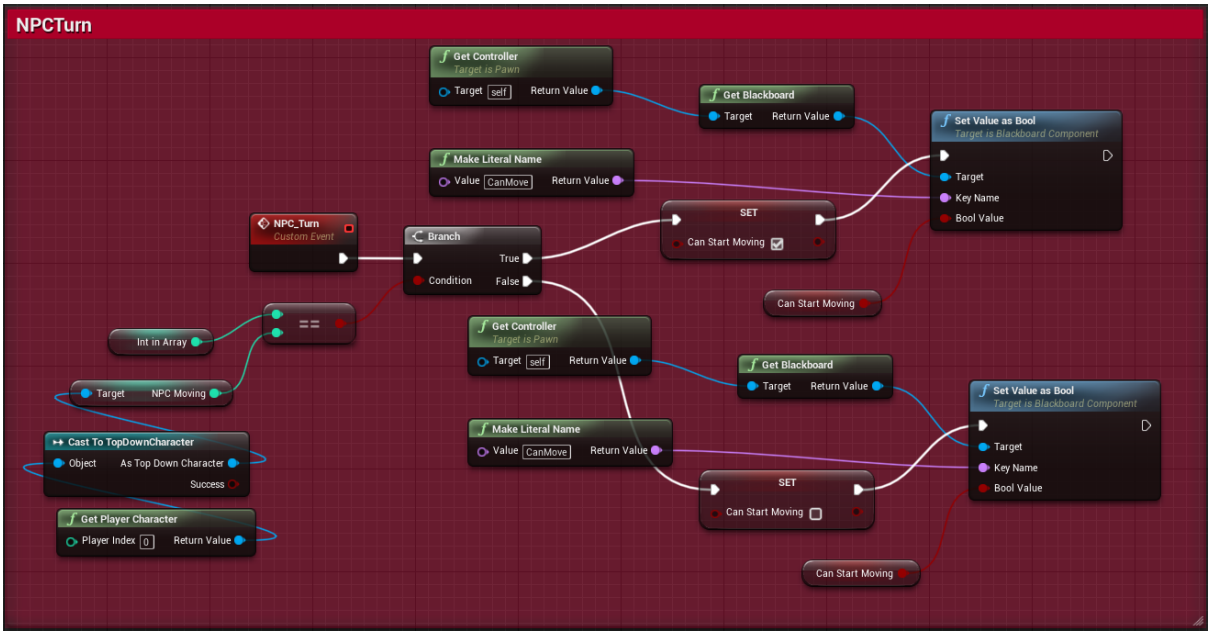


Figura 138: Captura de la funció NPCTurn

6.8.1.6. Reset

Aquesta funció (veure Figura 139) estableix les variables que gestionen el torn. Es fa de la següent manera:

- La variable ActualActions es guarda a 0.

- La variable MyTurn es posa en true i es passa a la Blackboard.

- Es modifica la variable de NPC_Moving i es guarda a 0.

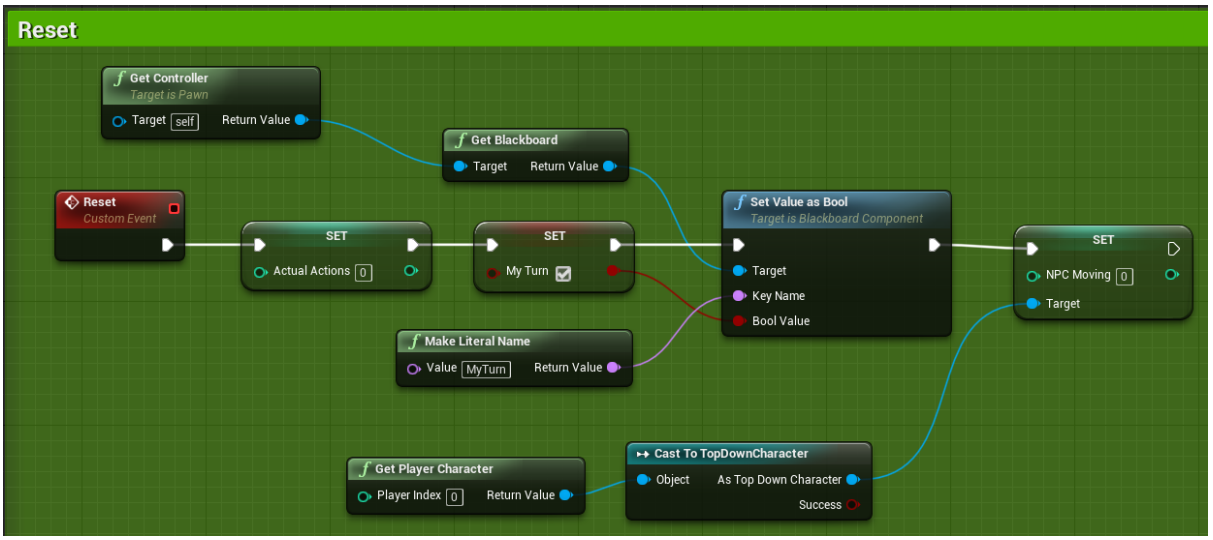


Figura 139: Captura de la funció Reset

6.8.1.7. ActionDone

Aquesta funció (veure Figura 140), s'encarrega de detectar quan es fa una acció i actuar en conseqüència. Aquesta funció fa el següent:

- Modifica el text referent a les accions perquè mostri el nombre d'accions fetes aquest torn.

- Si el nombre d'accions és igual a dos, es modifica la variable MyTurn a false i es passa a la Blackboard, s'afegeix la variable MyTurn a un array de booleans, es crida a la funció OnCover i s'incrementa en un la variable NPC_Moving.

- En cas de ser menor que 2, se li suma una acció i es torna a cridar a la funció OnCover.

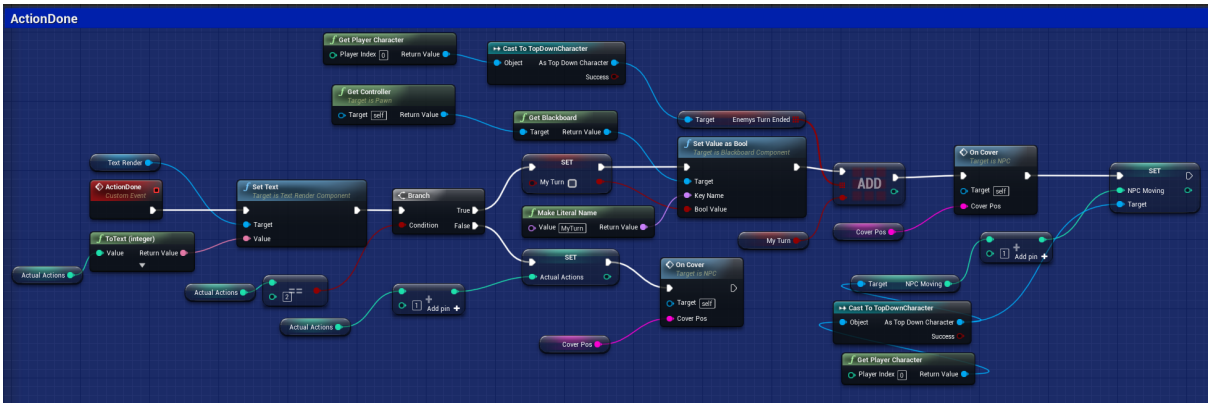


Figura 140: Captura de la funció ActionDone

6.8.1.8. Attack

Aquesta funció (veure Figura 141) s'encarrega de detectar si l'enemic pot atacar a un personatge o no. Aquesta funció fa el següent:

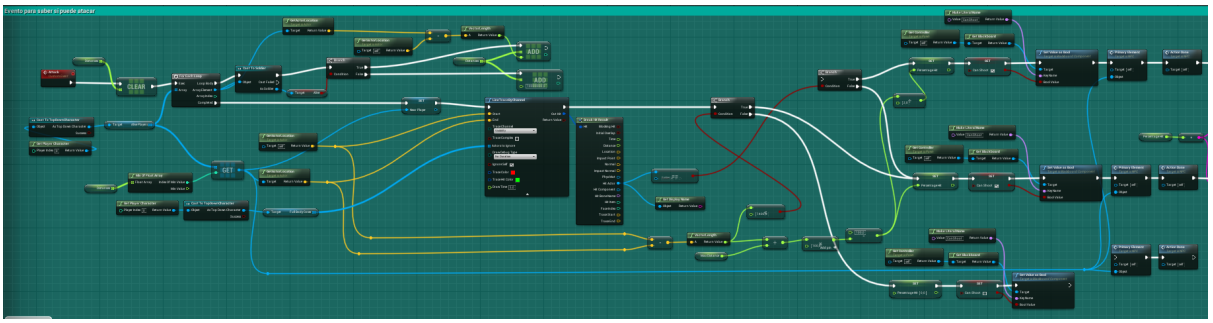


Figura 141: Captura de la funció Attack

- Buida l'array de floats anomenat Distancies.

- Mitjançant un bucle que recorre la llista d'actors dels personatges, mira si el personatge està viu. En el cas d'estar-ho, es guarda la distància entre l'un i l'altre. Veure Figura 142.

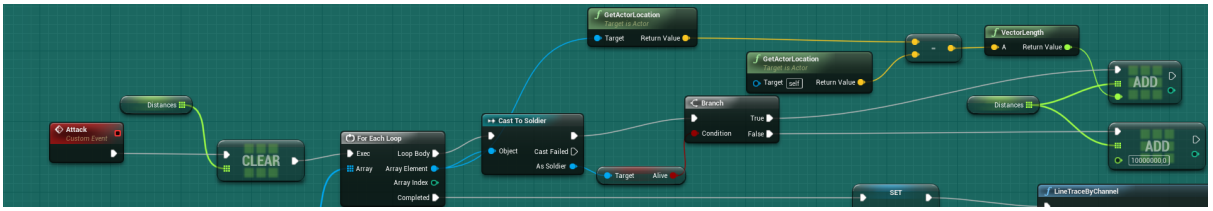


Figura 142: Captura de la funció Attack 1

- Un cop acabat, es mira quin és el valor més petit, i com l'output dona també l'índex, amb aquest índex busquem l'actor del personatge més proper i el guardem a la variable NearPlayer. Veure Figura 143.

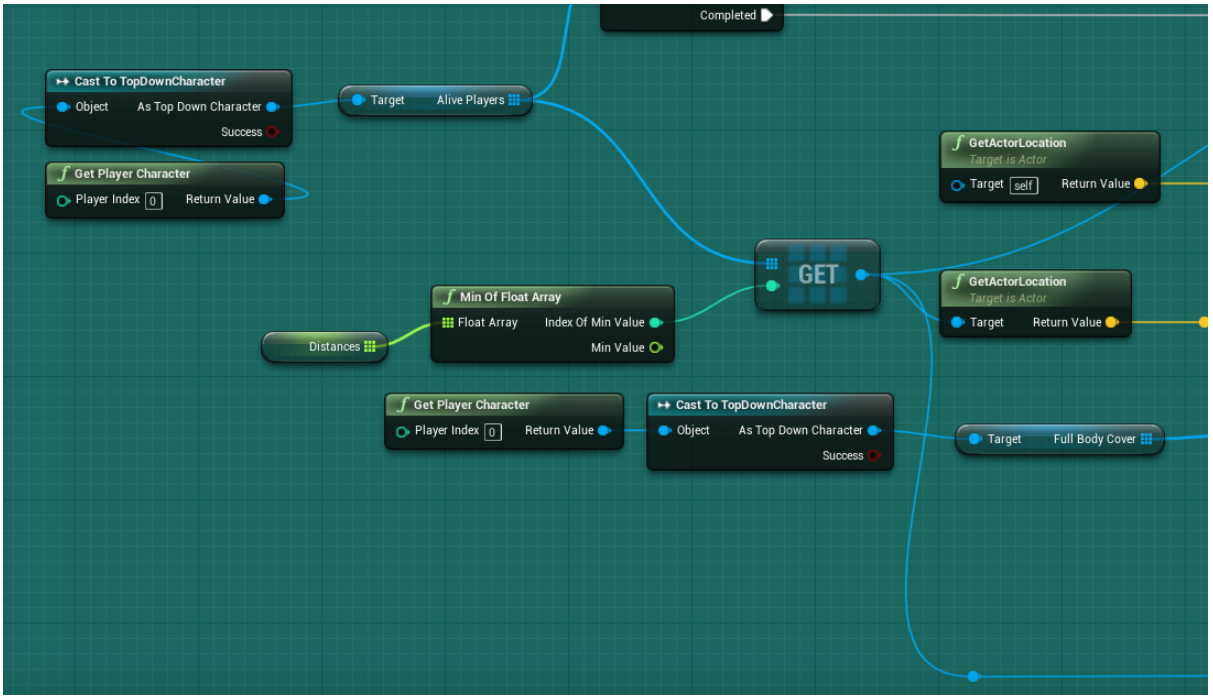


Figura 143: Captura de la funció Attack 2

- Es fa un LineTrace entre l'enemic (aquest actor) i el personatge més proper. Veure Figura 144.

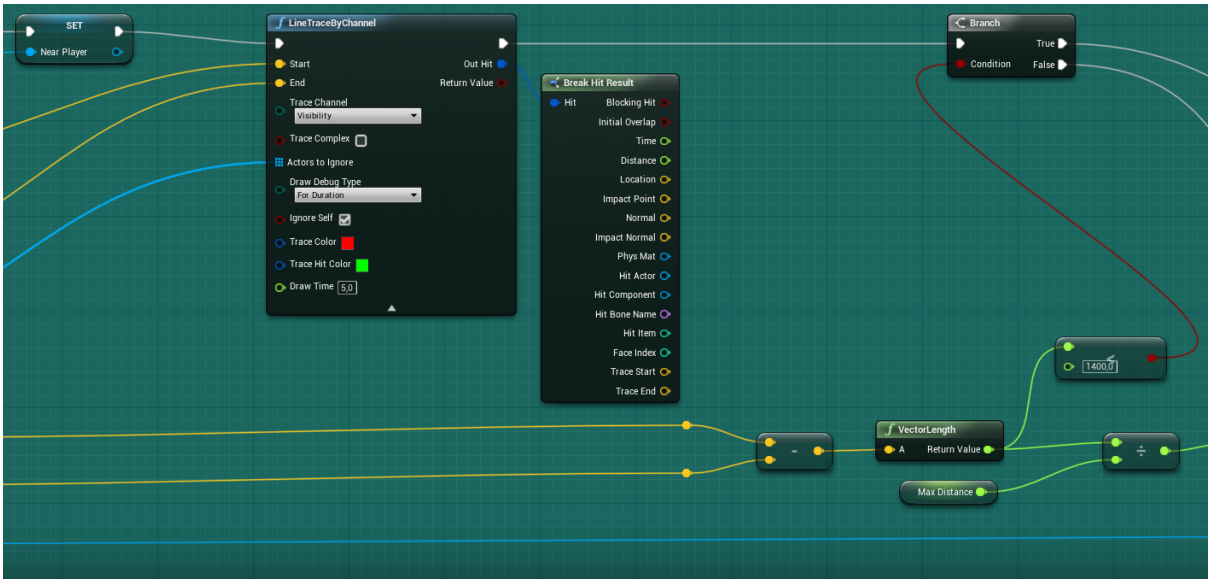


Figura 144: Captura de la funció Attack 3

- Es mira si la distància és superior a 1400, en cas de ser-ho, es calcula el percentatge d'encertar l'atac, es modifica la variable CanShot i es passa a la Blackboard.

- En cas de no ser-ho, es modifica el percentatge d'encertar l'atac a 0, es modifica la variable CanShot i es pasa a la Blackboard. Veure Figura 145.

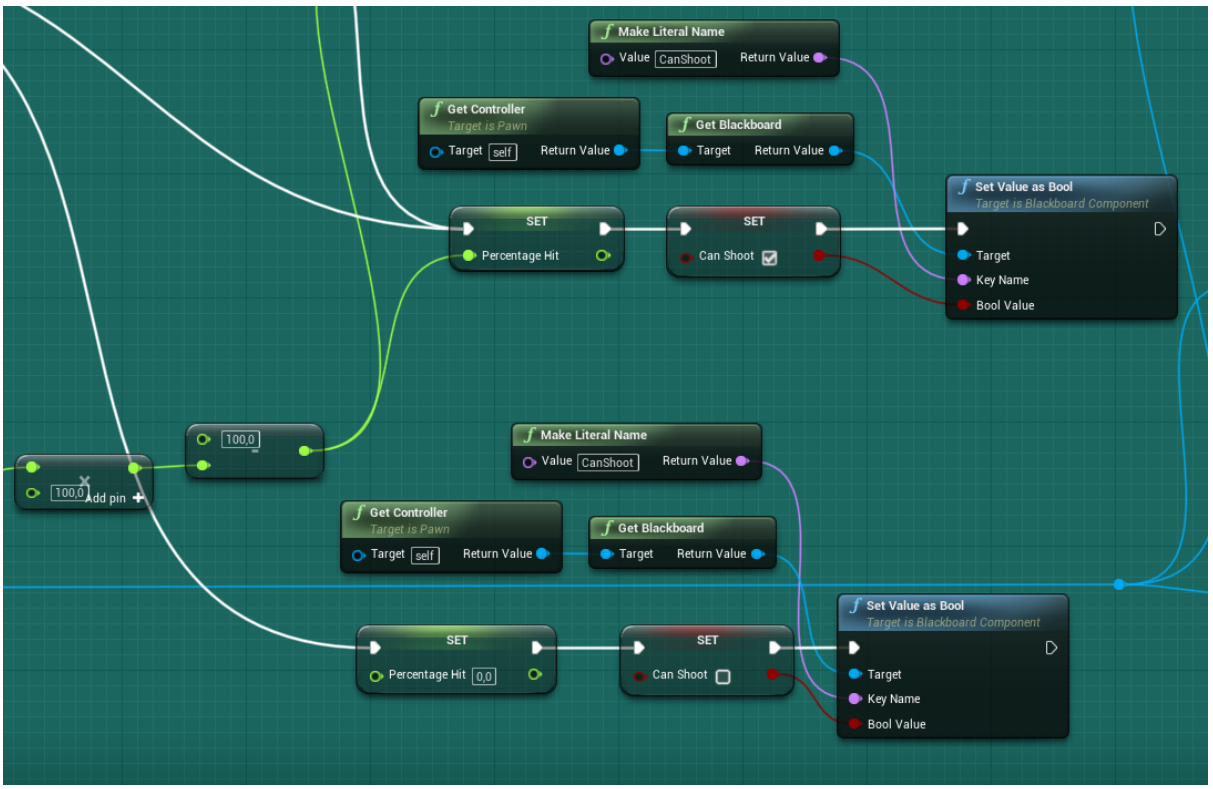


Figura 145: Captura de la funció Attack 4

6.8.1.9. Shooting

Aquesta funció (veure Figura 146) s'encarrega de calcular si encerta l'atac o no. Aquesta funció fa el següent:

- Calcula un número random entre 0 i 100.

- Compara si aquest número és superior al percentatge d'encertar l'atac.

- En cas de ser-ho, l'enemic falla l'atac.

- En cas de no ser-ho, s'agafa mitjançant un random integer el Spell que utilitzarà l'enemic i es mira a quins jugadors està flanquejant l'enemic, mitjançant un loop.

- Mitjançant la variable NearPlayer, es crida la funció del personatge GetHit, i se li passa la variable Encontrado que detecta si el jugador al que ataca està sent flanquejat per aquest enemic i l'id del Spell que ha utilitzat per atacar-lo.

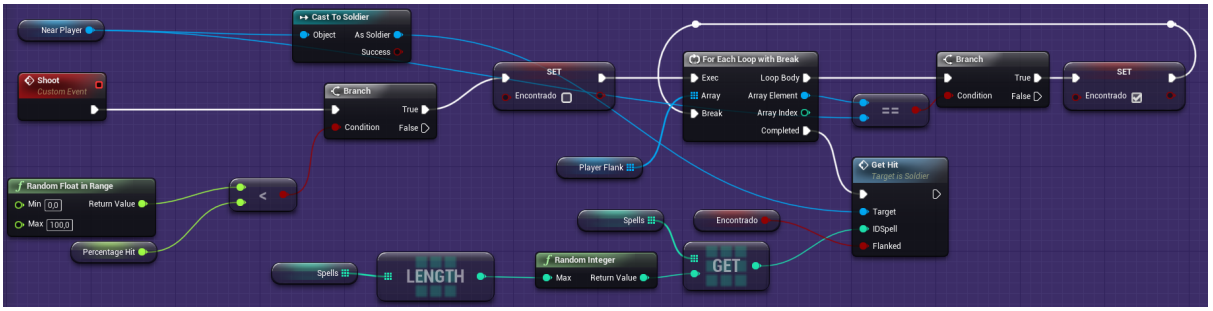


Figura 146: Captura de la funció Shoot

6.8.1.10. OnCover

Aquesta funció (veure Figura 147) serveix per saber a quins personatges està flanquejant i fa el següent:

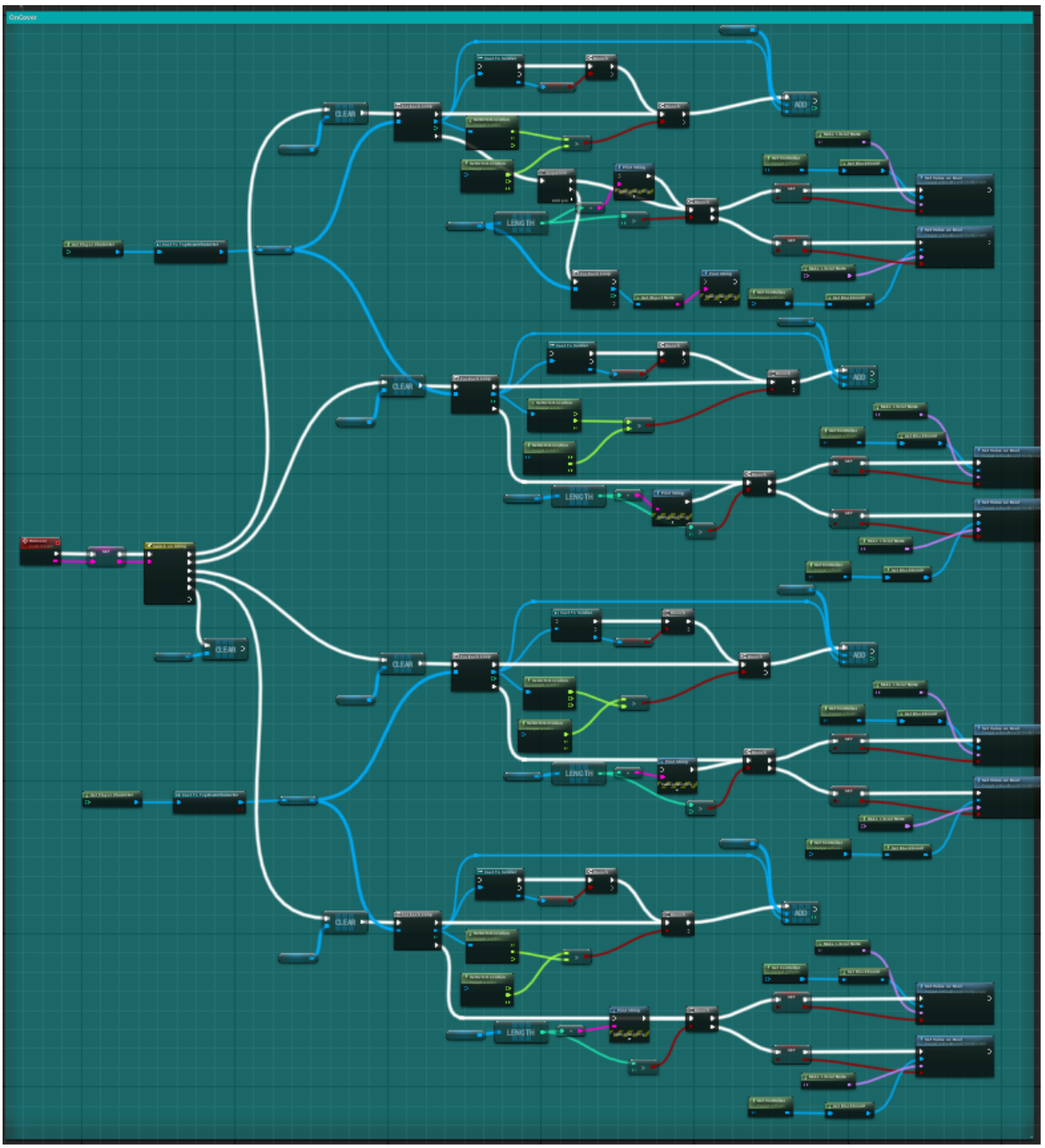


Figura 147: Captura de la funció OnCover

- Es guarda la posició de la cobertura en la qual es troba.

- Mitjançant un switch de string detecta a quina posició de la cobertura es troba. Veure Figura 148.

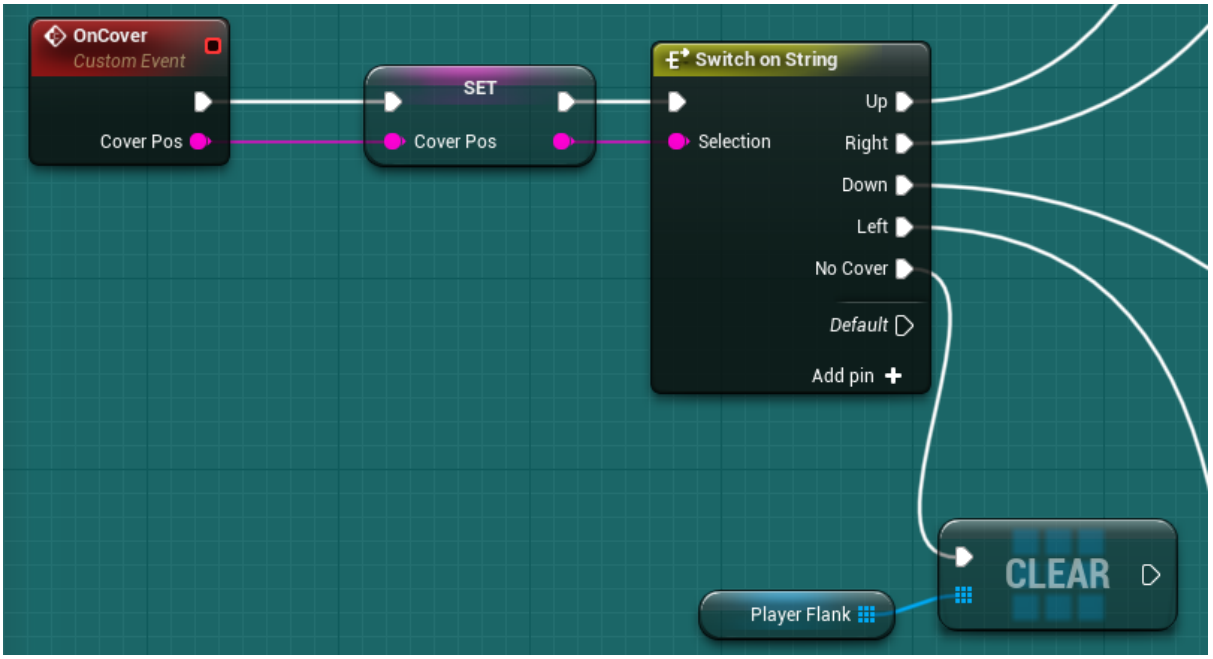


Figura 148: Captura de la funció OnCover 1

- Buida l'array d'actors PlayerFlank.

- Per cada personatge viu, compara la posició amb la de l'enemic i detecta si l'està flanquejant o no.

- En cas de flanquejar-lo, agrega a l'actor del personatge a l'array d'actors dels personatges flanquejats. Veure Figures 149-152.

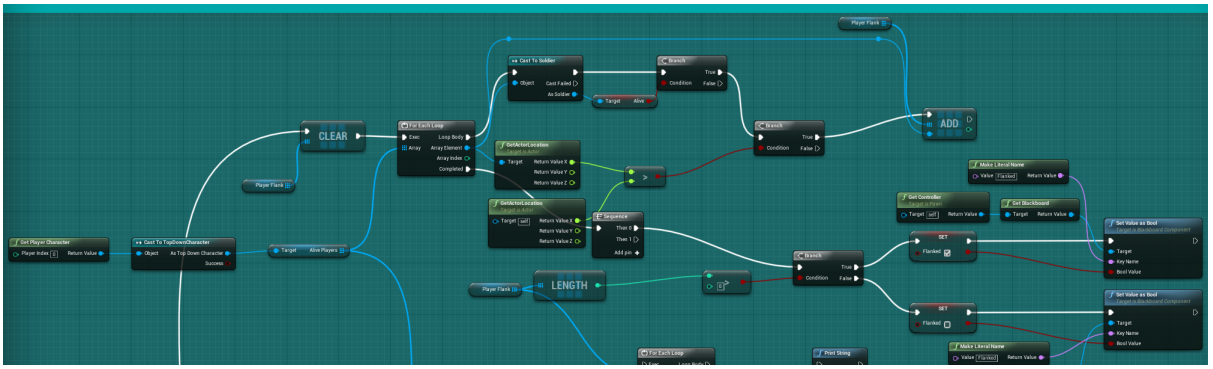


Figura 149: Captura de la funció OnCover 2

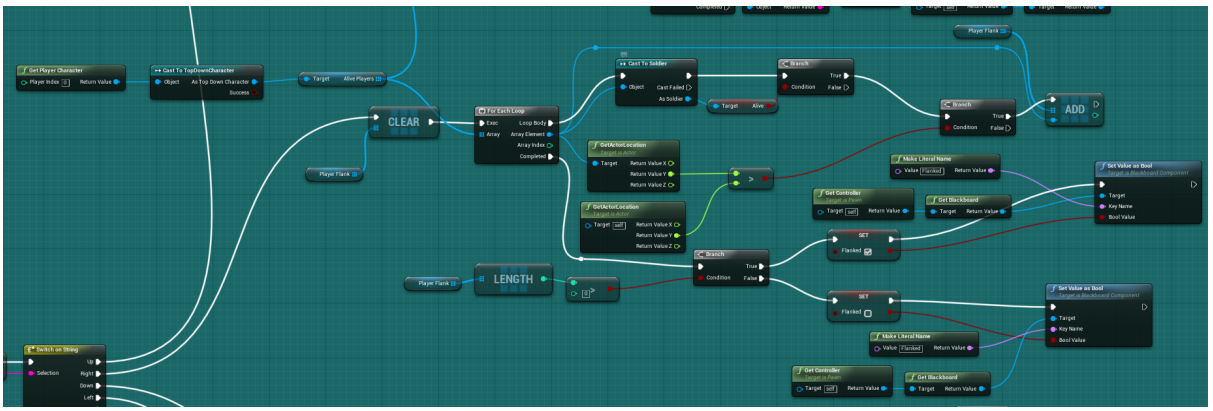


Figura 150: Captura de la funció OnCover 3

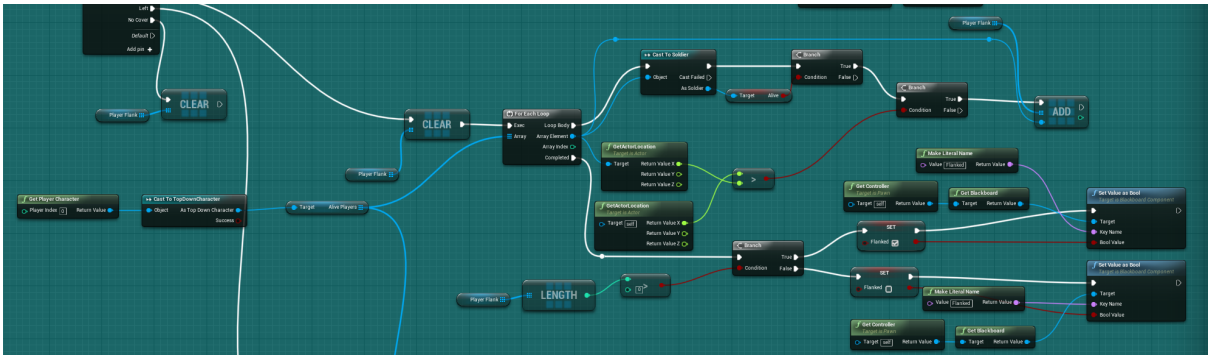


Figura 151: Captura de la funció OnCover 4

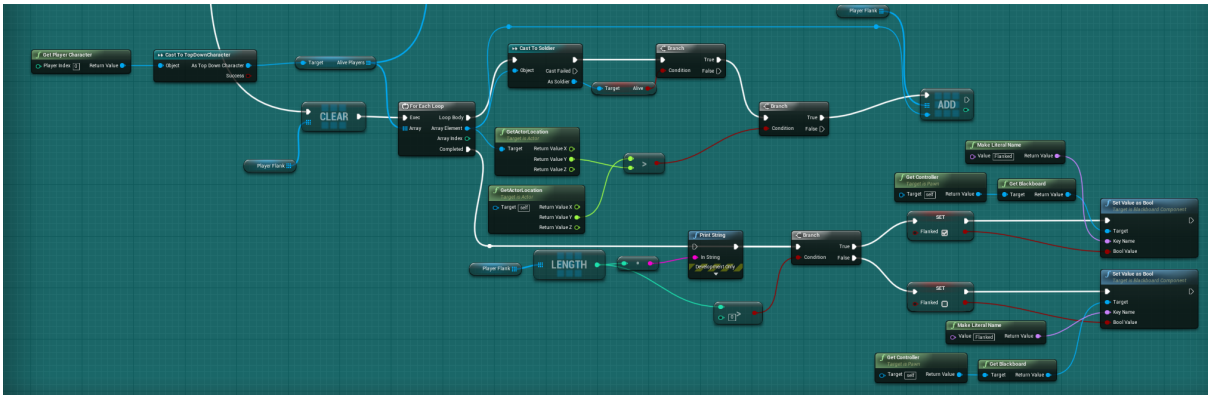


Figura 152: Captura de la funció OnCover 5

6.8.1.11. GetDamaged

Aquesta funció (veure Figures 153) és pràcticament igual a la funció de GetHit del personatge. És l'encarregada de gestionar i calcular el mal que rep l'enemic i la funció de DeleteEmitter (veure figura 121) s'encarrega d'eliminar el FX passat cert temps. Fa el següent:

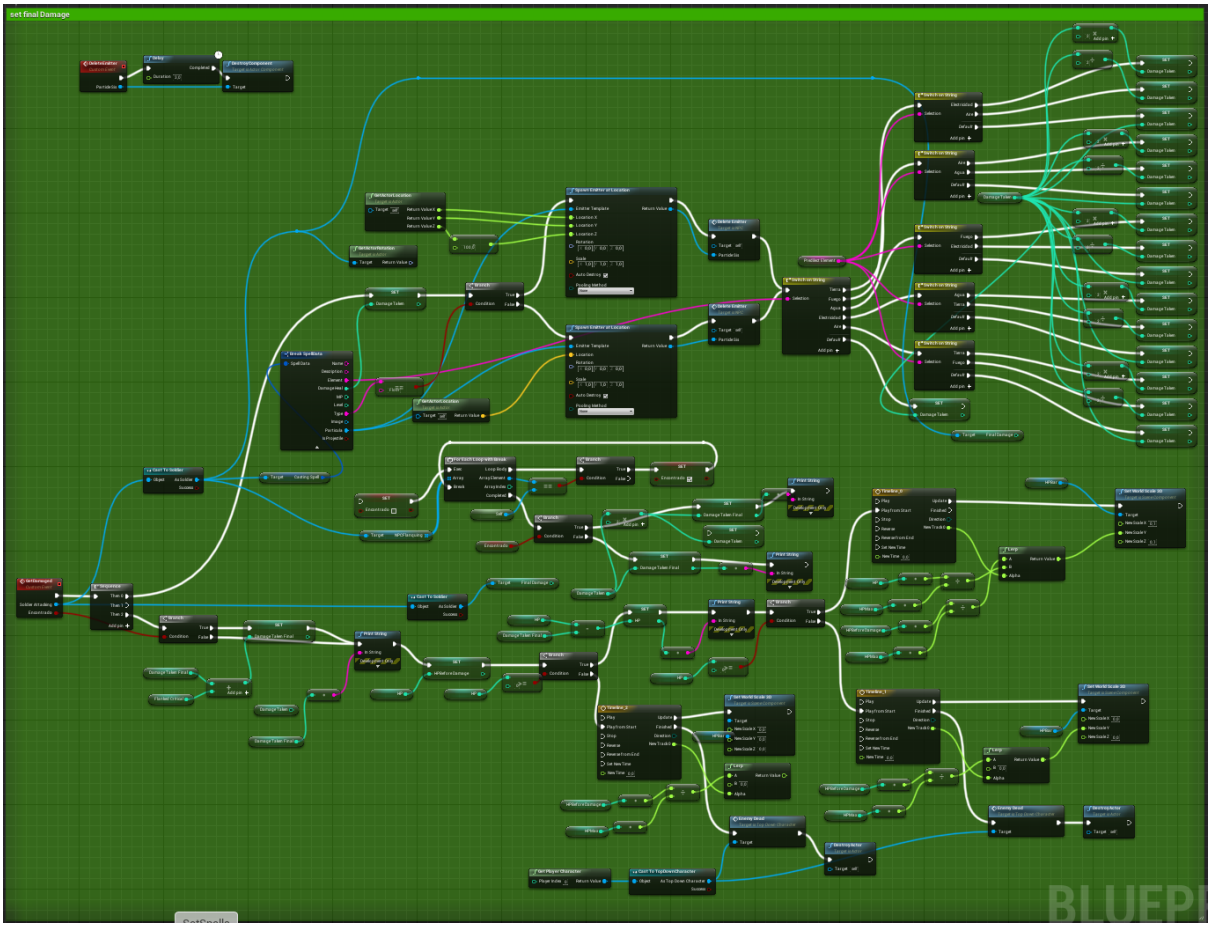


Figura 153: Captura de la funció GetDamaged

- A partir de l'actor del soldat atacant, agafem el Spell que està utilitzant. Veure Figura 154.

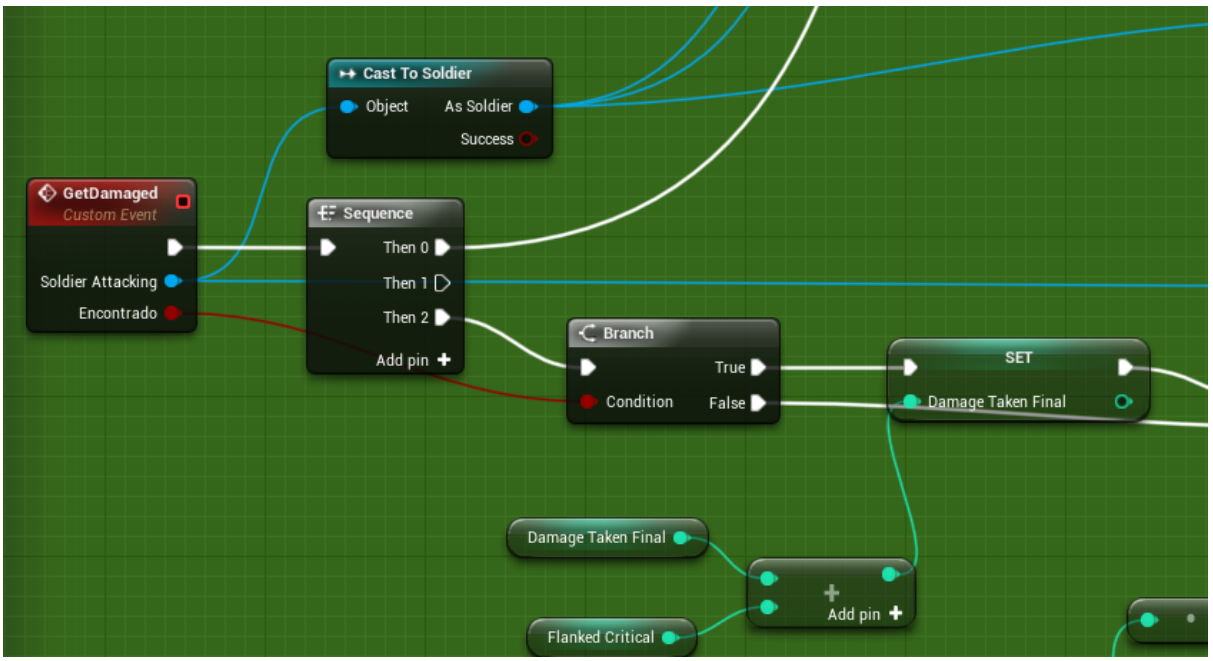


Figura 154: Captura de la funció GetDamaged 1

- Ens guardem el mal del Spell dins la variable “DamageTaken”.

- Mitjançat un branch, mirem quin tipus d’atac és el Spell.

- Depenent del tipus de Spell, creem un Emitter a una posició (centre del personatge) o a un altre (als peus del personatge).

- Cridem la funció DeleteEmitter. Veure Figura 155.

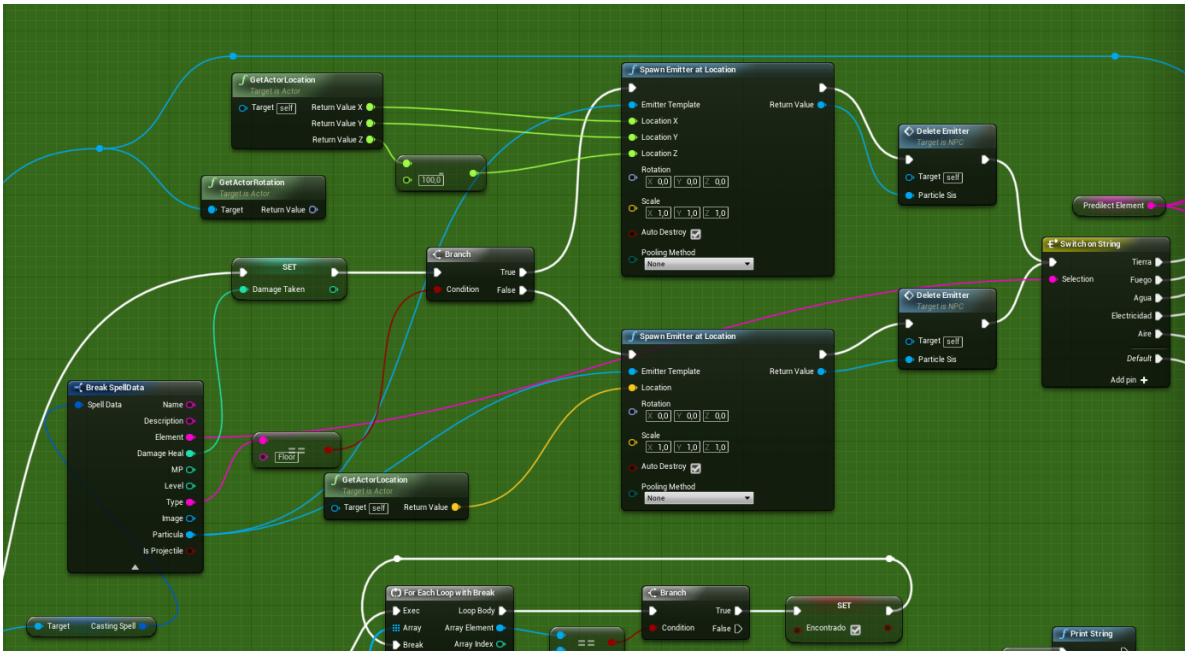


Figura 155: Captura de la funció GetDamaged 3

- Fem un switch en string a partir de la variable element del Spell.

- Per cada element del switch, fem un altre switch on string, però només dels dos elements que modifiquen el multiplicador del mal utilitzant la variable del personatge anomenada “Predilect Element”.

- Apliquem el multiplicador del mal a la variable “DamageTaken”. Veure Figura 156.

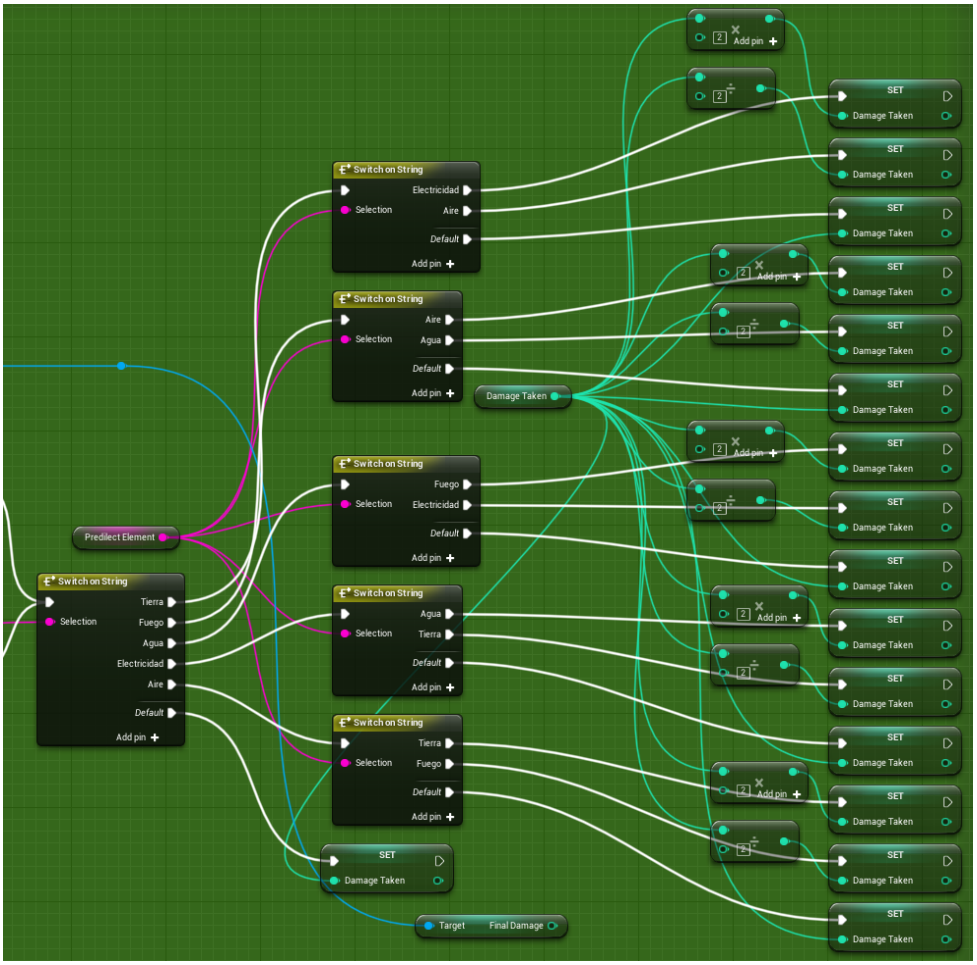


Figura 156: Captura de la funció GetDamaged 5

- Passem a la part 2 del sequence, on es mira si el personatge ha estat flanquejat o no mitjançant la variable bool passada per referència.

- Sumem la variable “FlankedCritical” a la variable “Damage Taken”.

- Restem la variable “DamageTaken” a la vida actual del personatge.

- Si la vida és més gran de 0, modifiquem la mesh de la barra de vida utilitzant l’update d’un timeline, amb un Lerp. Al Lerp li hem de posar a la variable A vida que es tenia abans del cop dividit la vida màxima, i a la variable B, la vida actual dividit la vida màxima, i a la variable alpha li hem de posar l’output del

timeline. D'aquesta manera la barra de vida fa una animació quan rep mal.

- Si la vida és inferior a 0, modifica la mesh de la barra de vida amb una timeline com abans i crida a la funció PlayerDead del gestor. Veure Figura 157.

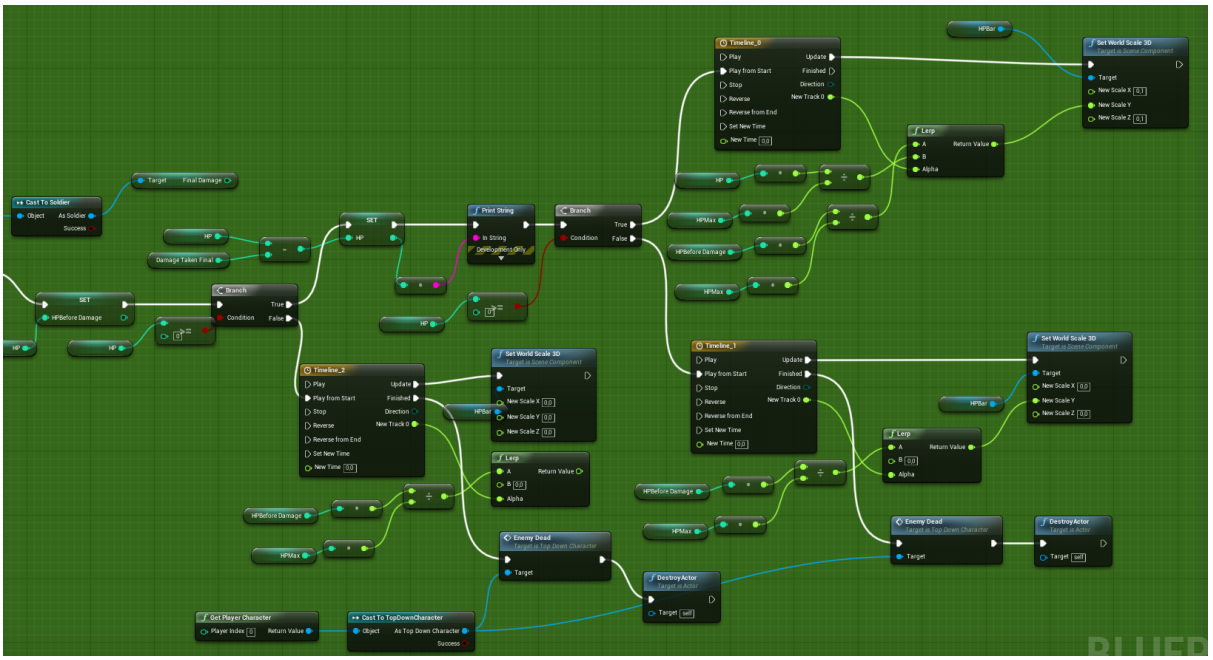


Figura 157: Captura de la funció GetDamaged 6

6.8.1.12. GoBetterPos

Aquesta funció (veure Figura 158) serveix per detectar si l'enemic pot moure's a una millor posició. Fa el següent:

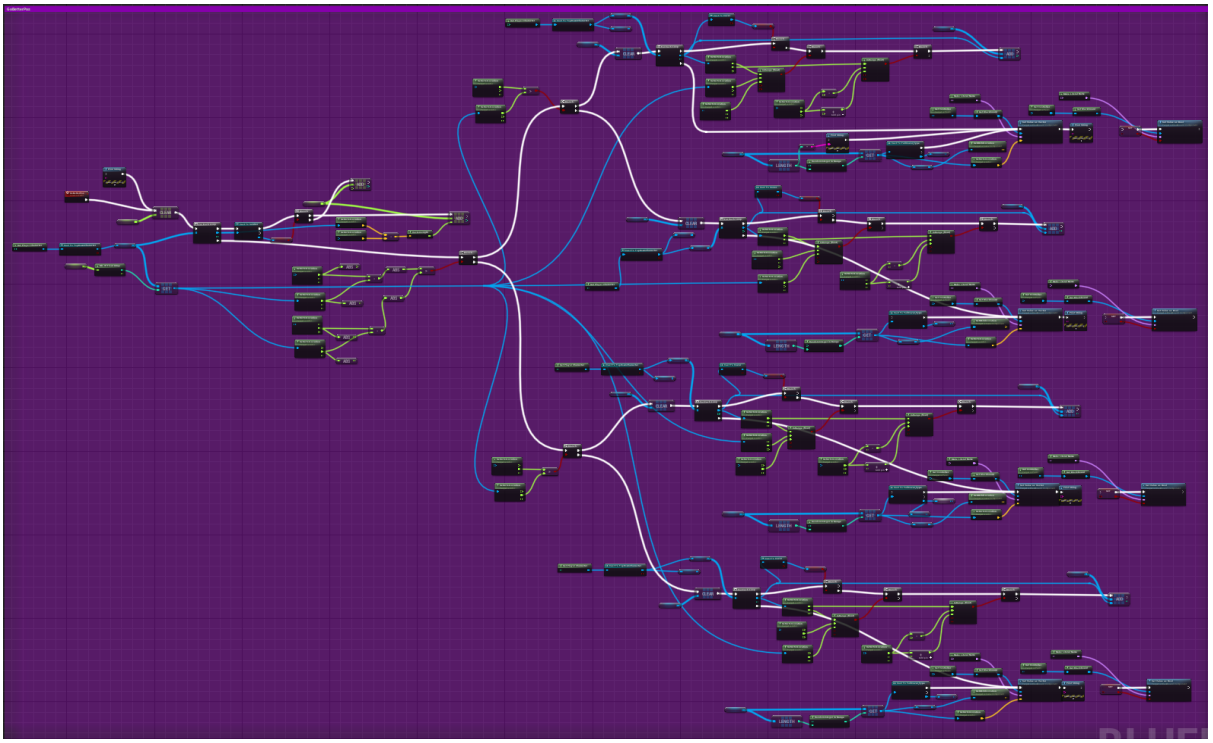


Figura 158: Captura de la funció GoBetterPos

- Buida l'array de distàncies.

- Per cada personatge, mira si està viu calcula la distància entre l'enemic i aquell personatge.

- Un cop acabat, mira la variable més petita dins de l'array i obté l'índex.

- Amb aquest índex, obté l'actor del personatge i calcula si la distància x és més gran que la distància y entre aquests dos actors (personatge més a prop i l'enemic).

- Si la distància més gran és la x, podem considerar que el personatge està més a prop dels valors Up/Down, que dels valors Right/Left. Veure Figura 159.

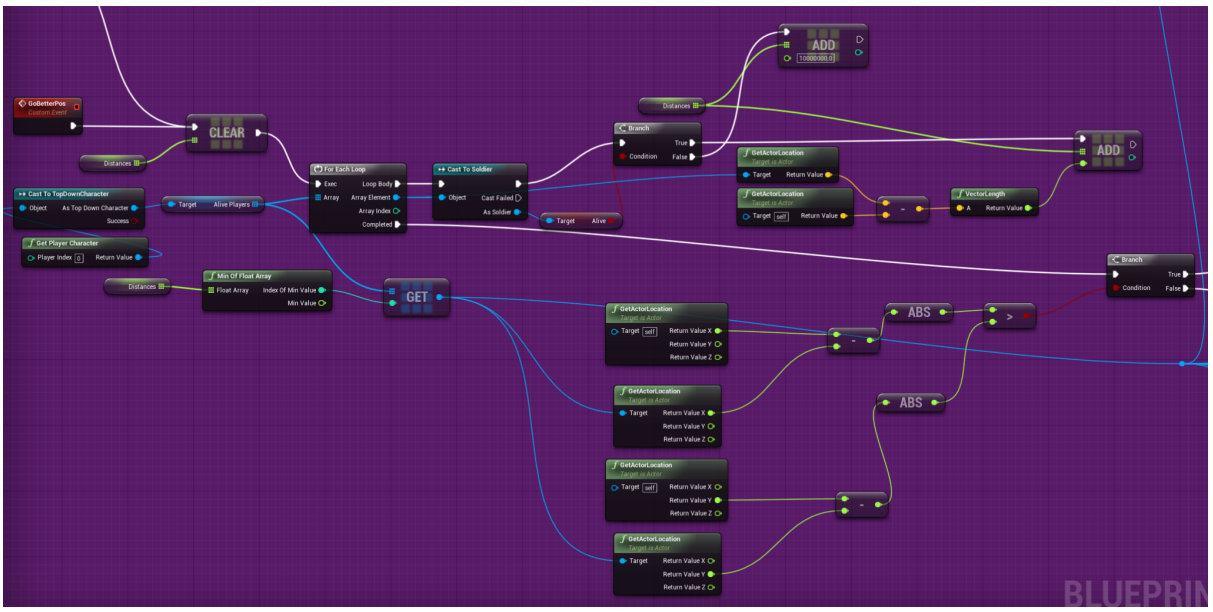


Figura 159: Captura de la funció GoBetterPos 1

- Un cop sabent això, comparem els valors x del personatge i de l'enemic per saber si està davant (Up) o darrere (Down). Veure Figura 160.

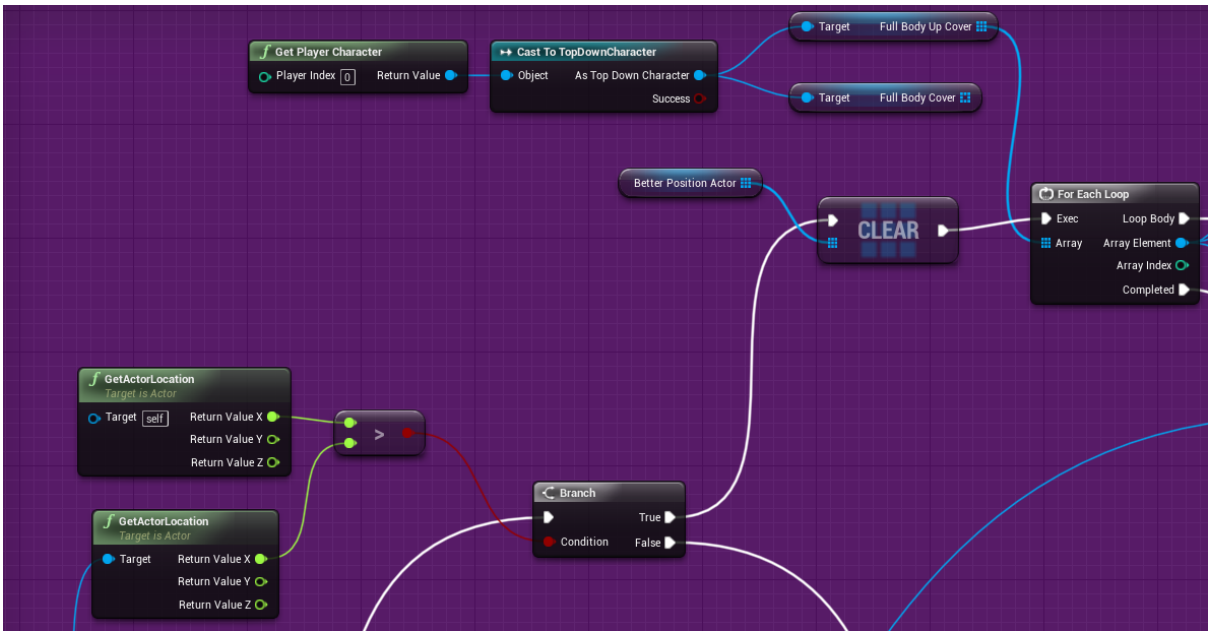


Figura 160: Captura de la funció GoBetterPos 2

- Buida l'array auxiliar d'actors anomenat BetterPositionActor i, mitjançant un bucle, mira cada cobertura que sigui d'aquella

posició (en el cas que estem comentant, es tractarien de les cobertures Up).

- Per cada cobertura, mira si està ocupat, si la posició en el valor buscat (en aquest cas el valor de la posició x) es troba entre el personatge i l'enemic.

- En cas de ser el cas, guarda l'actor a l'array d'actors BetterPositionActor.

- Un cop acabat el bucle, agafa un actor random de la llista de l'array.

- Obté el vector de la posició i l'envia a la Blackboard. Veure Figures 161-164.

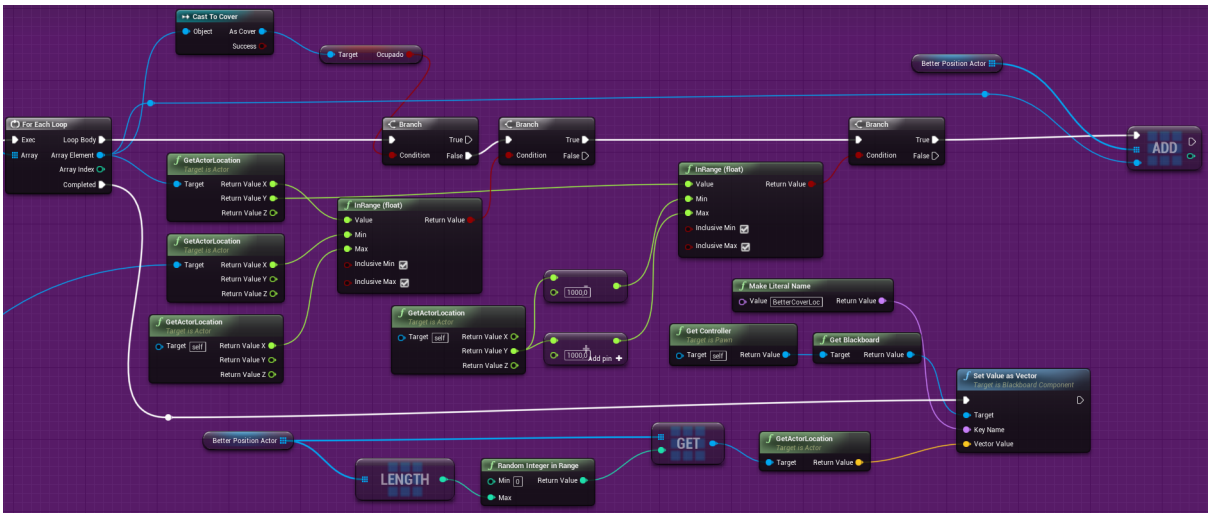


Figura 161: Captura de la funció GoBetterPos 3

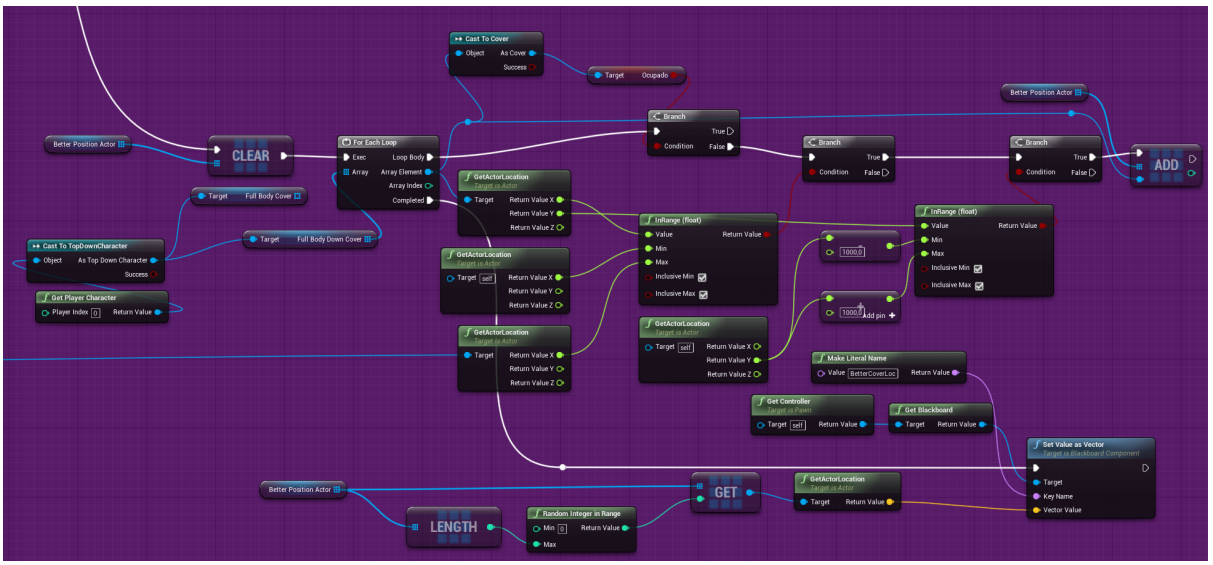


Figura 162: Captura de la funció GoBetterPos 4

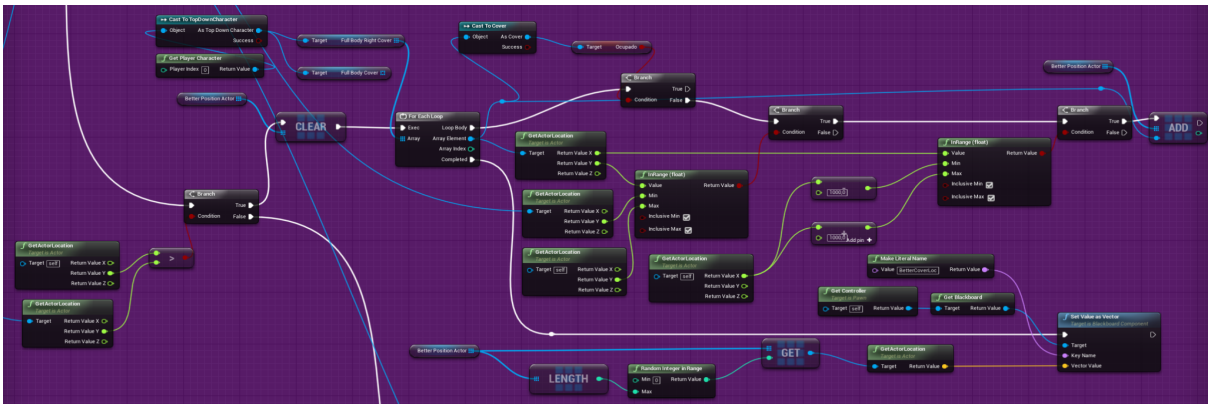


Figura 163: Captura de la funció GoBetterPos 5

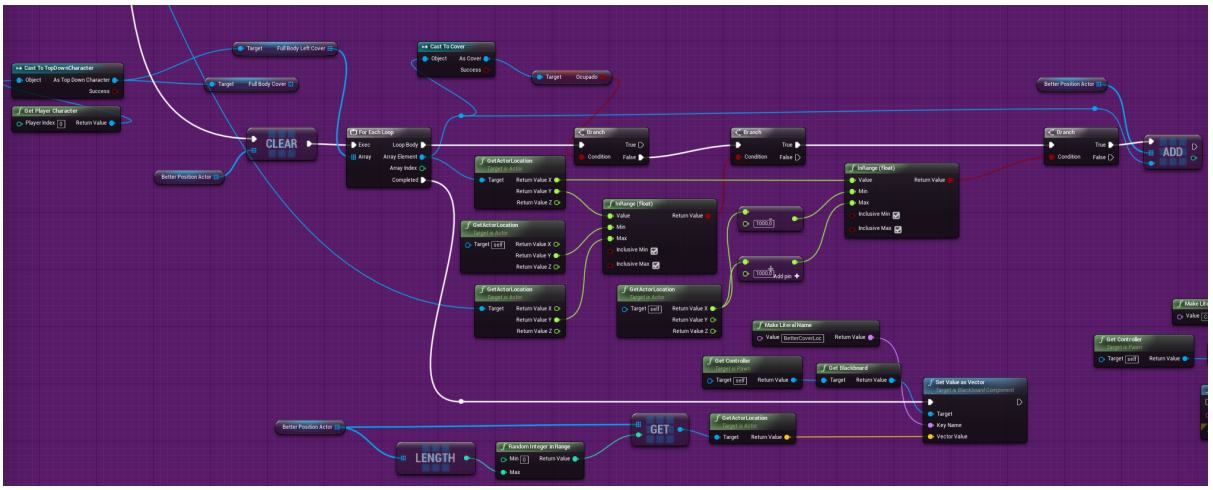


Figura 164: Captura de la funció GoBetterPos 6

6.8.1.13. SearchNearestCover

Aquesta funció (veure Figures 165-180) serveix per buscar la cobertura més propera en cas de no trobar-se en cap. Aquesta funció fa:

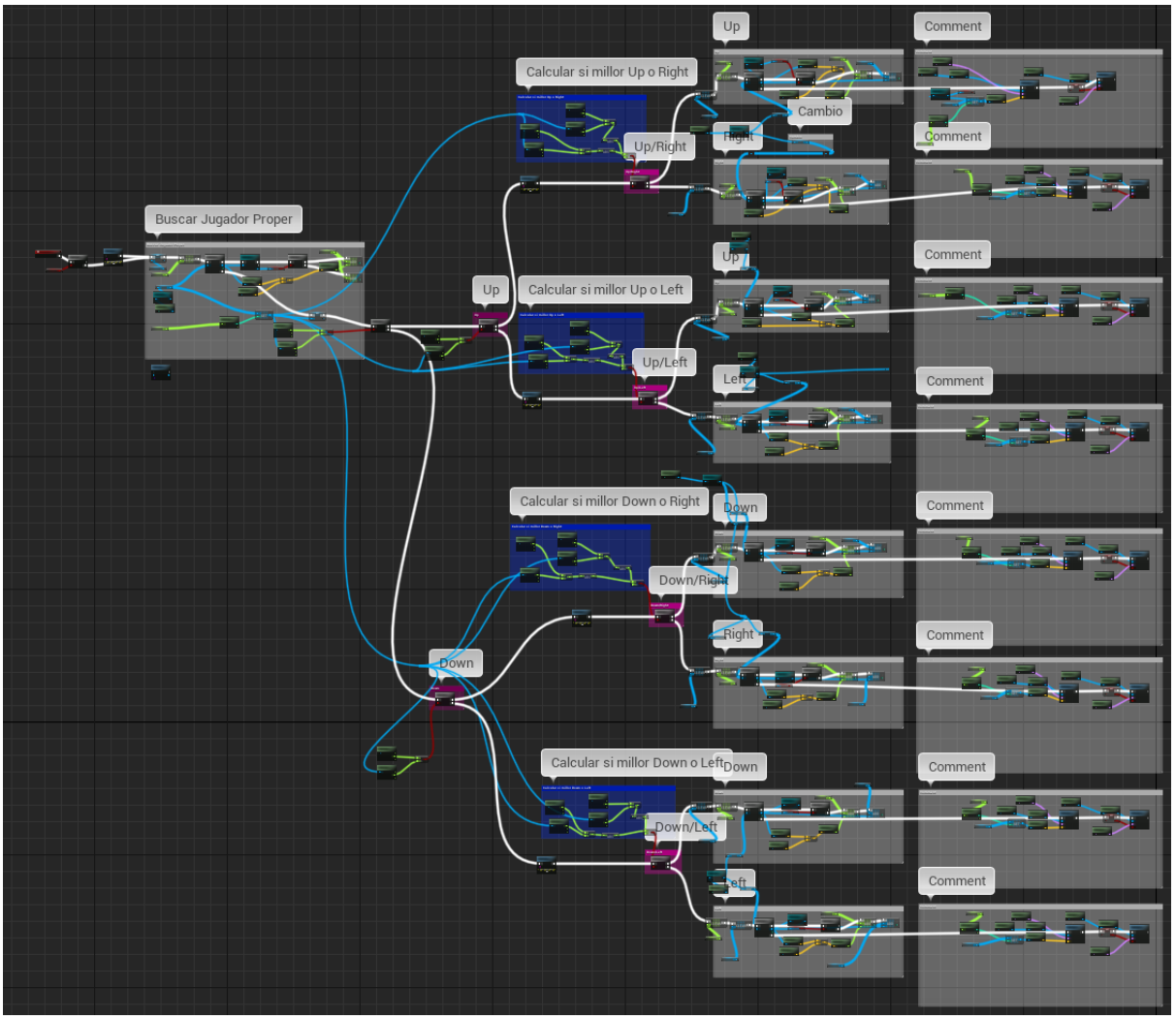


Figura 165: Captura de la funció SearchNearestCover

- Mira si l'enemic es troba en una cobertura.

- En cas negatiu, buida l'array de distàncies i borra l'actor de NearPlayer.

- Mitjançant l'array de personatges del gestor, utilitzant un bucle, busca el personatge viu més a prop de l'enemic. Veure Figura 166.

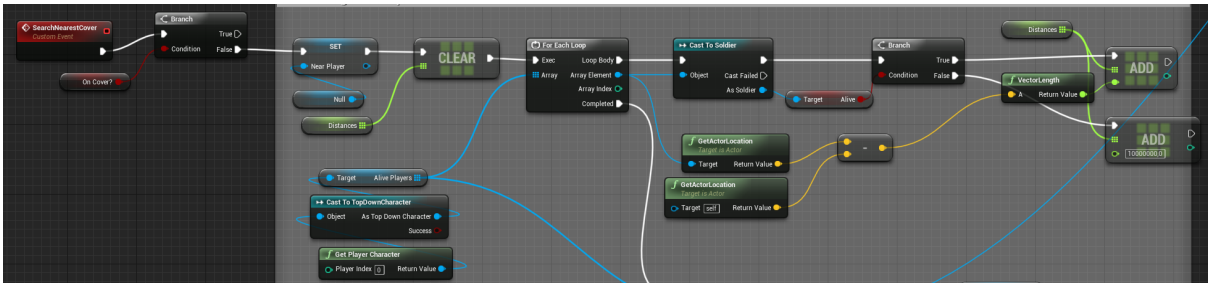


Figura 166: Captura de la funció SearchNearestCover 1

- Un cop acabat, agafa l'índex del valor més petit de l'array de distàncies i obté l'actor de l'array de personatges comentat anteriorment i el guarda a la variable NearPlayer.

- Similar a la funció anterior, un cop es té l'actor, compara les distàncies en el valor x per saber a quina direcció es troba. Veure Figures 167 i 168.

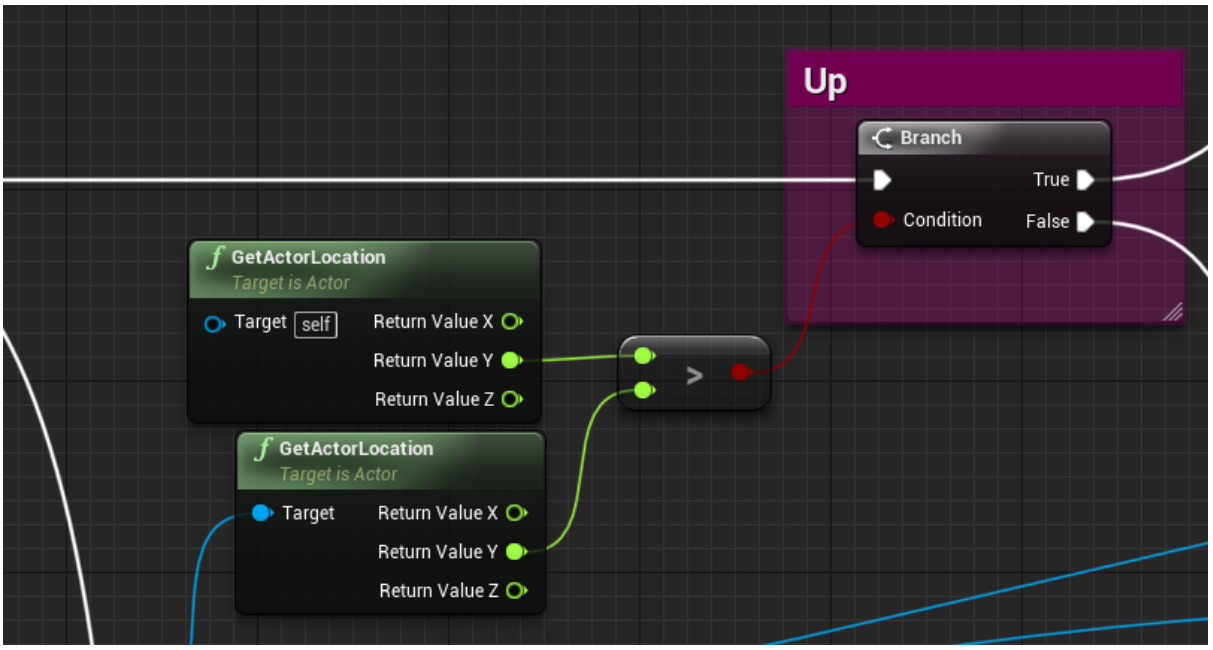


Figura 167: Captura de la funció SearchNearestCover 1

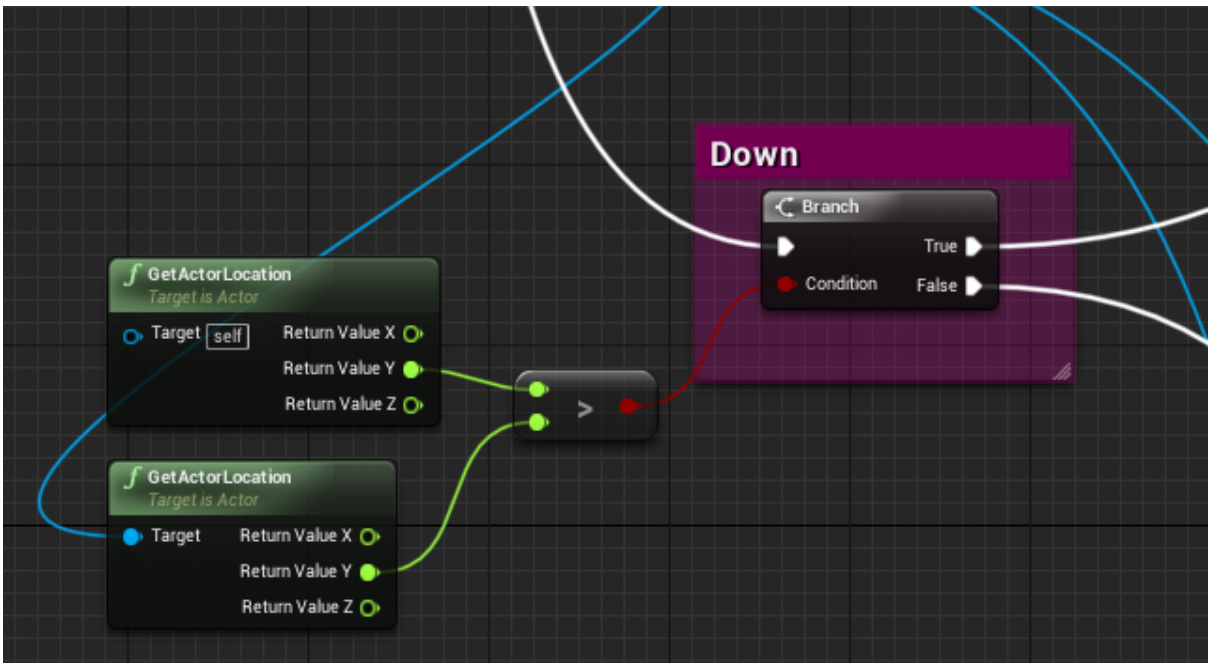


Figura 168: Captura de la funció SearchNearestCover 2

- Un cop comparant el valor x, es compara el valor Y. D'aquesta forma el troba la posició (Up/Right, Up/Left, Down/Right, Down/Left).

- Un cop sabem en quina posició es troba, compara les dues distàncies (per exemple, si és Up/Left, es compara la distància Up amb Left). Veure Figures 169-172.

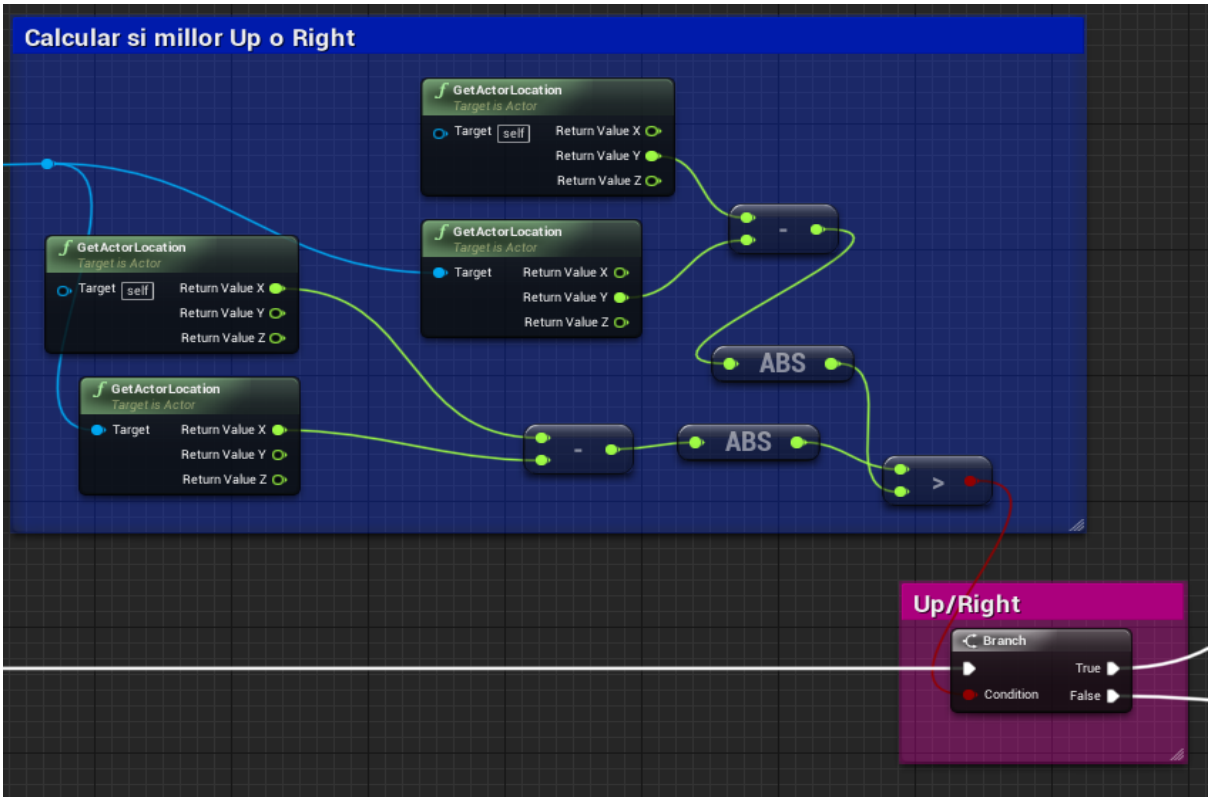


Figura 169: Captura de la funció SearchNearestCover 3

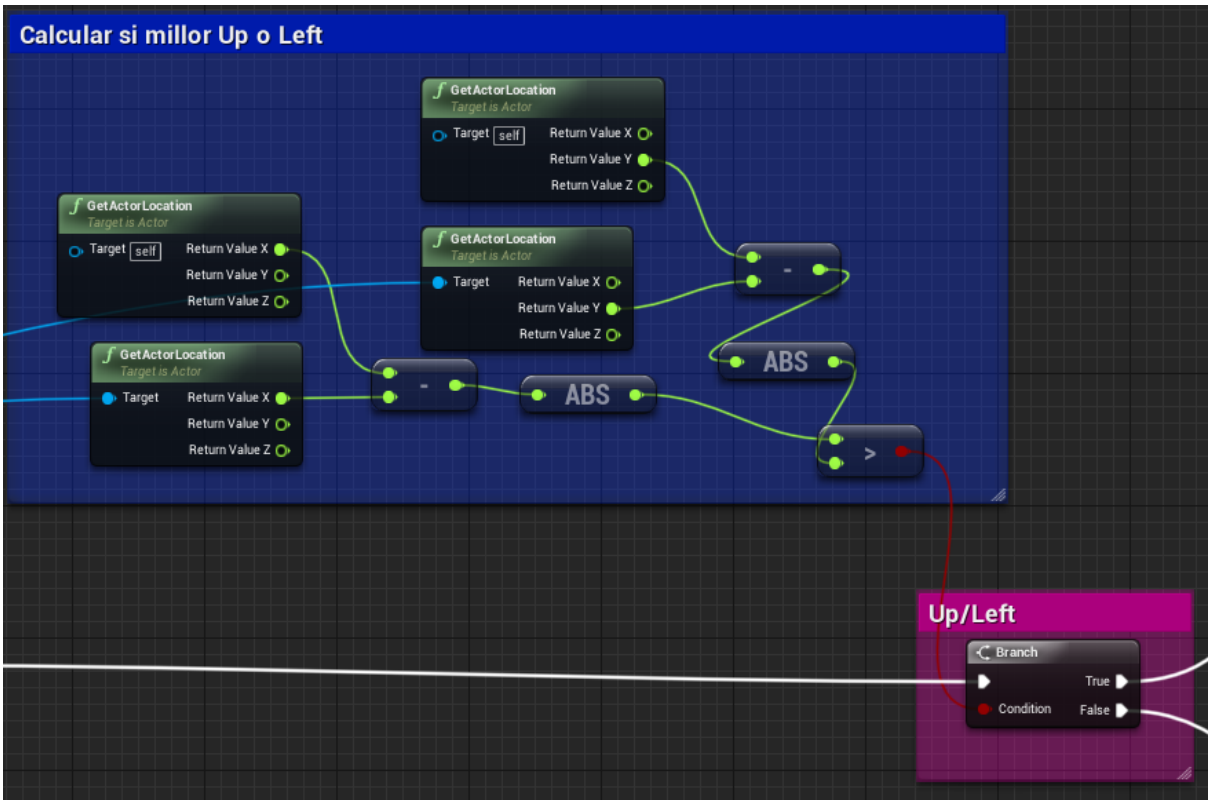


Figura 170: Captura de la funció SearchNearestCover 4

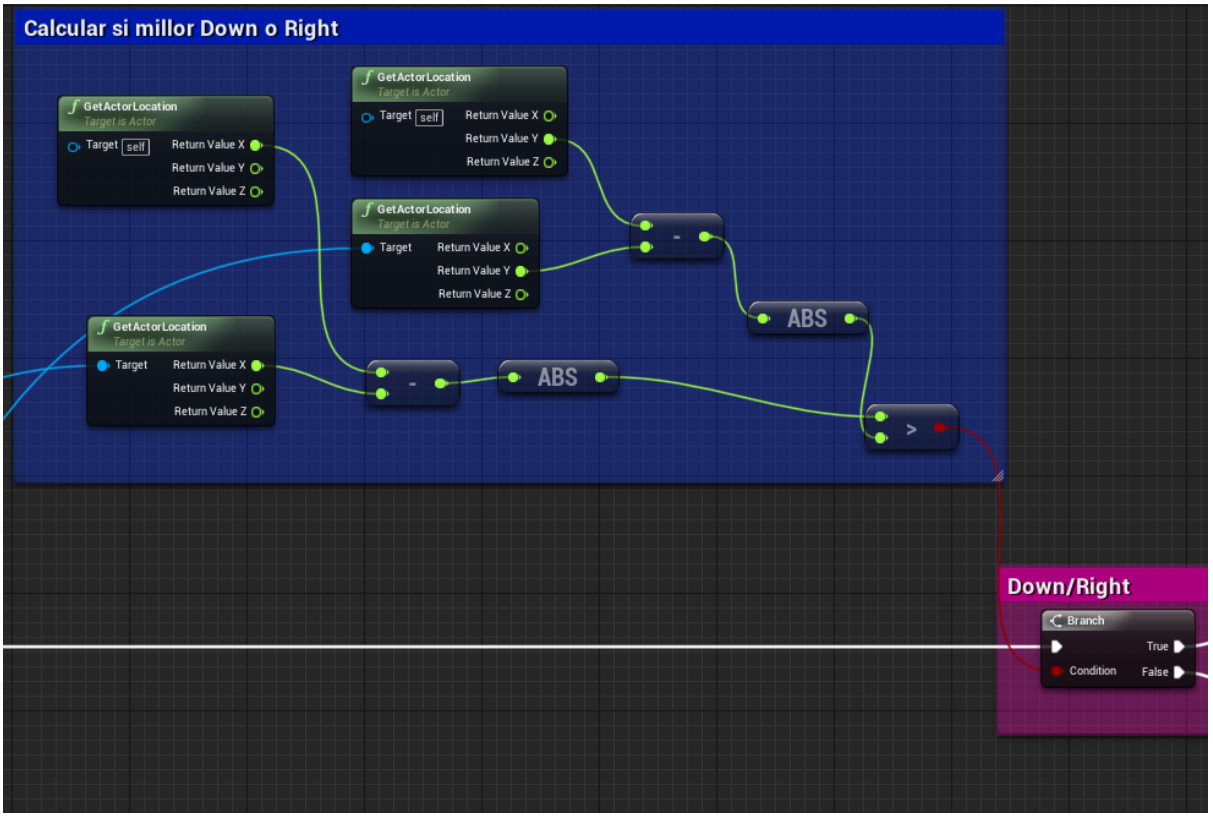


Figura 171: Captura de la funció SearchNearestCover 5

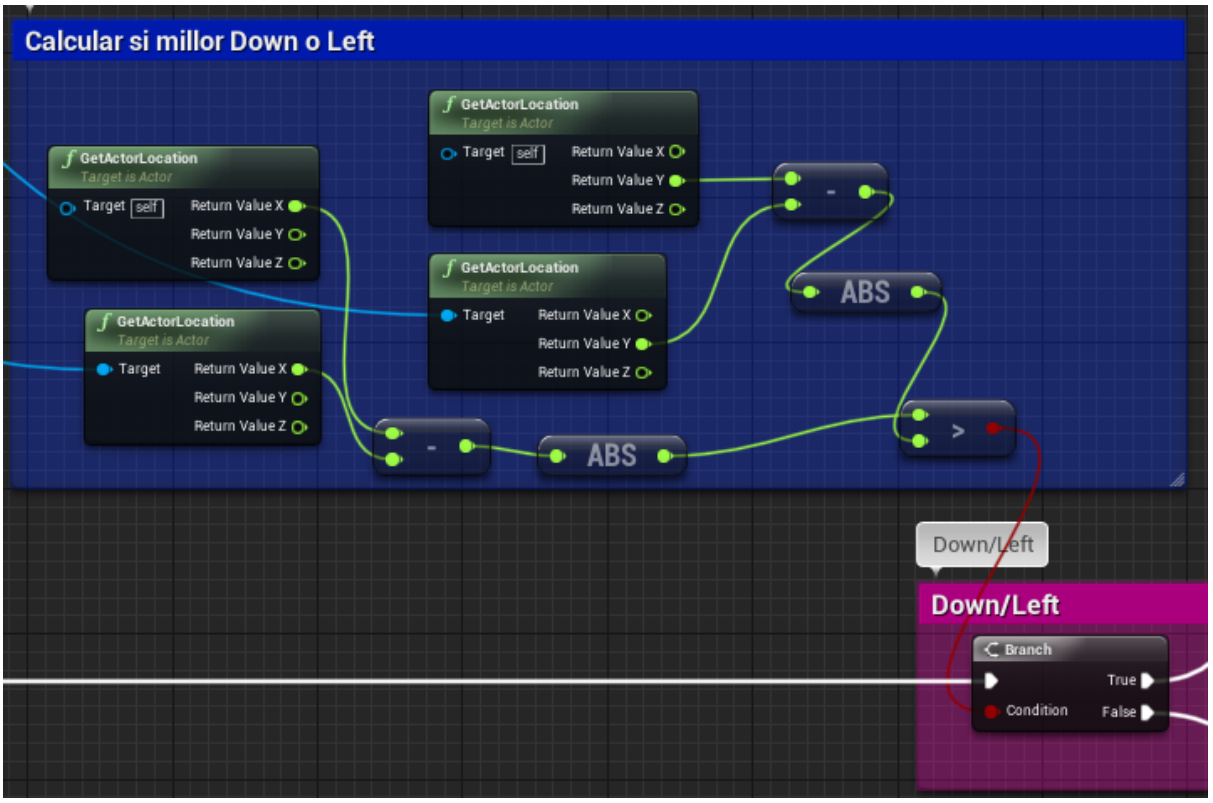


Figura 172: Captura de la funció SearchNearestCover 6

- Un cop tenint el valor més gran, agafa l'array de cobertures en aquella posició del gestor de personatges i, mitjançant un bucle, guarda les distàncies de l'enemic i les cobertures i es guarda en un array, juntament amb l'actor de la cobertura.

- Un cop acabat, s'agafa el valor més petit de l'array de distàncies entre l'enemic i les cobertures.

- Un cop trobat, s'envia la posició de la cobertura a la Blackboard. Veure Figures 173-180.

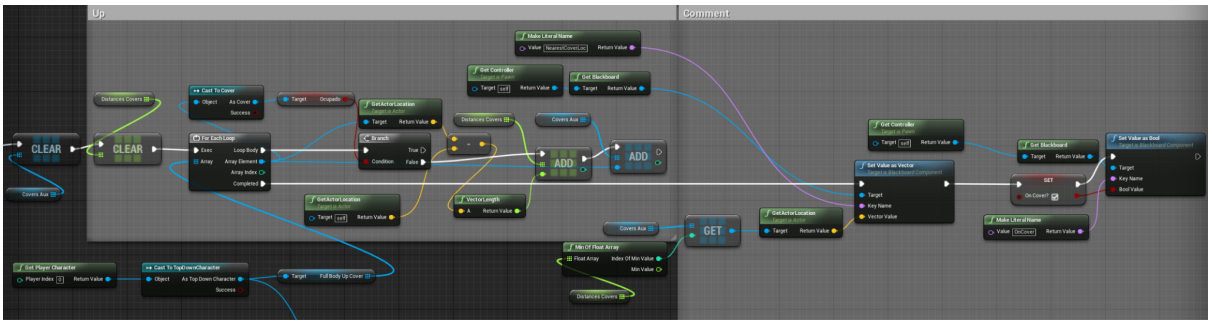


Figura 173: Captura de la funció SearchNearestCover 7

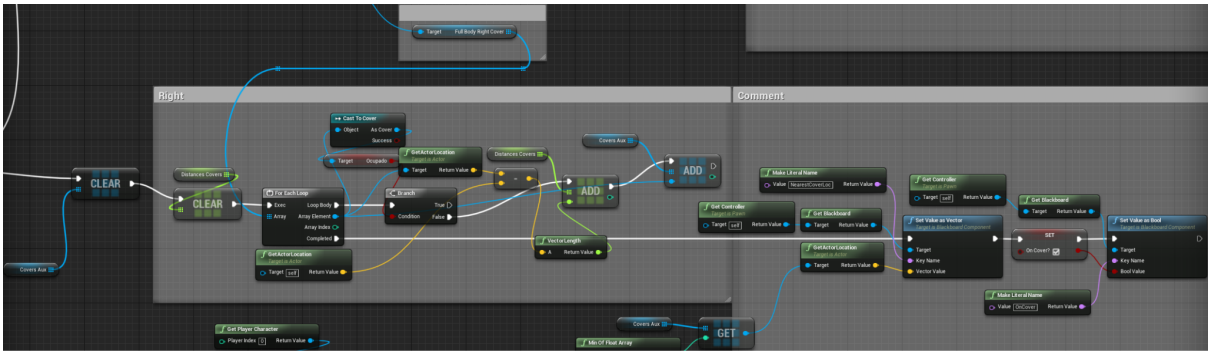


Figura 174: Captura de la funció SearchNearestCover 8

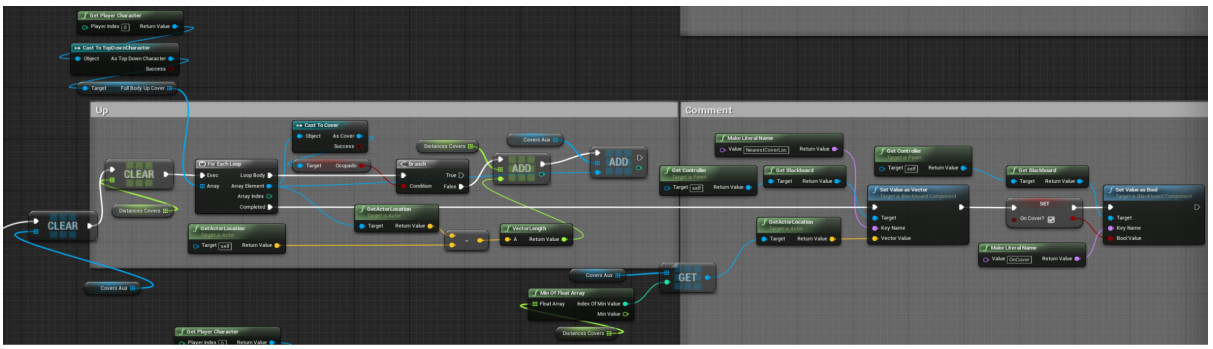


Figura 175: Captura de la funció SearchNearestCover 9

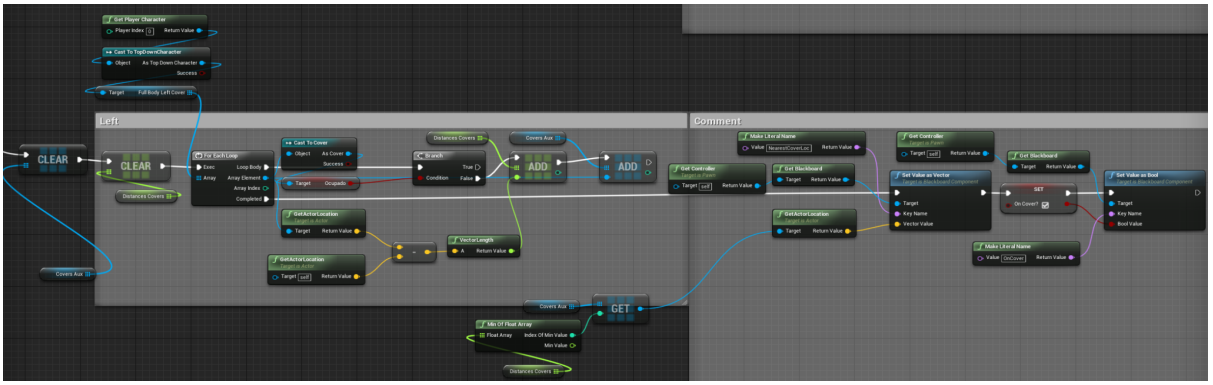


Figura 176: Captura de la funció SearchNearestCover 10

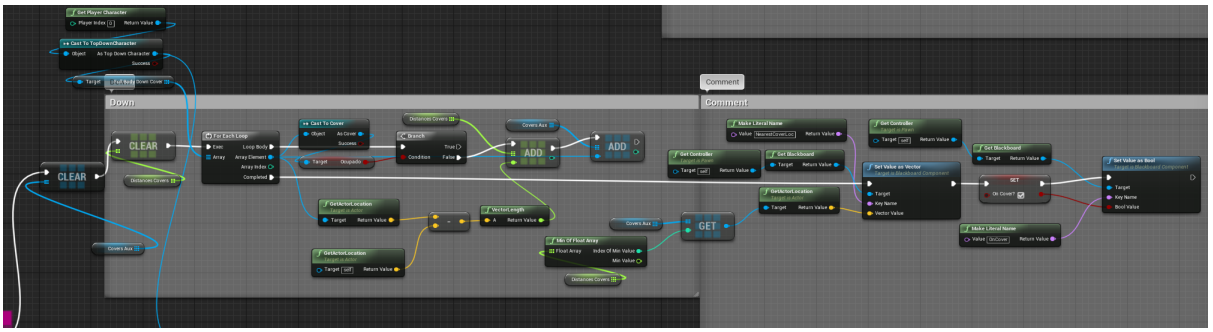


Figura 177: Captura de la funció SearchNearestCover 11

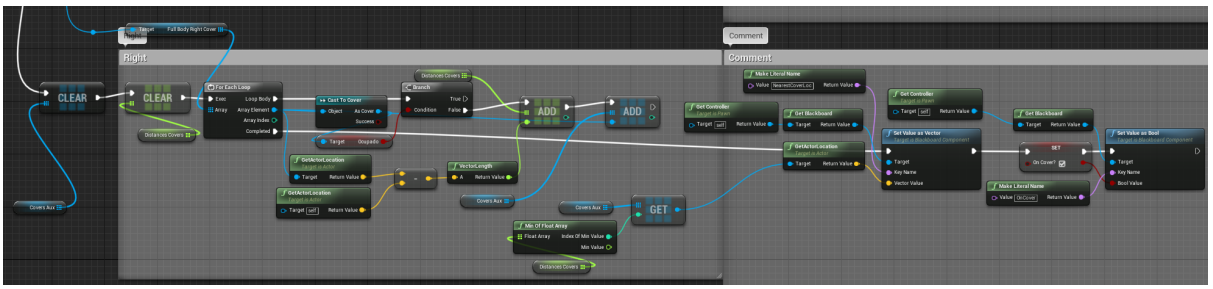


Figura 178: Captura de la funció SearchNearestCover 12

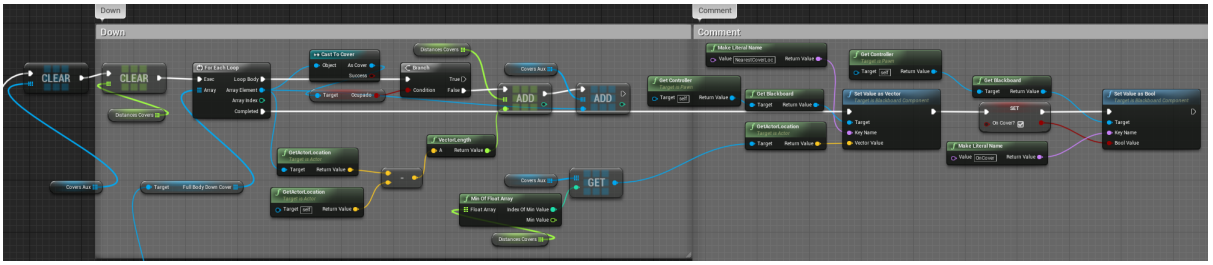


Figura 179: Captura de la funció SearchNearestCover 13

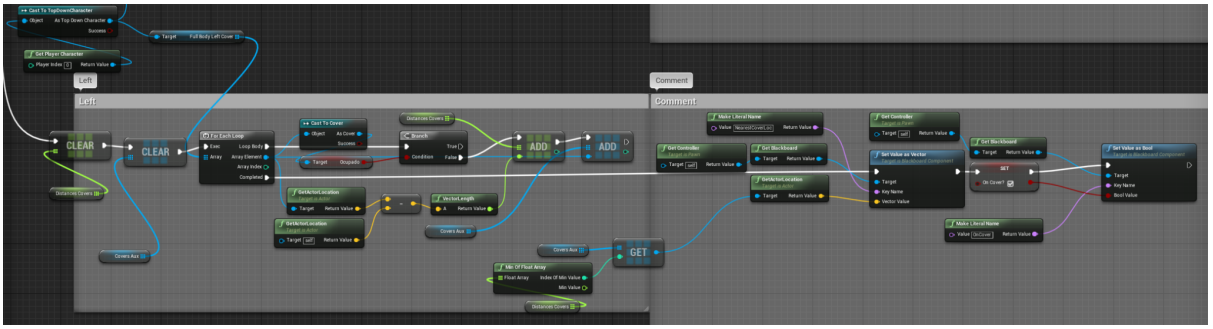


Figura 180: Captura de la funció SearchNearestCover 14

6.8.2. BlackBoard

La Blackboard de l'enemic, NPC_BB, té les següents variables (veure Figura 181):

- Location(Vector): vector de la posició a la qual es mourà l'enemic.

- PlayerSeen?(boolean): detecta si ha vist algun personatge o encara no.

- SelfActor(Actor): és una referència a l'enemic (la instància) que controla el Behaviour Tree.

- MyTurn(boolean): gestiona si és el torn de l'enemic o no.

- NearestCoverLoc(Vector): vector de la posició de la cobertura més propera perquè es mogui l'enemic.

- PlayerAlreadySeen(boolean): detecta si ha vist algun personatge, o encara no.

- OnCover(boolean): detecta si l'enemic es troba en una cobertura o no.

- PlayerNear(boolean): detecta si hi ha algun personatge a prop de l'enemic.

- BetterCoverLoc(Vector): vector de la posició d'una cobertura perquè es mogui l'enemic.

- CanShoot(boolean): detecta si l'enemic pot atacar o no.

- Flanked(boolean): detecta si està sent flanquejat.

- NonFlankLoc(Vector): detecta una posició a on no és flanquejat.

- CanMove(boolean): detecta si es pot moure.

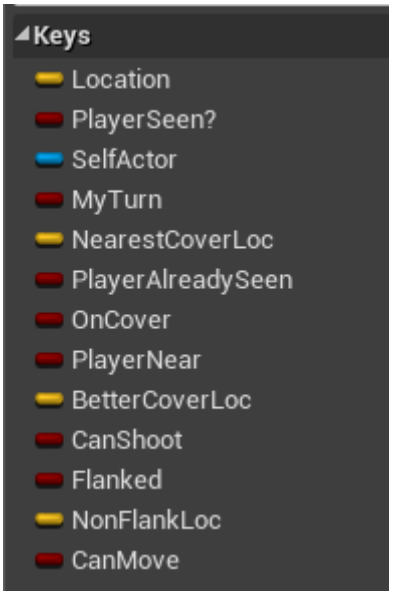


Figura 180: Captura de les variables del Blackboard de l'enemic

Respecte a aquestes variables, hi ha un parell que no han estat implementades completament, com podria ser la variable NonFlankLoc, però són mecàniques que seran agregades en futures versions del projecte.

6.8.3. AI Controller

Respecte a l'actor AI Controller NPC_Controller(veure Figura 181), no hi ha molt a afegir, ja que aquest només s'utilitza per cridar a la Blackboard pertinent.

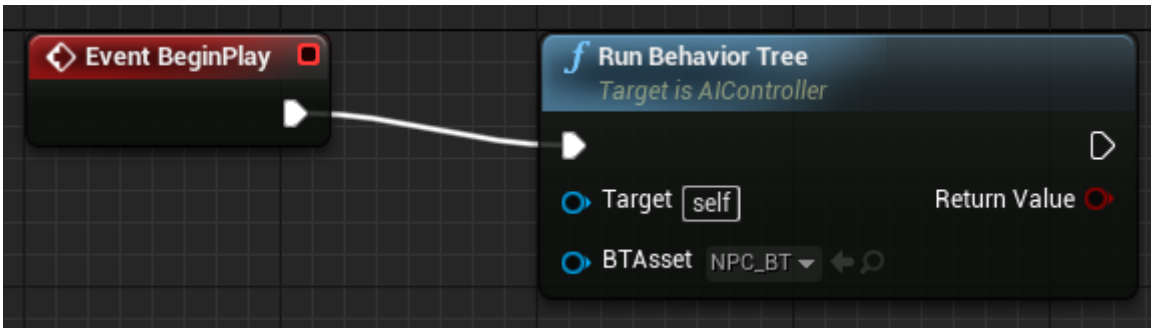


Figura 181: Captura de l'event BeginPlay de l'actor NPC_Controller

6.8.4. Behaviour Tree i Tasks

A l'hora d'implementar un arbre de comportament o Behaviour Tree a Unreal Engine, és molt recomanable entendre prèviament quins nodes incorporen aquests grafs i com funcionen en el flux d'execució:

- Node Selector: executa els nodes fill d'esquerra a dreta. L'execució s'atura quan un dels fills té èxit. Si tots els fills del Selector tenen èxit, el Selector té èxit, i si un fill del Selector falla, el Selector falla. Veure Figura 182.

Figura 182: Captura del node Selector del Behaviour Tree de l'enemic

- Node Sequence: executa els fills d'esquerra a dreta. L'execució s'atura quan un fill falla. Si un fill falla, la Sequence falla, i si tots els fills tenen èxit, la Sequence té èxit. Veure Figura 183.

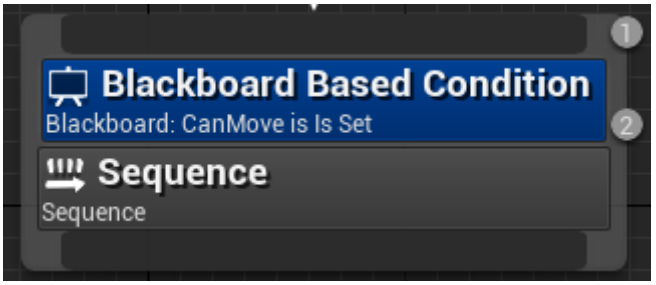


Figura 183: Captura del node Selector del Behaviour Tree de l'enemic

A més, qualsevol d'aquests nodes poden utilitzar condicions d'entrada, anomenades Blackboard decorator nodes.

El Behaviour Tree que implementem (veure Figura 184), anomenat NPC_BB, fa servir un Sequence com a node arrel. Amb un altre Sequence mirem si l'enemic pot moure's gràcies als decorator nodes mencionats (requadres blaus a la imatge). Fem servir un Selector, i cada branca filla d'aquest node és un Sequence amb altres condicions de Blackboard. Per una altra banda, els nodes fulla de les branques són crides a Tasks (tasques) que s'implementen a part del Behaviour Tree (requadres de color púrpura a la part inferior de la imatge).

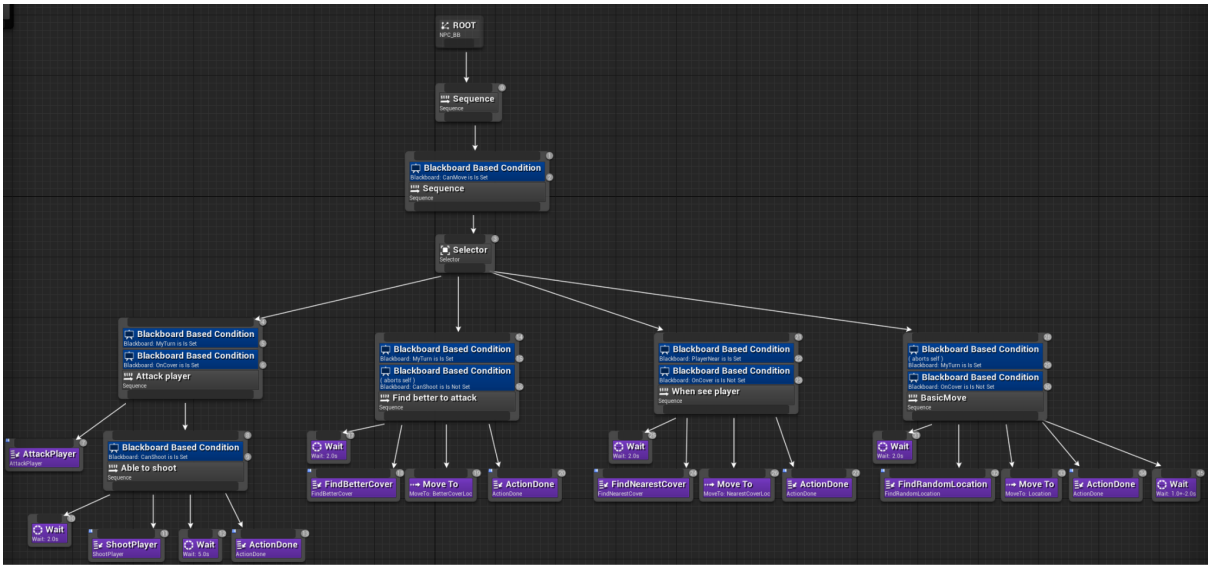


Figura 183: Captura general del Behaviour Tree de l'enemic

Anirem explicant cada Task a mesura que expliquem la branca a la qual apareix. A la Figura 184 podem trobar totes les tasques utilitzades.

Figura 184: Captura de les tasques de l'enemic

Donada la naturalesa dels nodes Sequence i Selector d'Unreal Engine, el mètode "natural" d'implementació dels arbres és posicionar les branques amb comportaments més "prioritaris" més aviat a l'esquerra, i deixar aquells comportaments que tenen menys transcendència cap a la dreta. Encara això, alguns comportaments de l'enemic no són conceptualment més prioritaris que altres, per la qual cosa utilitzem moltes condicions d'entrada (Blackboard decorator nodes) per assegurar que, encara que a priori les branques s'executin d'esquerra a dreta, el flux d'execució sigui el desitjat.

Amb aquesta decisió de disseny ja aclarida, dividim l'arbre en 7 branques principals (ordenades d'esquerra a dreta):

- Attack player(atacar al jugador)
- Find better to attack (trobar millor posició per atacar)
- When see player (quan veus al jugador)
- BasicMove (Moviment bàsic)

Amb el propòsit d'intentar explicar de la manera més senzilla possible el funcionament de l'arbre de comportament i de cadascuna de les branques, explicarem l'arbre de comportament seguint un dels possibles ordres d'execució que es podria donar perfectament al joc:

- L'enemic es mou pel mapa.
- L'enemic veu a un personatge.
- L'enemic busca una cobertura.
- L'enemic mira si la posició és bona per atacar (ho és)
- L'enemic ataca.
- El personatge que estava sent atacat es mou.
- L'enemic mira si la posició és bona per atacar (no ho és)
- L'enemic busca una millor cobertura.

Seguint aquest ordre d'esdeveniments especificat, el flux d'execució és el següent:

- Sequence Moviment bàsic

Per entrar en aquesta branca (la menys prioritària de tot l'arbre) i que l'enemic es mogui a una posició random del nivell, les condicions que

s'han de donar són que la variable MyTurn sigui true (que sigui el torn de l'enemic) i que la variable OnCover sigui false (que no es trobi en cap cobertura). Veure Figura 185.

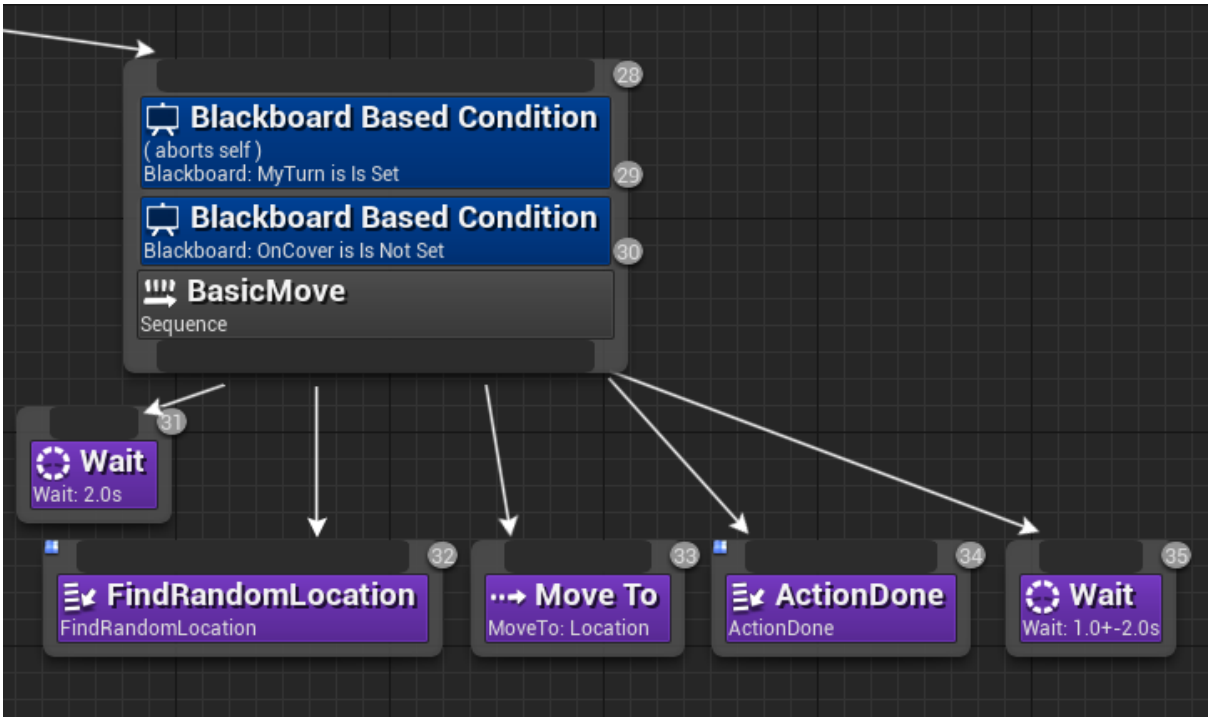


Figura 185: Captura del Sequence de BasicMove de l'enemic

En aquest node Sequence podem veure que una de les condicions del Blackboard, la condició referent a la variable MyTurn incorpora un comentari "(aborts self)" Aquesta és una configuració de Flow Control que serveix per permetre avortar l'execució de la branca en qualsevol moment gràcies a la configuració "(aborts self)". Veure Figura 186.

Figura 186: Captura de la configuració de Flow control de la condició de l'arbre referent a la variable MyTurn

En aquest Sequence trobem cinc tasques diferents. Aquestes cinc tasques són:

- Wait: aquest primer node serveix per evitar que l'execució del turn dels enemics sigui instantània.

- FindRandomLocation(veure Figura 187): aquesta tasca ha estat creada per mi i funciona de manera que, mitjançant la posició de l'enemic, es cridi el node "GetRandomReachablePointInRadius" per buscar una posició random dins d'un radi amb l'enemic com al centre. Un cop trobat el punt, es crida el node "LineTrace" agafant com a origen el punt trobat, i com a final fem una recte perpendicular al terra i que passi pel punt trobat. D'aquesta manera, el node "LineTrace" ens retorna un actor (que serà sempre un PlaceToStop), agafem la posició d'aquest actor i li enviem el vector resultant a la Blackboard.

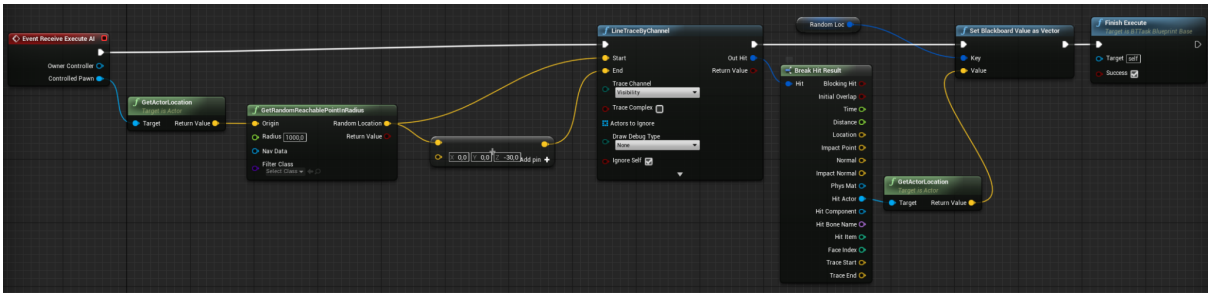


Figura 187: Captura de la funció de la tasca FindRandomLocation

- MoveTo: aquest node serveix per moure l'enemic a una posició guardada com a vector. Per tant, utilitzant el vector obtingut en la tasca anterior, cridem aquest node per moure l'enemic.

- ActionDone(veure Figura 188): En aquest cas, l'execució és més senzilla que en l'anterior. En aquest cas, només agafem l'actor d'aquest enemic per cridar la funció que gestiona les accions.

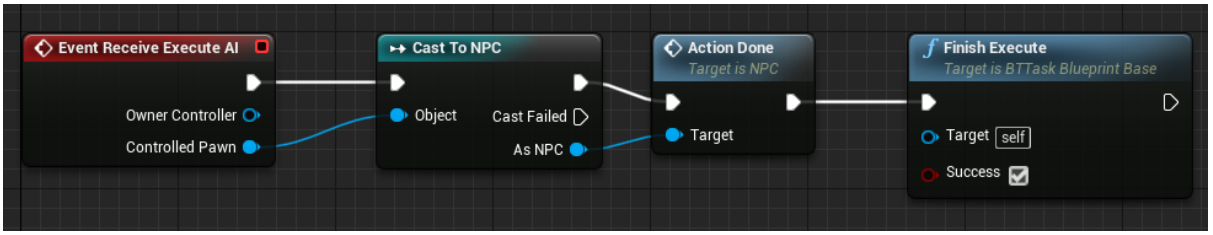


Figura 188: Captura de la funció de la tasca ActionDone

- Wait: similar a l'anterior tasca wait, aquest node serveix per evitar que l'execució del torn dels enemics sigui instantània, però contràriament a l'anterior, en aquest hem posat una variable que aconsegueix aplicar-li una derivació del temps, fent que cada execució trigui un temps diferent.

Figura 188: Captura de la configuració de la tasca wait

- Sequence when see player

Per entrar en aquesta branca i que l'enemic es mogui a la cobertura més propera del nivell, les condicions que s'han de donar són que la variable PlayerNear sigui true (que l'enemic hagi vist al personatge) i que la variable OnCover sigui false (que no es trobi en cap cobertura). Veure Figura 189.

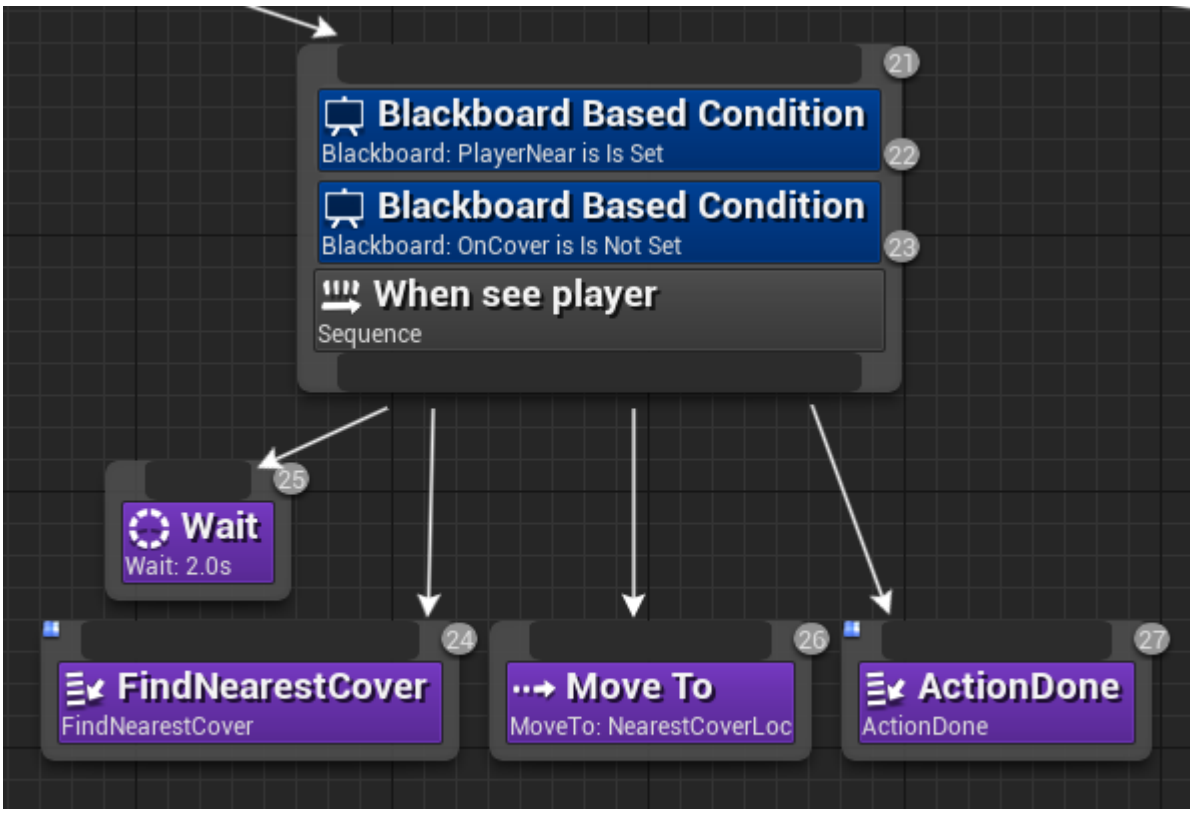


Figura 189: Captura del Sequence de WhenSeePlayer de l'enemic

En aquest Sequence trobem quatre tasques diferents. Aquestes quatre tasques són:

- Wait: aquest primer node serveix per evitar que l'execució del turn dels enemics sigui instantània.

- FindNearestCover(veure Figura 190): aquesta tasca crida a la funció per trobar la cobertura més propera de l'actor de l'enemic.

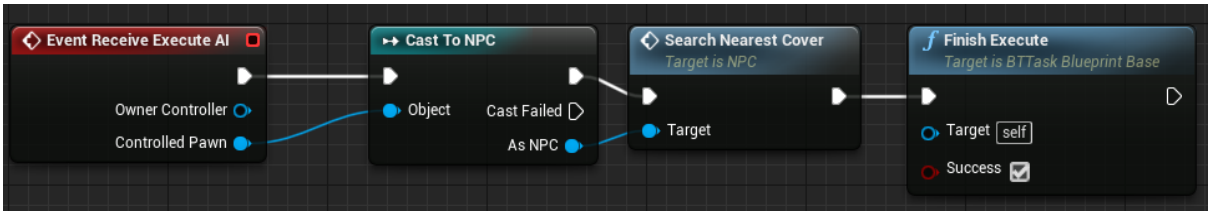


Figura 190: Captura de la funció de la tasca FindNearestCover

- MoveTo: aquest node serveix per moure l'enemic a una posició guardada com a vector, per tant, utilitzant el vector obtingut en la tasca anterior. Cridem aquest node per moure l'enemic. En aquest cas fa servir el vector NearestCoverLoc.

- ActionDone: aquesta és la mateixa tasca que la de l'apartat anterior. Agafem l'actor d'aquest enemic per cridar la funció que gestiona les accions.

- Find better to attack

Per entrar en aquesta branca i que l'enemic es mogui a la cobertura més propera del nivell, les condicions que s'han de donar són que la variable MyTurn sigui true (que sigui el torn de l'enemic) i que la variable CanShoot sigui false (que no pugui atacar). Veure Figura 191.

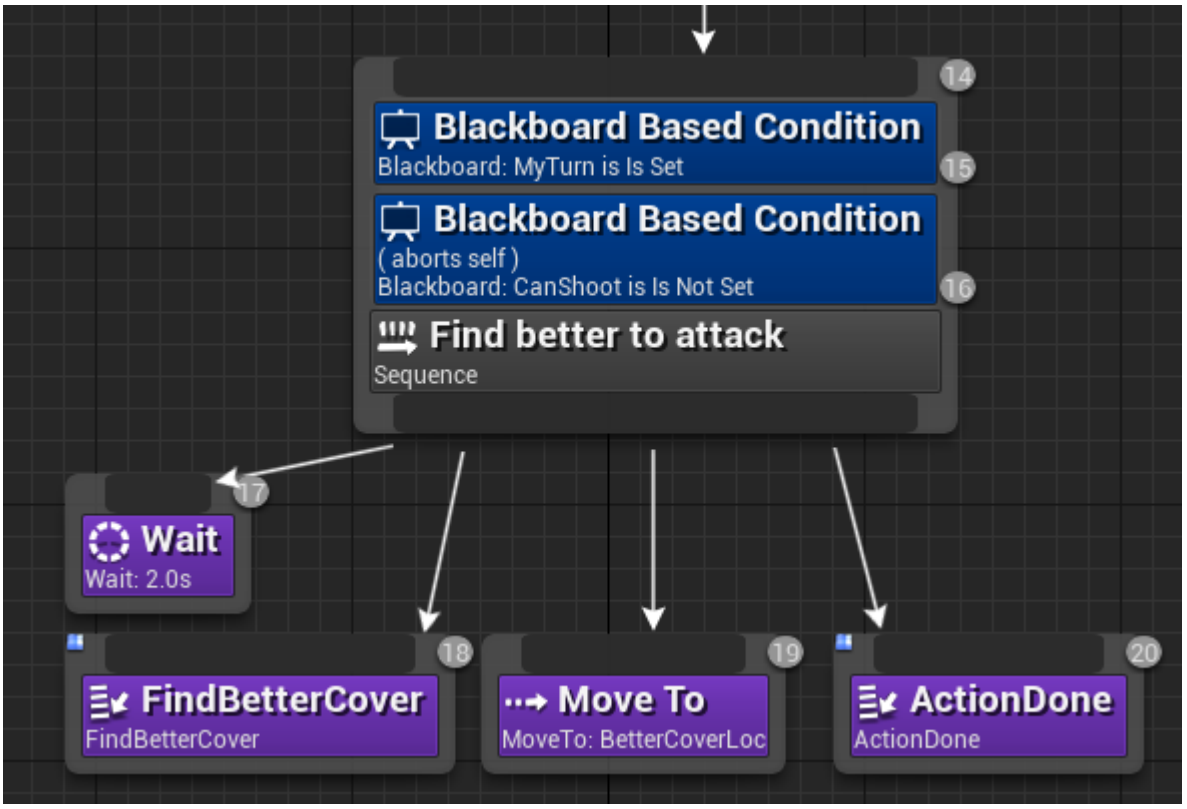


Figura 191: Captura del Sequence de Find Better To Attack de l'enemic

En aquest Sequence trobem quatre tasques diferents. Aquestes quatre tasques són:

- Wait: aquest primer node serveix per evitar que l'execució del torn dels enemics sigui instantània.

- FindBetterCover(veure Figura 192): aquesta tasca crida a la funció per trobar una millor cobertura de l'actor de l'enemic.

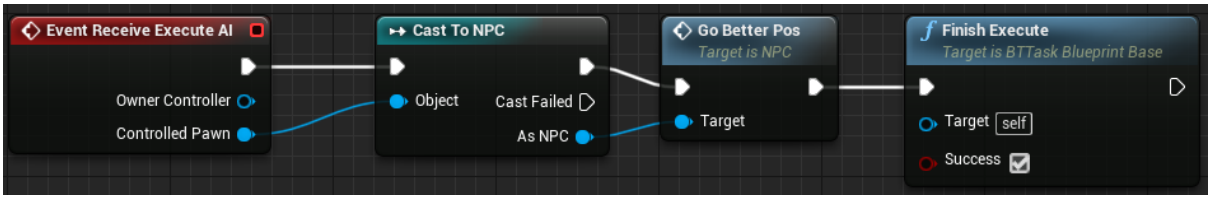


Figura 192: Captura de la funció de la tasca FindBetterCover

- MoveTo: aquest node serveix per moure l'enemic a una posició guardada com a vector. Per tant, utilitzant el vector obtingut en la tasca anterior cridem aquest node per moure l'enemic. En aquest cas fa servir el vector BetterCoverLoc.

- ActionDone: aquesta és la mateixa tasca que la de l'apartat anterior, agafem l'actor d'aquest enemic per cridar la funció que gestiona les accions.

- AttackPlayer

Per entrar en aquesta branca i que l'enemic ataqui, les condicions que s'han de donar són que la variable MyTurn sigui true (que sigui el torn de l'enemic) i que la variable OnCover sigui true (que es trobi en una cobertura). Un cop accedit, dins d'aquest es troba un altre Sequence anomenat Able to attack, només es pot accedir si la variable canShoot es true (si l'enemic pot atacar). Veure Figura 193.

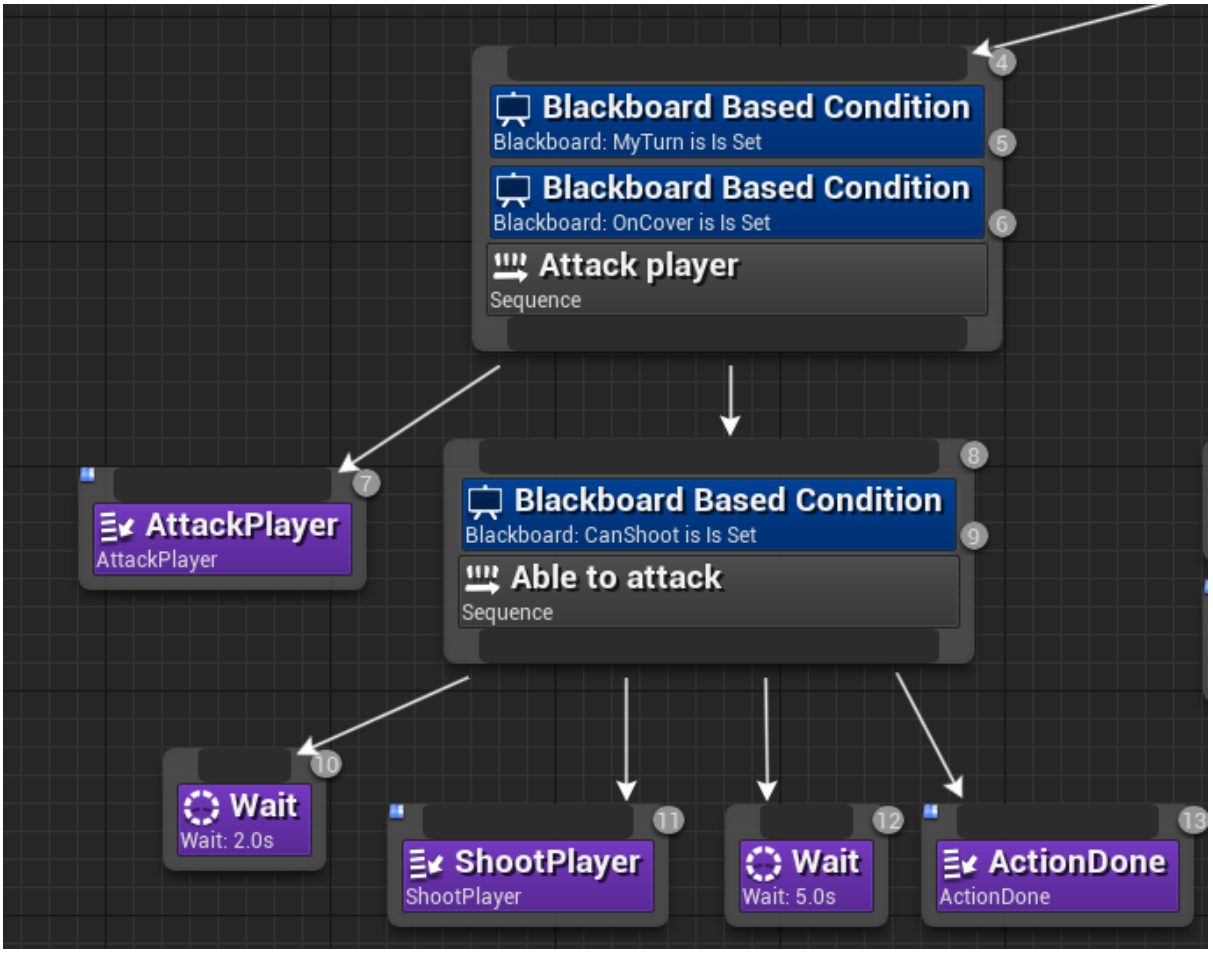


Figura 193: Captura del Sequence de AttackPlayer de l'enemic

En aquest Sequence trobem cinc tasques diferents. Aquestes cinc tasques són:

- AttackPlayer(veure Figura 194): aquesta tasca crida a la funció encarregada de trobar si pot atacar l'actor de l'enemic.

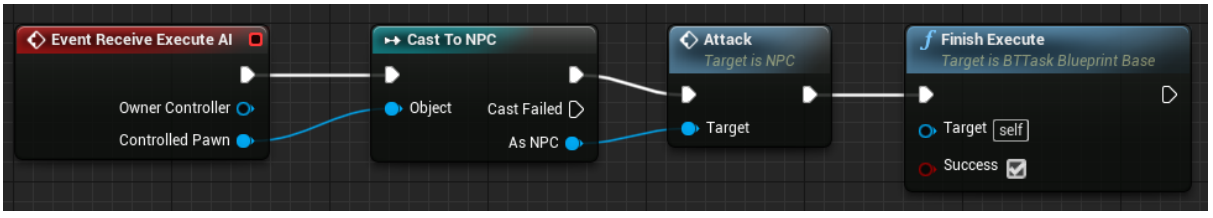


Figura 194: Captura de la funció de la tasca AttackPlayer

- Wait: aquest node serveix per evitar que l'execució del torn dels enemics sigui instantània.

- ShootPlayer(veure Figura 195): aquesta tasca crida a la funció encarregada d'atacar al personatge de l'actor de l'enemic.

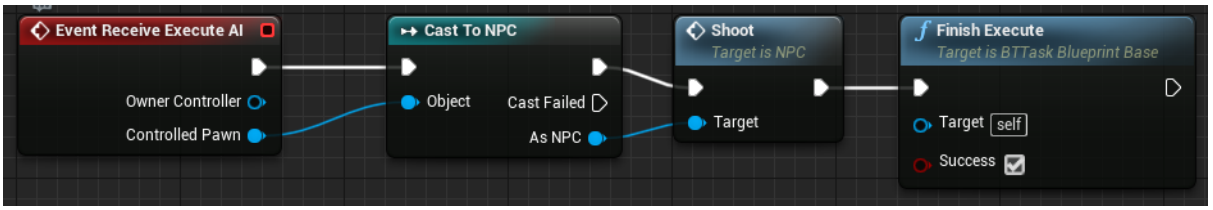


Figura 195: Captura de la funció de la tasca ShootPlayer

- Wait: aquest primer serveix per evitar que l'execució del torn dels enemics sigui instantània.

- ActionDone: aquesta és la mateixa tasca que la de l'apartat anterior, agafem l'actor d'aquest enemic per cridar la funció que gestiona les accions.

6.9. Implementació widgets

Els widgets han estat elements claus pel desenvolupament del joc, sobretot a les escenes dels menús. Seguidament, farem un repàs d'aquests menús, juntament amb els widgets respectius.

6.9.1. Menú SeleccionarMapa

Aquest menú ha estat el més senzill de muntar, ja que disposa del widget SeleccionarMapa (veure Figures 196 i 197). A partir de l'actor del gestor, modifiquem els botons per fer-los clicables o no.

Figura 196: Captura del widget SeleccionarMapa

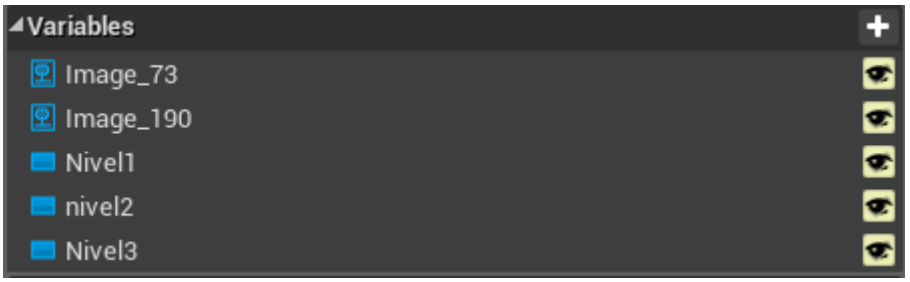


Figura 197: Captura de les variables del widget SeleccionarMapa

El graph d'aquest widget (veure Figura 198) és el més senzill de tots, ja que només ha d'obrir el nivell pertintent.

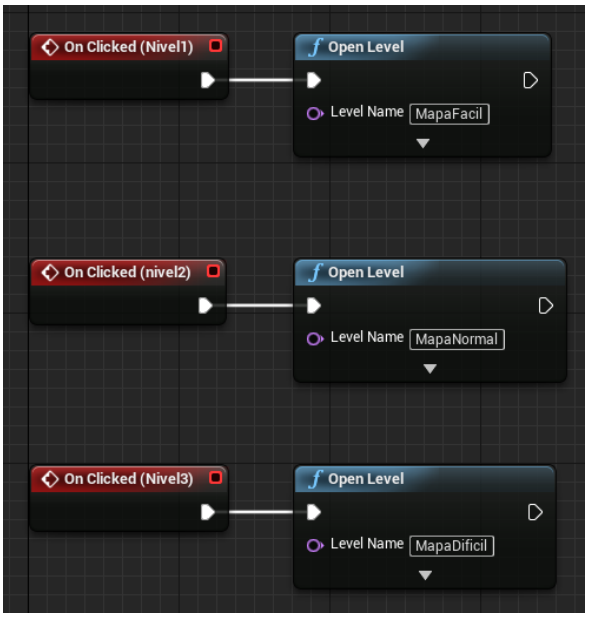


Figura 198: Captura del Graph del widget SeleccionarMapa

Aquest widget es crea a l'actor del gestor dels personatges. Mitjançant un bucle, que recorre un boolean array, es mira per cada element si el valor és true o false i s'aplica aquesta variable al botó per definir si és clicable o no. Veure Figura 199.

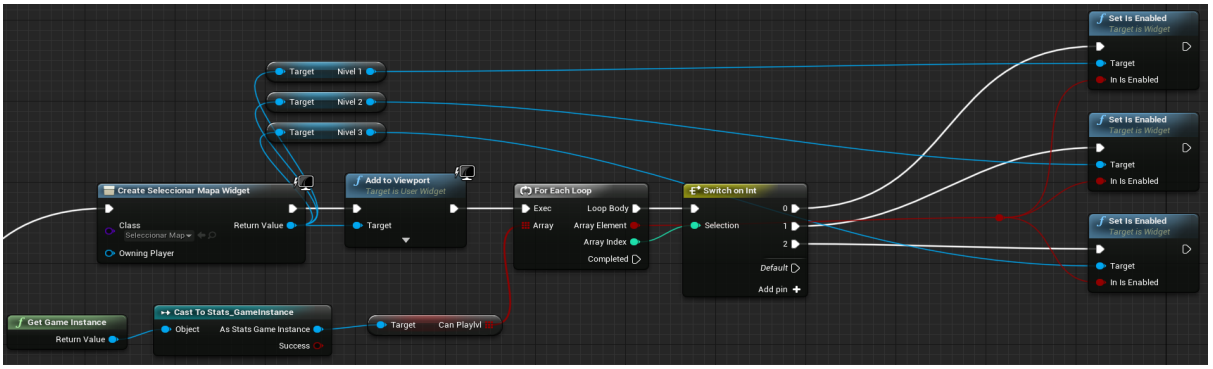


Figura 199: Captura de la part de la funció BeginPlay del gestor dels personatges referent al menú SeleccionarMapa

6.9.2. Menú PlayerSelected

Aquest menú s'encarrega de llistar els personatges que es portarà durant la partida. Disposa de diferents botons per modificar els Spells del personatge, eliminar-lo o afegir un personatge més si hi ha espai. Veure Figures 200 i 201.

Figura 200: Captura del widget PlayerSelected

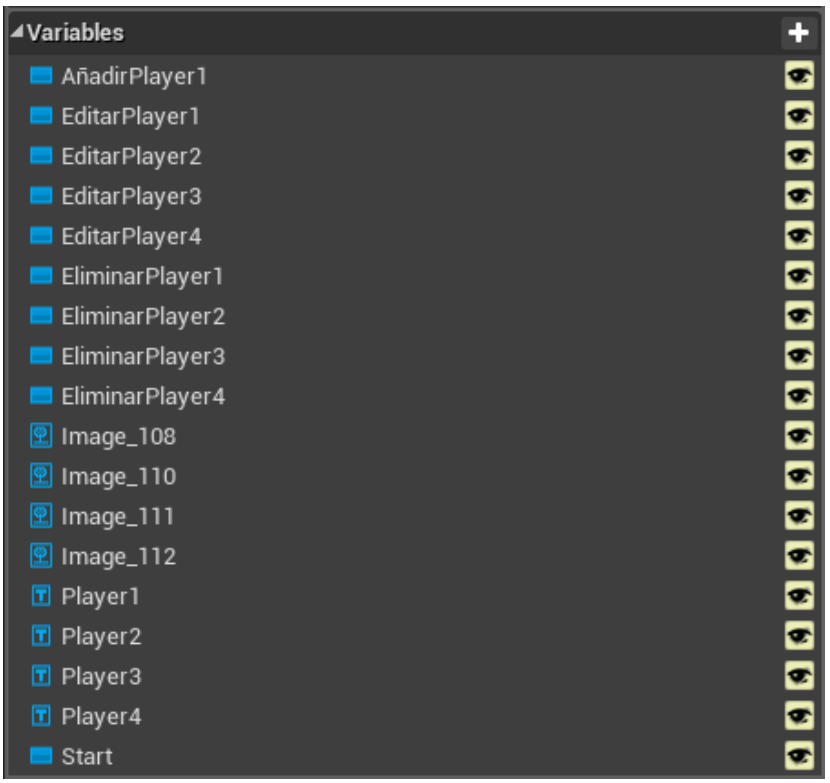


Figura 201: Captura de les variables del widget PlayerSelected

En aquest cas, el Graph d'aquest widget és bastant més complex, ja que ha de permetre diferents opcions pels botons. Aquests botons són:

6.9.2.1. Botó Start

Aquest botó obre el nivell per seleccionar el mapa que jugar. Veure Figura 202.

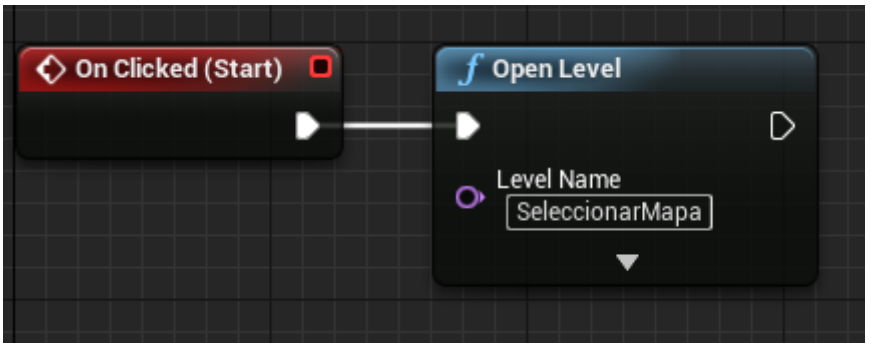


Figura 202: Captura del Graph del widget PlayerSelected referent al botó Start

6.9.2.2. Botó afegir personatge

Aquest botó permet afegir un altre personatge a l'equip. El codi és molt senzill, ja que només ha d'obrir el nivell "ArmyEditor". Veure Figura 203.

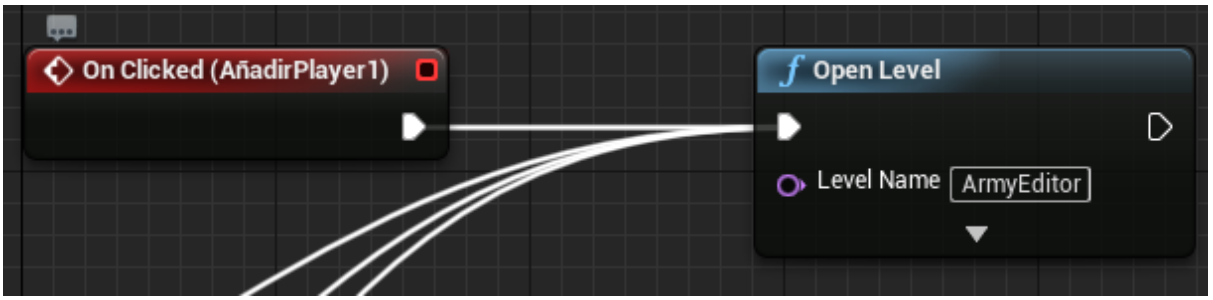


Figura 203: Captura del Graph del widget PlayerSelected referent al botó d'afegir un personatge

6.9.2.3. Botó editar personatge

Aquest botó permet editar els Spells del personatge. Aquesta funció (veure Figura 204) fa el següent:

- Cada un dels quatre botons referents als personatges criden la mateixa funció, però modificant la variable id referent a l'id del personatge.

- Aquesta funció comprova que l'id sigui vàlid.

- En cas de ser-ho, utilitzant la variable de la instància del joc referent als personatges seleccionats (NumPlayers), agafem l'element guardat a l'índex referent a l'id vàlida.

- Un cop tenim el valor, el guardem a la variable EditingPlayerId i obrim el nivell "EditPlayer".

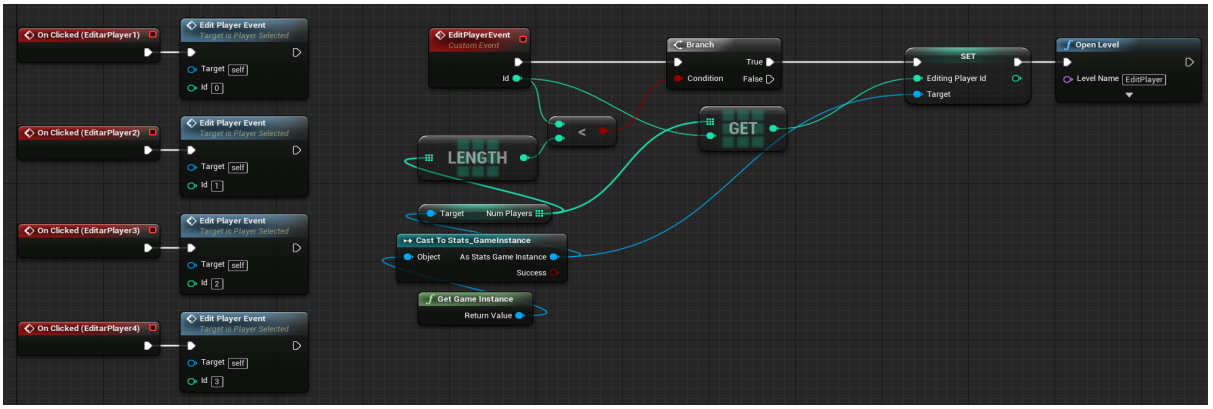


Figura 204: Captura del Graph del widget PlayerSelected referent als botons d'editar el personatge

6.9.2.4. Botó eliminar personatge

Per acabar, tenim els botons referents a eliminar un personatge, aquests botons (veure Figura 205) fan el següent:

- Criden a la funció de QuitPlayer del gestor a on li passen l'element de la variable NumPlayers referent a l'id de l'equip.

- Es modifica el text a un string buit.

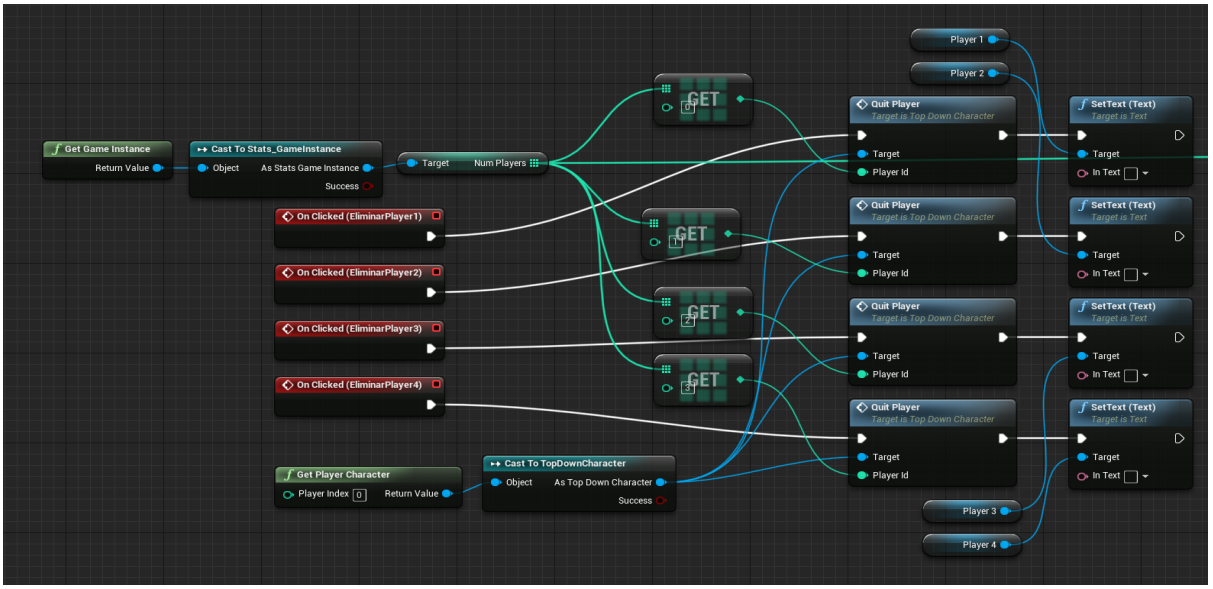


Figura 205: Captura del Graph del widget PlayerSelected referent als botons d'eliminar un personatge

6.9.2.5. Creació widget

Respecte a la creació del widget, aquest es crea dins de l'event BeginPlay (veure figura 206) de l'actor level Blueprint del nivell "PlayerSelected". Aquesta funció fa el següent:

- Crea el widget.

- Utilitzant un bucle, que recorre la variable NumPlayers.

- Per cada un d'aquests, sobreescriu el text referent al nom del personatge.

- Un cop acabat amb el bucle, mostra el widget final per pantalla.

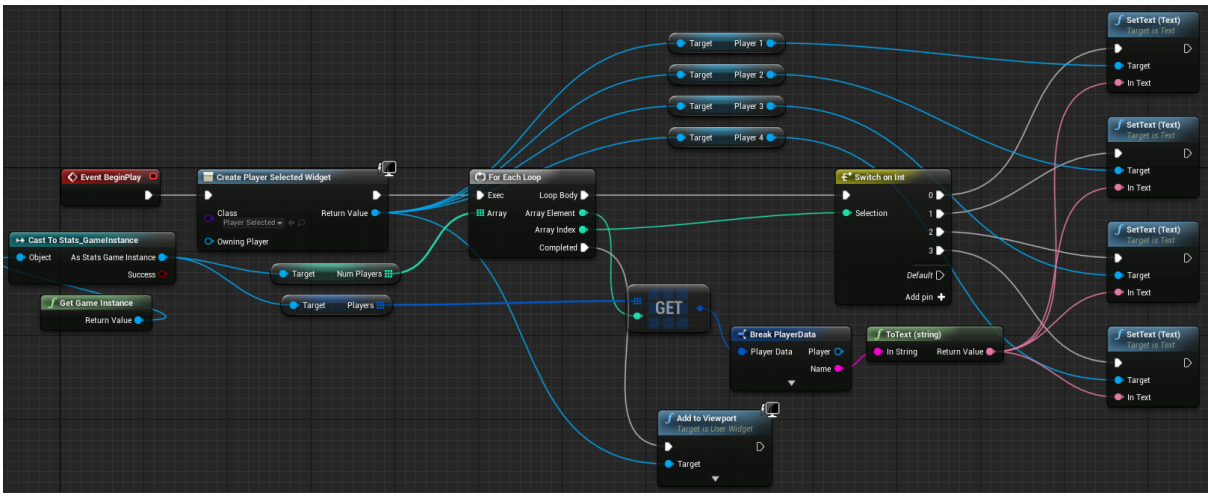


Figura 206: Captura de la funció BeginPlay del level Blueprint de l'escena PlayerSelected

6.9.3. Menú EditPlayer

Aquest menú (veure Figura 207) és el menú que permet al jugador modificar els Spells dels personatges. Aquest menú consta de diferents widgets.

Figura 207: Captura del resultat mostrat per pantalla l'entrar al Menú EditPlayer

6.9.3.1. Widget PlayerEditor

Aquest widget (veure Figures 208 i 209) és l'encarregat de crear la base del menú.

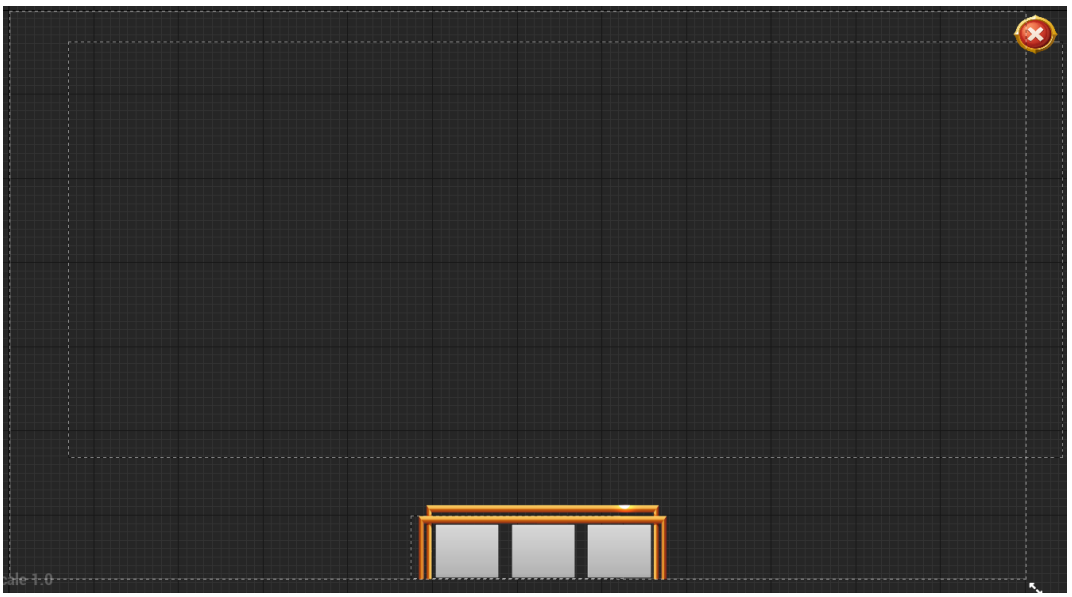


Figura 208: Captura del widget PlayerEditor

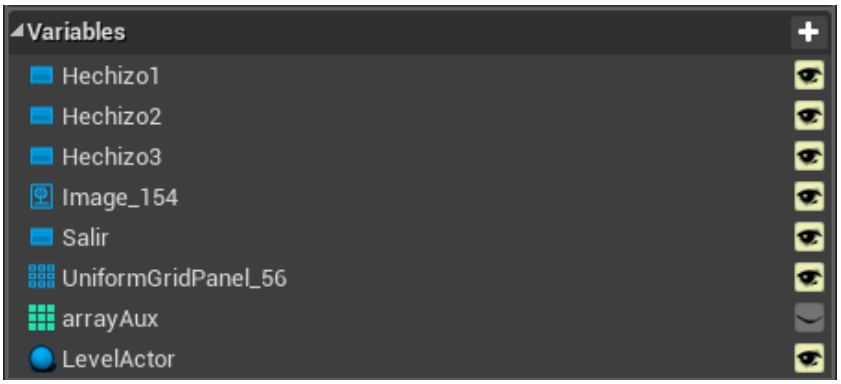


Figura 209: Captura de les variables del widget PlayerEditor

Aquest widget disposa de diferents apartats a comentar. Aquests són:

- Elements importants

  Aquest widget conté un element que serà necessari per mostrar per pantalla els Spells, aquest es tracta del component “UniformGridPanel”, i es comentarà a l’apartat referent a la creació d’aquest widget 6.9.3.4.

- Botó Sortir

  El botó de sortir, en aquest nivell, és una creu que es troba a la cantonada superior dreta de la pantalla (es pot veure a la Figura 208 vista prèviament). Aquest botó serveix per obrir un altre nivell. Veure Figura 210.

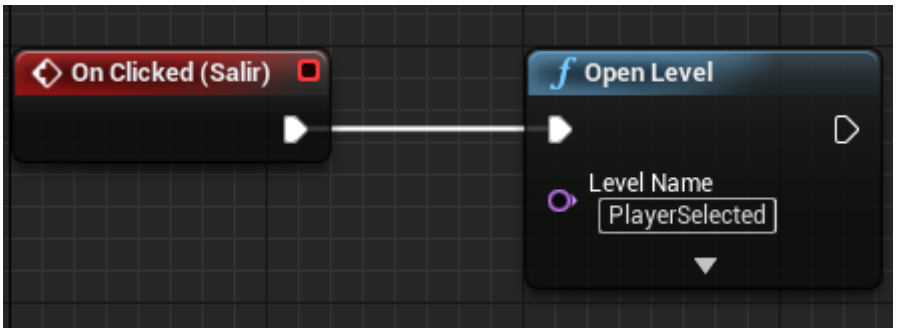


Figura 210: Captura del Graph del widget PlayerEditor referent al botó Sortir

- Botó eliminar Spells

  Aquests botons (veure Figures 211 i 212) fan referència als tres Spells que pot portar cada personatge de l'equip. Aquests botons fan el següent:

  - Criden a la funció DeleteSpell però passant-li una id diferent en funció de si és el primer, segon o tercer Spell.

  - La funció DeleteSpell, agafa de la instància de joc les variables EditingPlayerId mencionada anteriorment i la variable Players.

  - Utilitzant la variable EditingPlayerId com a índex. Agafem el Player d'aquella posició i ens guardem la seva variable Spells a un array d'integrals auxiliar.

  - En aquest nou array, modifiquem el valor i li establim un -1.

- Un cop tenim l'array modificat, el sobreescrivim al mateix personatge.

- Per acabar, cridem una interfície anomenada EditPlayerInterface que utilitzarem al level Blueprint d'aquest nivell.

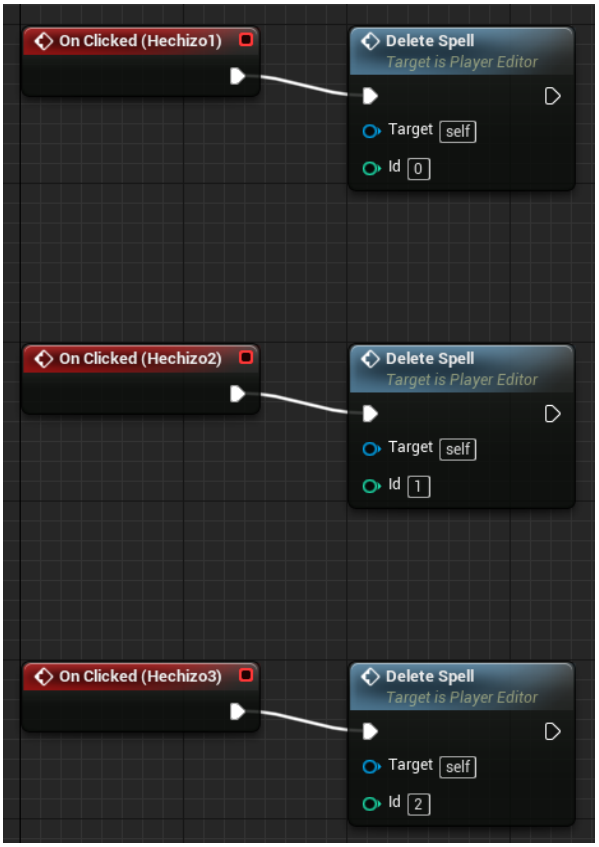


Figura 211: Captura del Graph del widget PlayerEditor referent als botons d'eliminar Spells

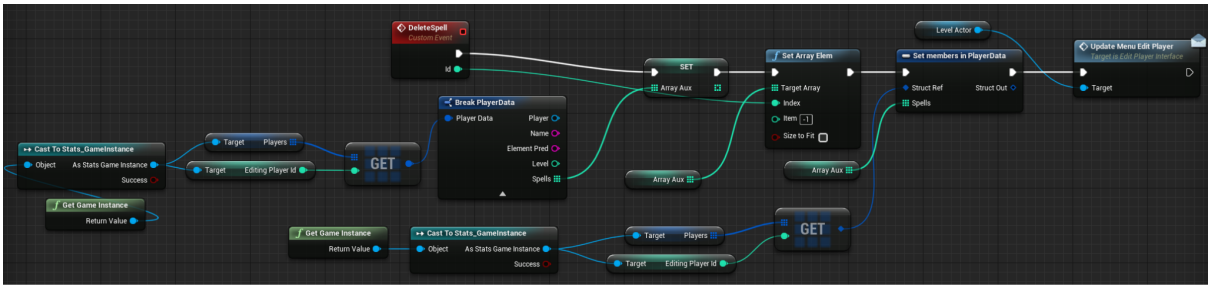


Figura 212: Captura del Graph del widget PlayerEditor referent a la funció DeleteSpell

6.9.3.2. Widget PlayerEditorElements

Aquest widget (veure Figures 213 i 214) és l'encarregat de crear les columnes del menú final.

Figura 213: Captura del widget PlayerEditor_Elements

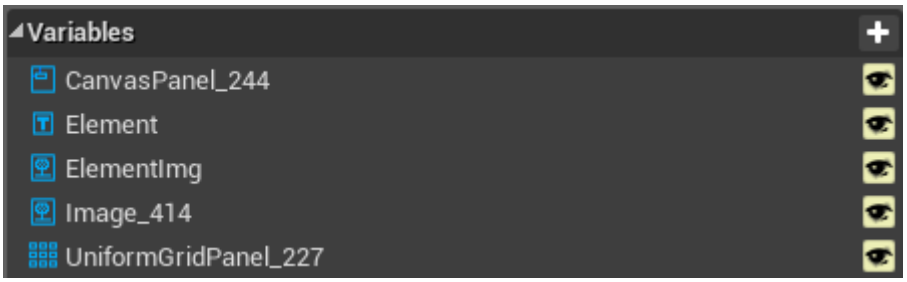


Figura 214: Captura de les variables del widget PlayerEditor_Elements

Aquest widget disposa d'un apartat a comentar, que és:

- Elements importants

  Similar al widget anterior, aquest també es tracta del component "UniformGridPanel", i es comentarà a l'apartat referent a la creació d'aquest widget, 6.9.3.4.

6.9.3.3. Widget PlayerEditorSpells

Aquest widget (veure Figures 215 i 216) és l'encarregat de crear els quadres a on es mostrin els Spells amb les seves dades.

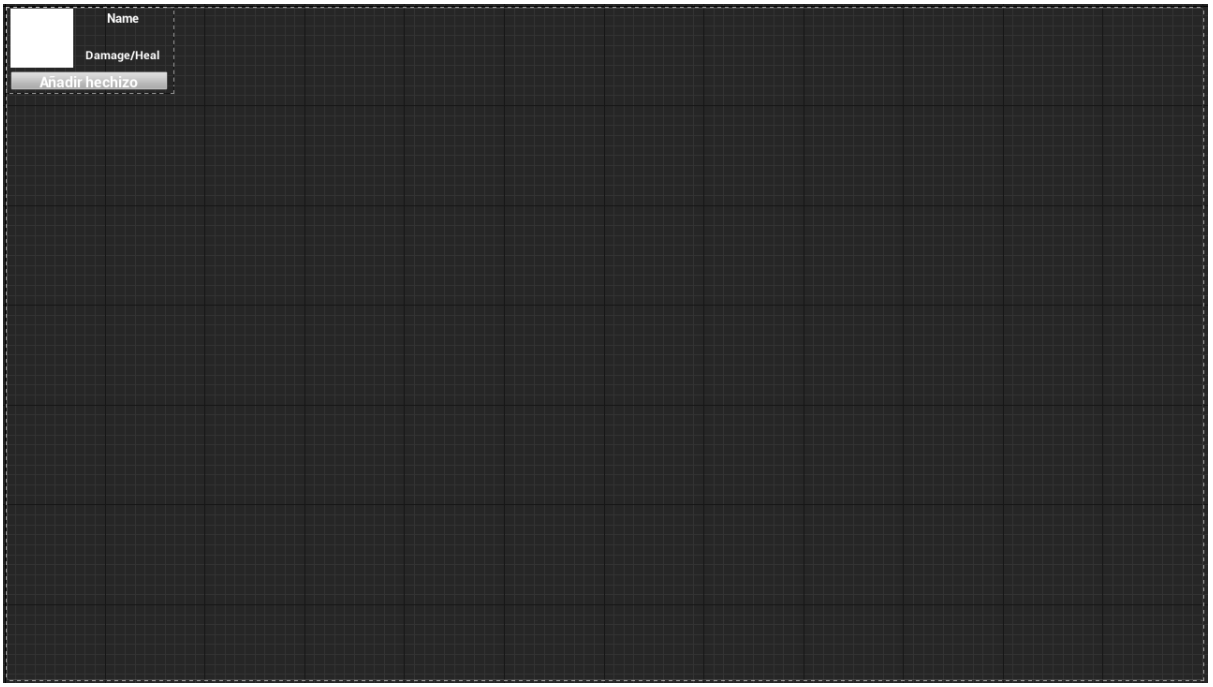


Figura 215: Captura del widget PlayerEditor_Spell

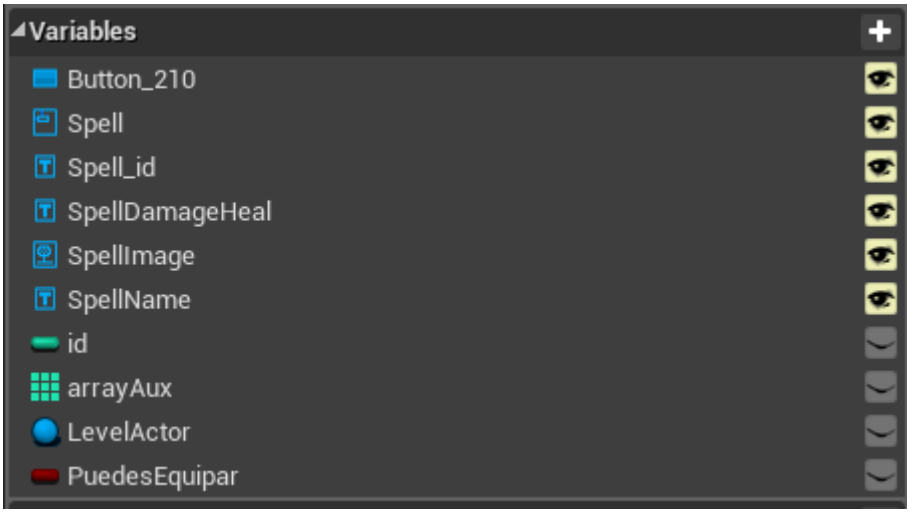


Figura 216: Captura de les variables del widget PlayerEditor_Spell

Aquest widget disposa de diferents apartats a comentar, aquests són:

- Botó afegir Spell

  Aquest apartat (veure Figura 217) s'encarrega de permetre que el jugador pugui afegir Spells i els gestiona. Aquesta funció fa:

  - Defineixes la variable PuedesEquipar a false i es guarda la llista de Spells del personatge que s'està editant com a una variable auxiliar.

  - Mitjançant un bucle amb un break que recorre la llista de Spells del personatge, es mira si tots els valors són iguals o superiors a 0.

  - En el cas de trobar un número inferior a 0, es guarda l'id dins de l'array en una variable que es diu id i guarda la variable de PuedesEquipar a true.

  - En sortir del bucle, mirem el valor de la variable PuedesEquipar, en el cas de ser false no es fa res.

- En el cas que el valor sigui true modifiquem l'array auxiliar creat a l'inici. Per fer-ho, agafem l'id guardat dins del bucle i com a ítem, guardem el text del component Spellid.

- Sobreescrivim l'array de Spells del personatge amb l'array auxiliar i la variable de sortida del personatge, la sobreescrivim de l'array de players.

- Un cop acabat i utilitzant una instància de joc, cridem a la funció del level Blueprint encarregada de refrescar la pantalla.

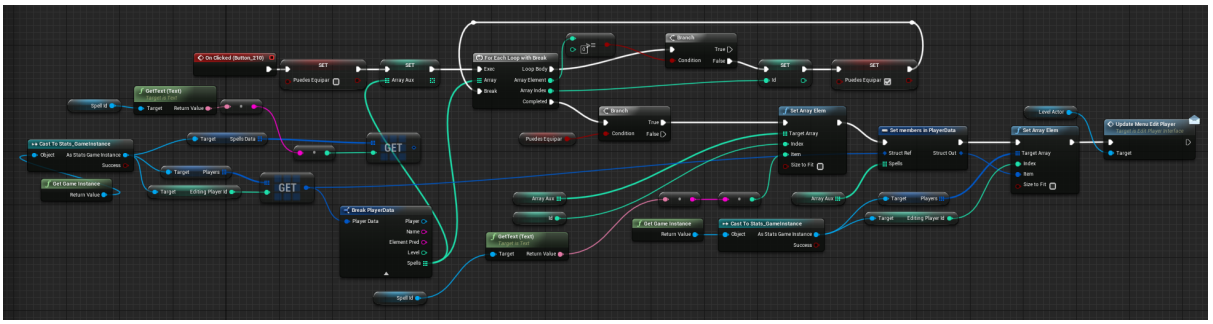


Figura 217: Captura del Graph del widget PlayerEditor_Spellreferent a la funció per afegir un Spell

6.9.3.4. Creació widget

Per la creació d'aquest menú final, ha estat necessari utilitzar el level Blueprint. Aquest actor consta de dues funcions importants:

- Event BeginPlay

  Com el nom indica, aquesta funció (veure Figura 218) correspon a la inicialització de l'actor. Aquesta funció fa:

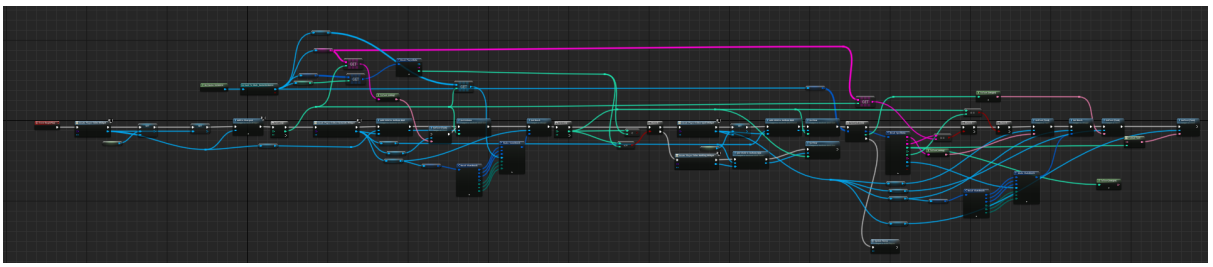


Figura 218: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció BeginPlay

- Crea el widget PlayerEditor, es guarda a la variable "PlayerEditor" i es mostra al viewport.

- Es crea un bucle que recorre el nombre total d'elements. Veure Figura 219.

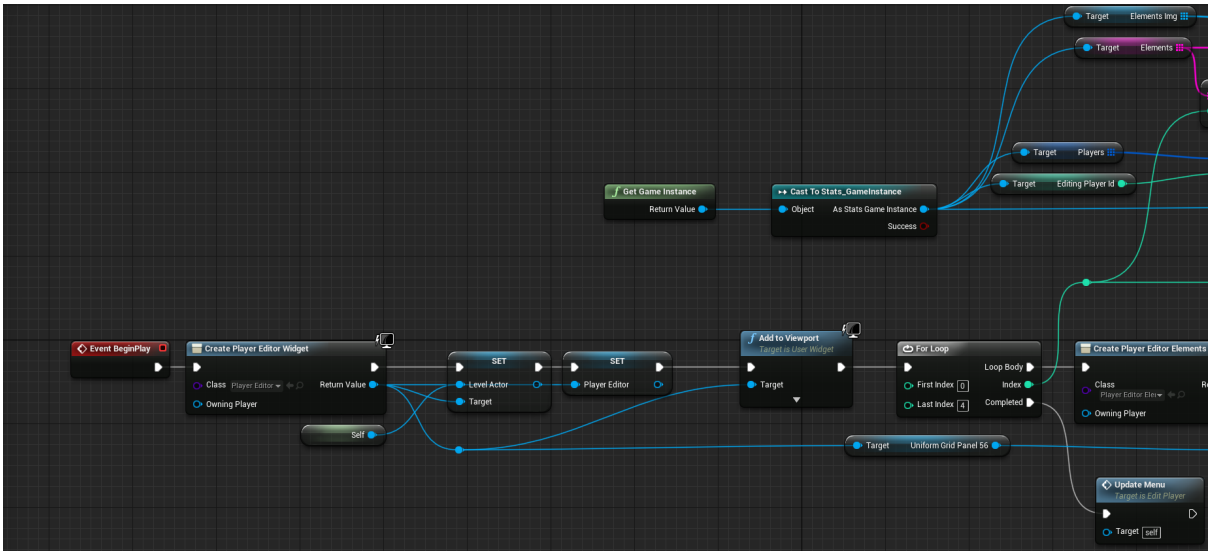


Figura 219: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció BeginPlay 1

- Per cada iteració, es crea un widget PlayerEditor_Element i s'afegeix a l'índex que toca com a fill de la variable UniformGrid del widget PlayerEditor.

- Modifica la imatge utilitzant el node “Set Brush” a partir de la variable de la instància de joc ElementsImg, agafant la imatge de la mateixa posició que l'índex. Veure Figura 220.

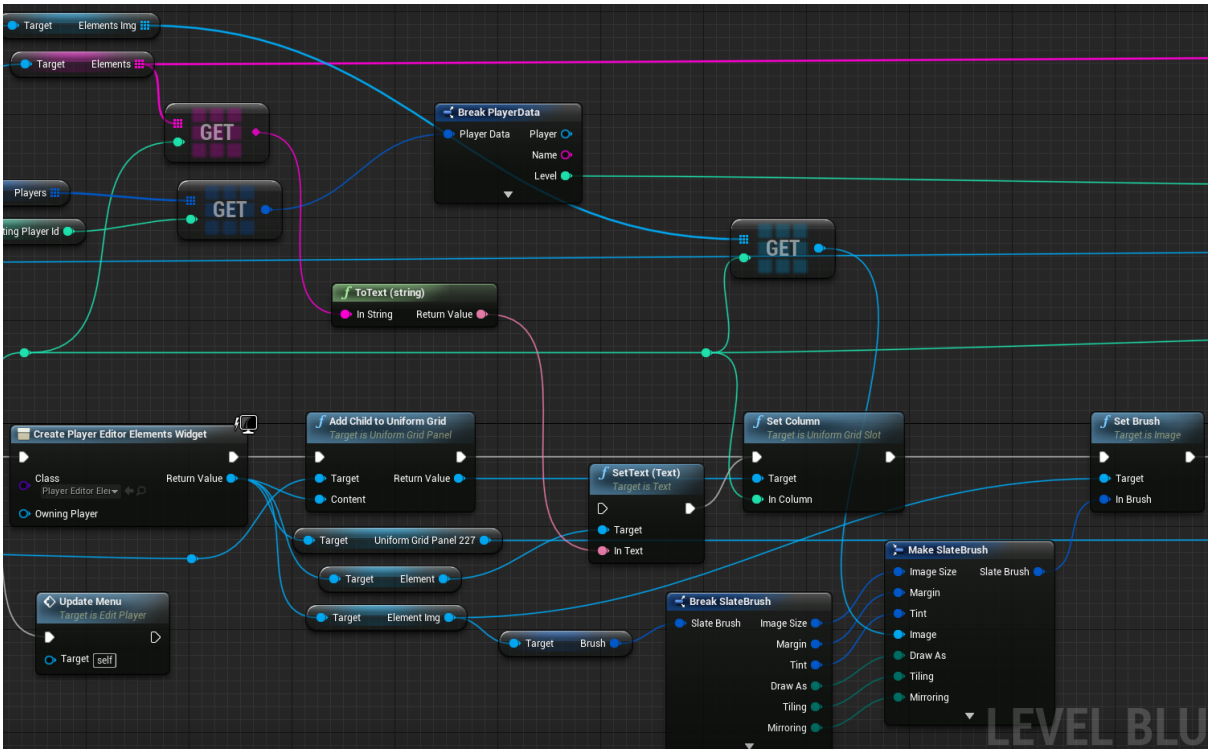


Figura 220: Captura de l’actor del level Blueprint de l’escena EditPlayer, de la funció BeginPlay 2

- Per cada widget PlayerEditor_Element i utilitzant un altre bucle que recorri del 0 fins al nivell màxim (en aquest cas 4) mira si el nivell del personatge és més gran o igual que l'índex de l’array.

- En el cas que el nivell del personatge sigui més petit, es crea un widget buit.

- En el cas que el nivell del personatge sigui més gran o igual, es crea un widget de PlayerEditor_Spell i l’afegeixes a l'índex que toca com a fill de la variable UniformGrid del widget PlayerEditor_Element. Veure Figura 221.

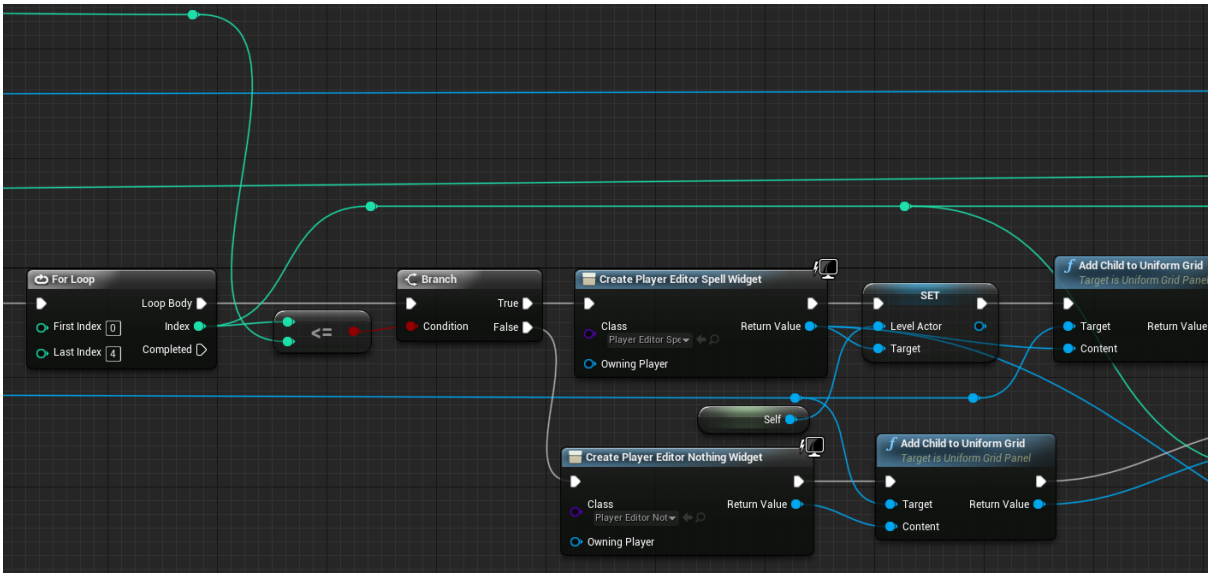


Figura 221: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció BeginPlay 3

- Per agafar el Spell que toca mostrar, es fa un altre bucle que recorri l'array de SpellData i per cada Spell es mira si l'element és el de la columna i si el nivell és igual a l'índex de l'anterior bucle.

- Un cop acabada l'execució, es crida la funció Event Update Menu Edit Player.Veure Figures 222 i 223.

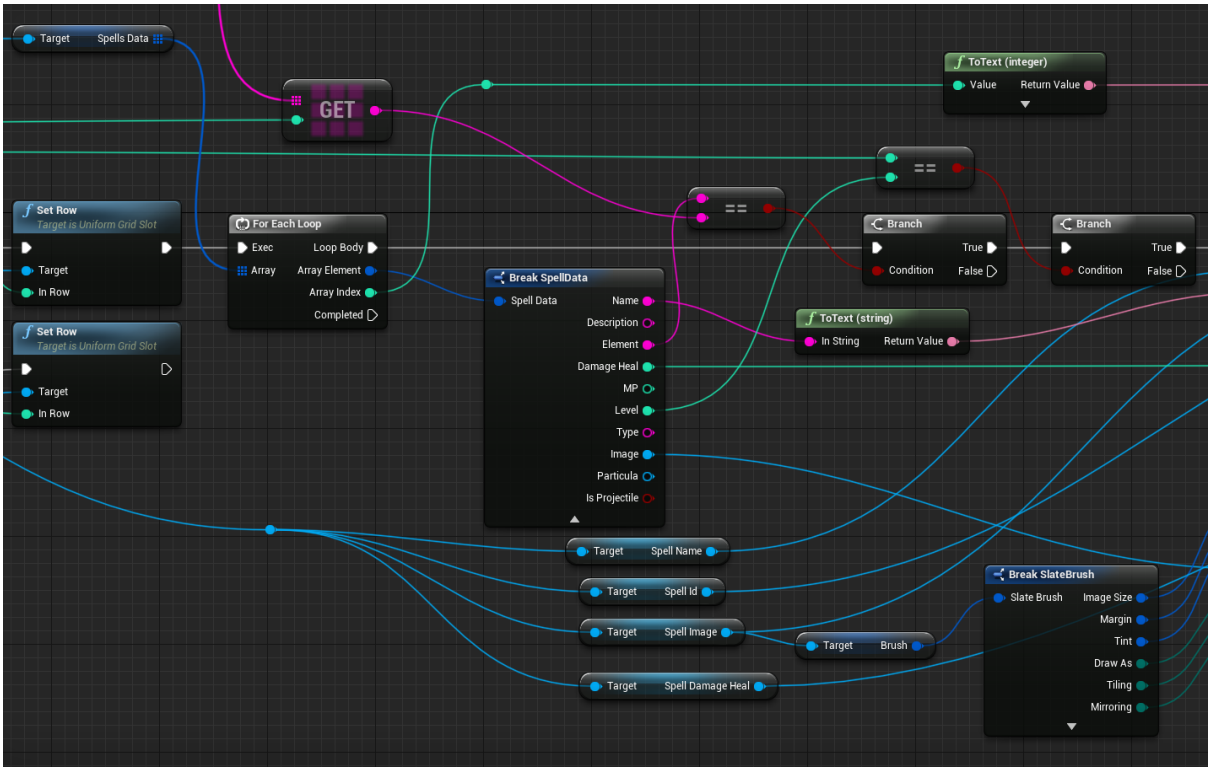


Figura 222: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció BeginPlay 4

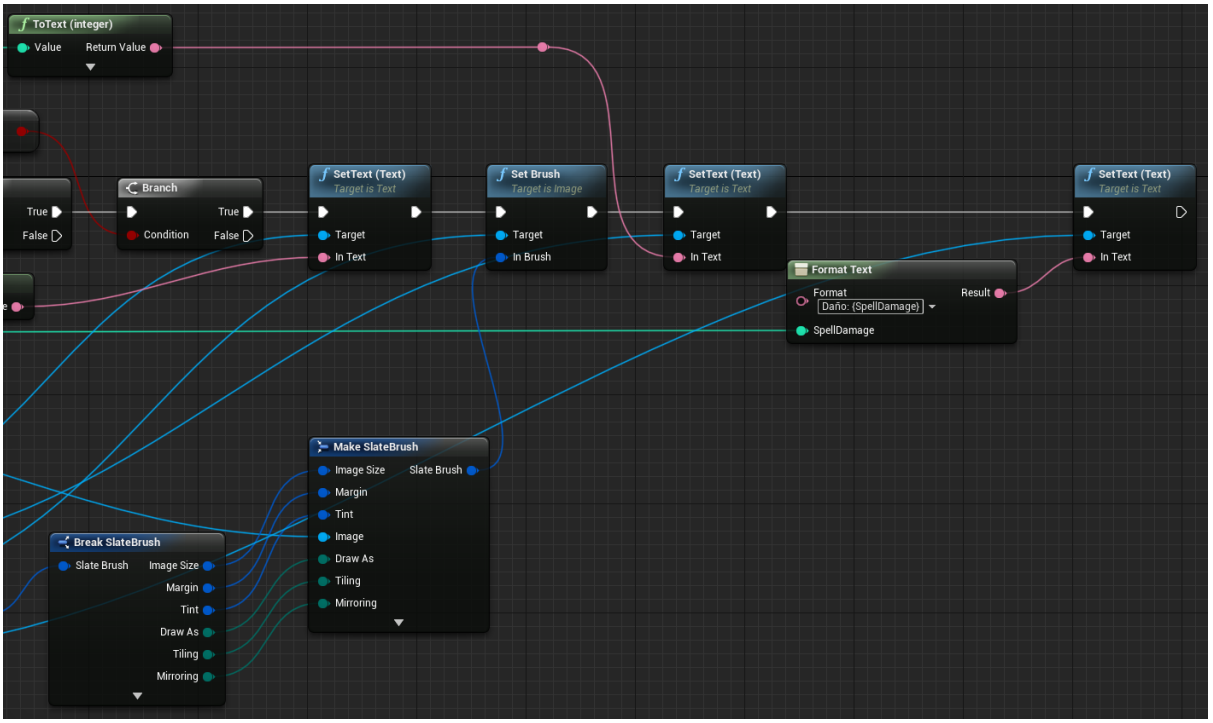


Figura 223: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció BeginPlay 5

- Update Menu

  Aquesta funció (veure Figura 224) s'encarrega de refrescar les imatges de l'escena. Aquesta funció fa:

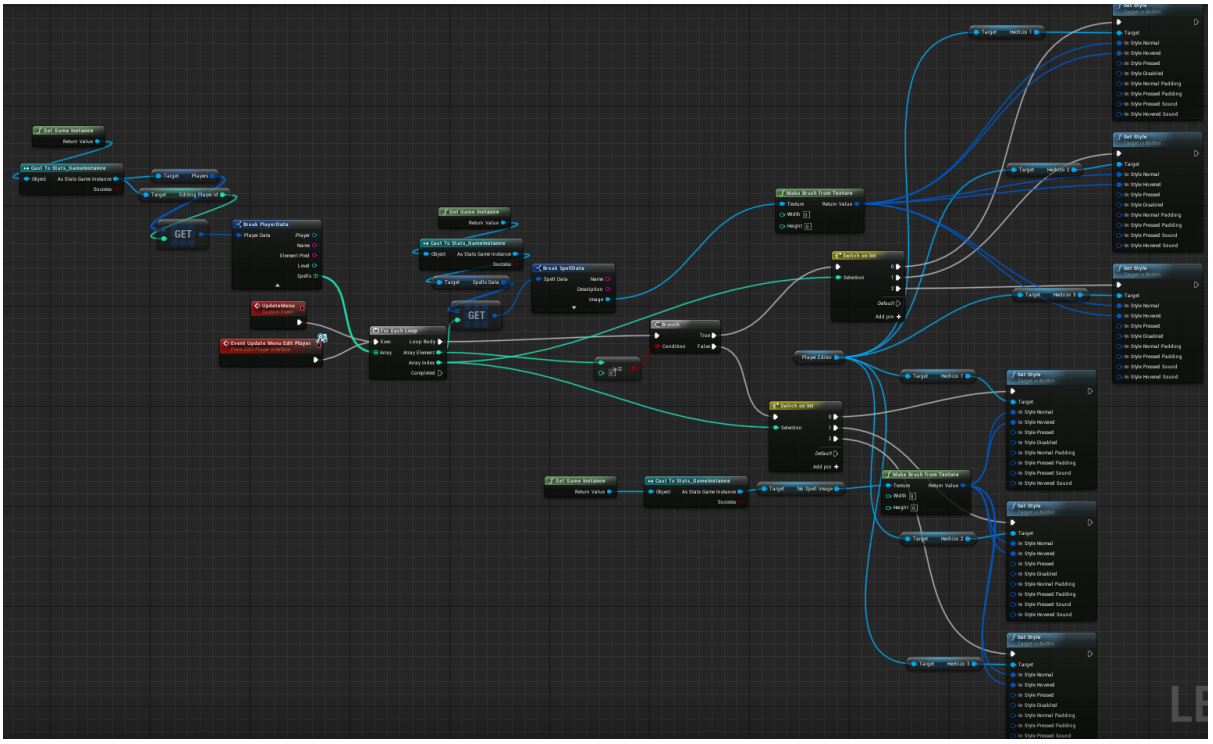


Figura 224: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció UpdateMenu / Event Update Menu Edit Player

  - Agafes els Spells del jugador que s'està editant.

  - Utilitzant un bucle que miri l'array dels Spells del personatge, per cada iteració, busca el Spell a l'array de SpellsData utilitzant el valor de la variable. Veure Figura 225.

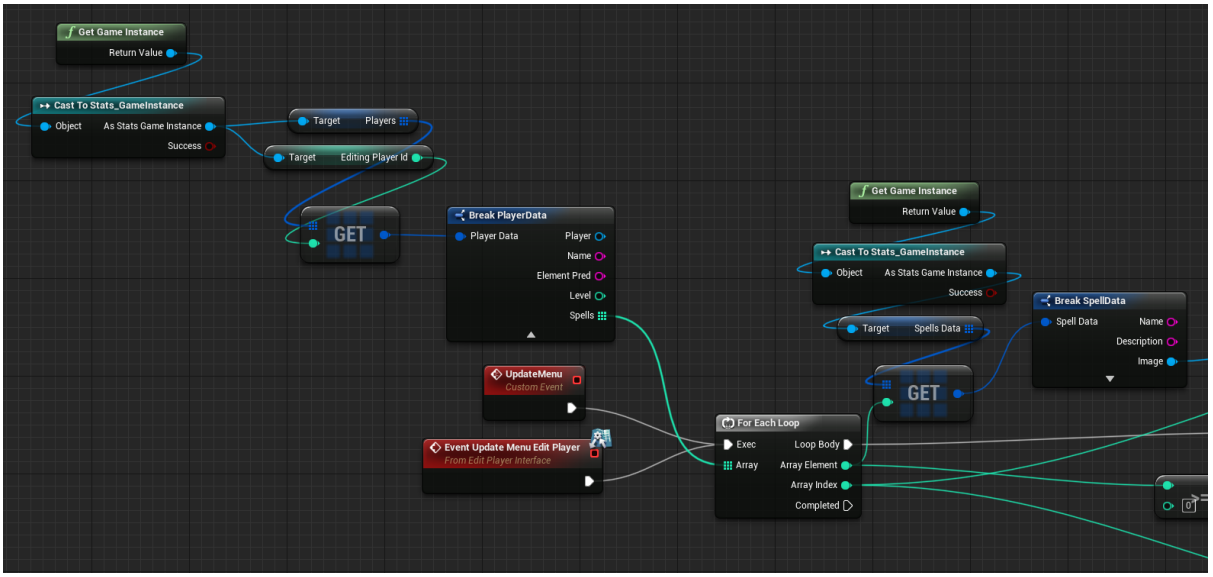


Figura 225: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció UpdateMenu / Event Update Menu Edit Player 1

- Ara, utilitzant un switch que detecti l'índex, modifiquem la imatge dels elements hechizo 1, hechizo 2 o hechizo 3, referents als botons situats a la part inferior del viewport del widget PlayerEditor. Veure Figura 226.

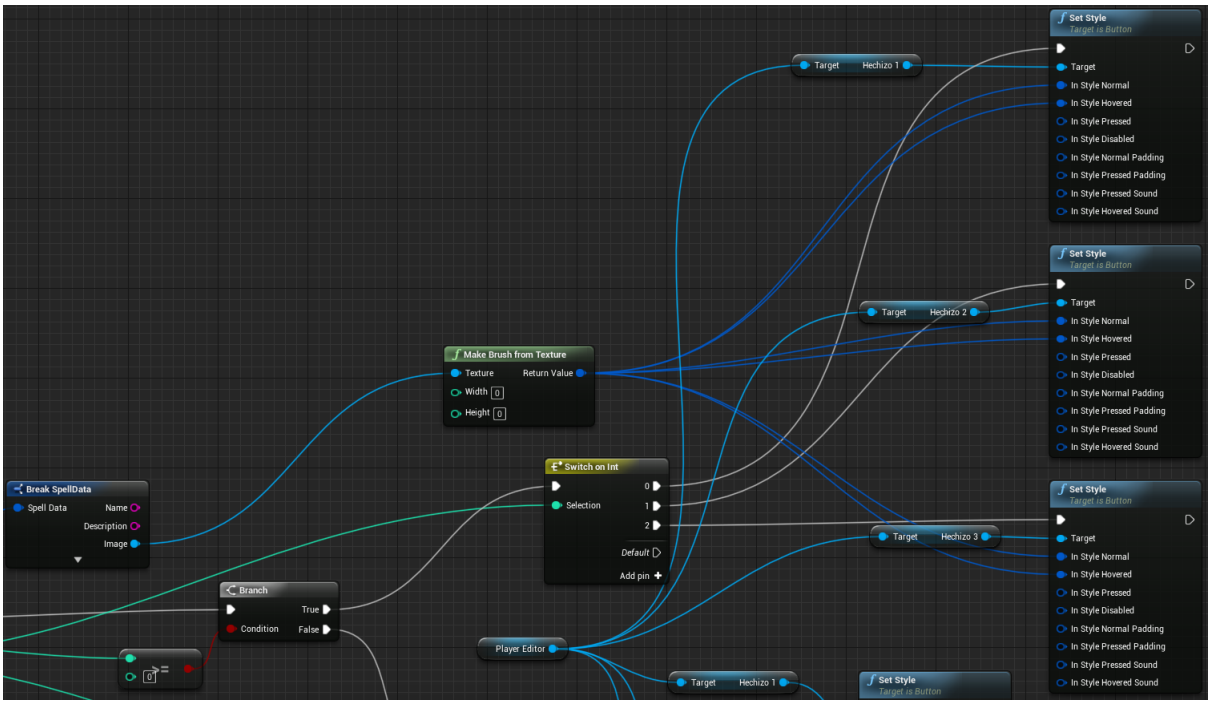


Figura 226: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció UpdateMenu / Event Update Menu Edit Player 2

- En el cas de que el valor de l'element sigui menor a 0, es guardarà una imatge d'una creu, referent al fet que no hi ha cap Spell en aquell índex. Veure Figura 227.

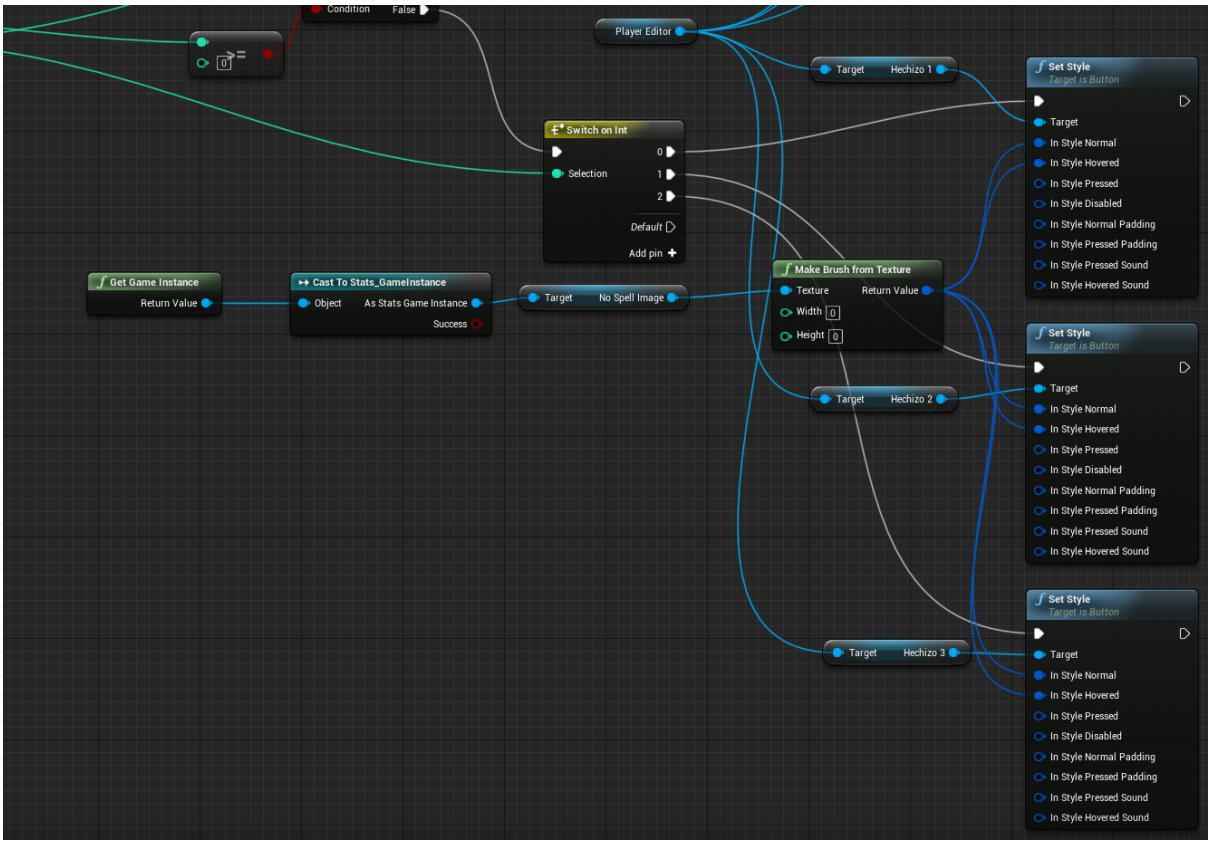


Figura 227: Captura de l'actor del level Blueprint de l'escena EditPlayer, de la funció UpdateMenu / Event Update Menu Edit Player 3

6.9.4. Menú ArmyEditor

Aquest menú (veure Figura 228) és el menú que llista els personatges i mostra informació rellevant. Aquest menú consta de diferents widgets.

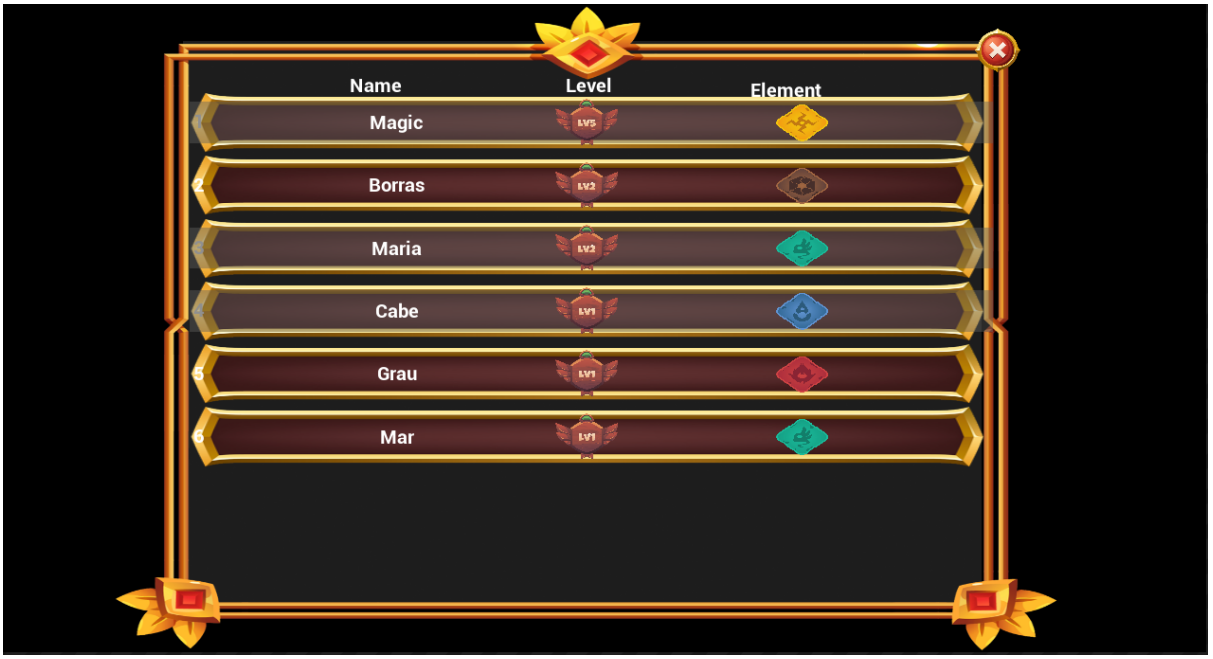


Figura 228: Captura del resultat mostrat per pantalla al entrar al Menú ArmyEditor

6.9.4.1. Widget ArmyGrid

Aquest widget (veure Figures 229 i 230) és l'encarregat de crear la base del menú.

Figura 229: Captura del widget ArmyGrid

Figura 230: Captura de les variables del widget ArmyGrid

Aquest widget disposa d'un apartat a comentar, aquests és:

- Elements importants

  Similar al widget PlayerEditor, aquest també es tracta del component "UniformGridPanel", i es comentarà a l'apartat referent a la creació d'aquest widget, 6.9.4.4.

6.9.4.2. Widget ArmyGridNothing

Aquest widget (veure Figura 231) és un widget buit que utilitzarem un cop llistats tots els personatges.

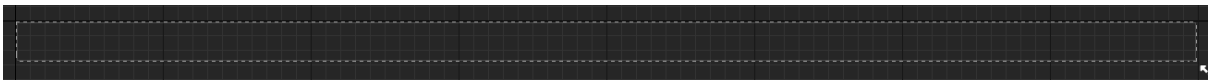


Figura 231: Captura del widget ArmyGridNothing

6.9.4.3. Widget ArmyGridPersonalStats

Aquest widget (veure Figures 232 i 233) és el widget encarregat de mostrar tots els personatges amb la seva informació.

Figura 232: Captura del widget ArmyGridPersonalStats

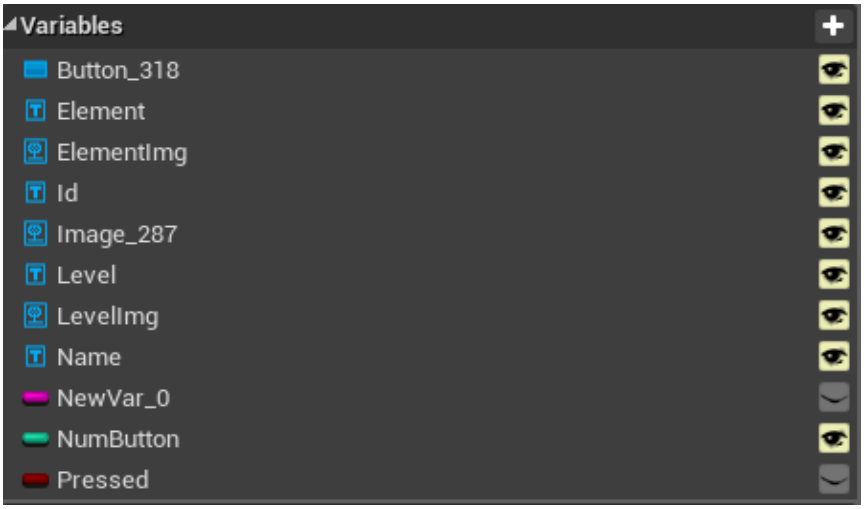


Figura 233: Captura de les variables del widget ArmyGridPersonalStats

Aquest widget disposa de diferents apartats a comentar, aquests són:

- Botó

  Aquest apartat (veure Figura 234) s'encarrega que el jugador pugui clicar aquest personatge i agregar-lo a la llista. Aquesta funció fa:

  - Cridar la funció AddPlayer del gestor.

  - Obrir el nivell PlayerSelected.

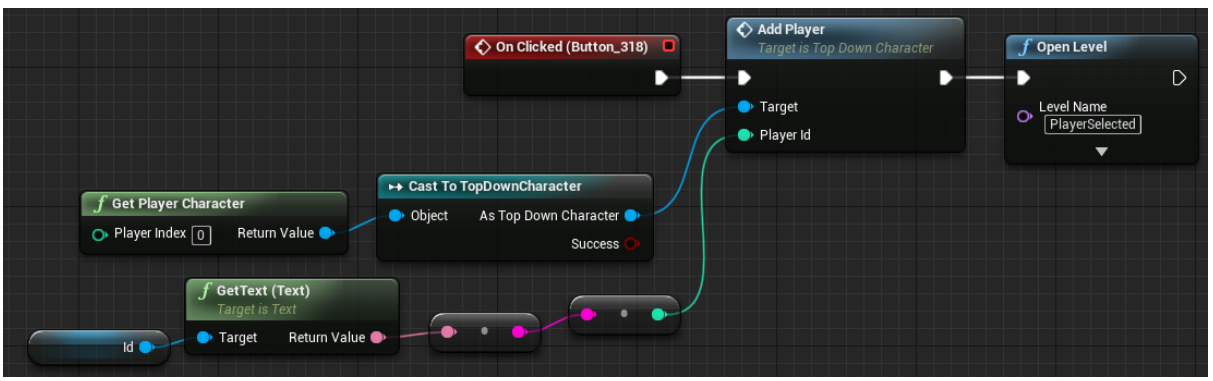


Figura 233: Captura del Graph del widget ArmyGridPersonalStats

6.9.4.4. Creació widget

Per la creació d'aquest menú final, s'ha implementat un apartat a la funció de BeginPlay del gestor (veure Figura 234). Aquest apartat fa:

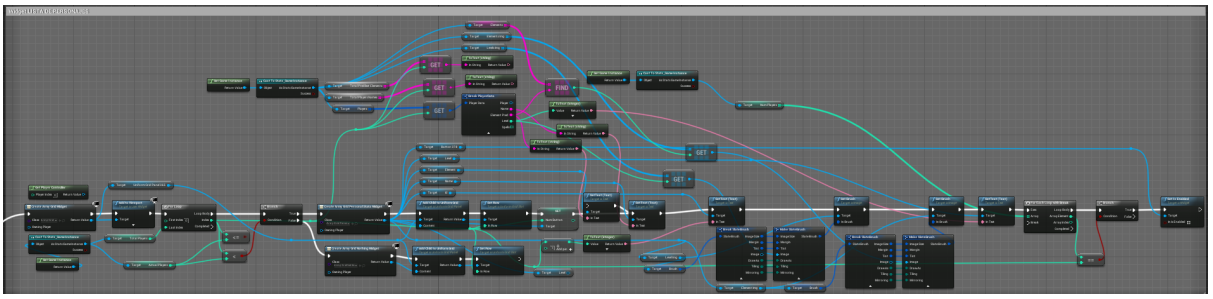


Figura 234: Captura de l'actor del gestor, la funció BeginPlay

- Crea el widget ArmyGrid i el mostra al viewport.

- Es fa un bucle a on la variable max és el valor de la variable de la instància de joc anomenada TotalPlayers.

- En aquest bucle es mira si l'índex és més petit que la variable de ActualPlayers de la instància de joc.

- En el cas que sigui fals, crea el widget ArmyGridNothing i l'afegeix a l'índex que toca com a fill de la variable UniformGrid del widget ArmyGrid.

- En cas de ser cert, es crea un widget ArmyGridPersonalStats i també l'afegeix com a fill de la variable UniformGrid del widget ArmyGrid. Veure Figura 235.

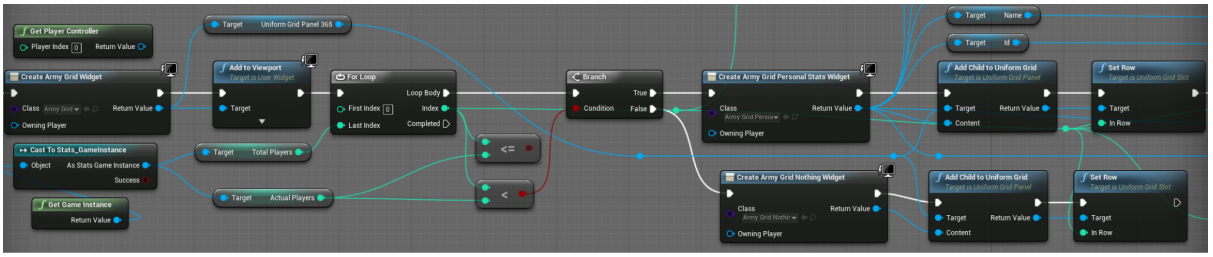


Figura 235: Captura de l'actor del gestor, la funció BeginPlay 1

- Un cop tenim el widget ArmyGridPersonalStats creat, agafem les variables de Button, Level, Element, Name i Id. A part, també agafem de la instància de joc l'array elements, l'array ElementsImg, l'array de NumPlayers i l'array de Players, del qual agafarem el personatge equivalent a l'índex del bucle per obtenir les variables del nom, Element i Level. Veure Figura 236.

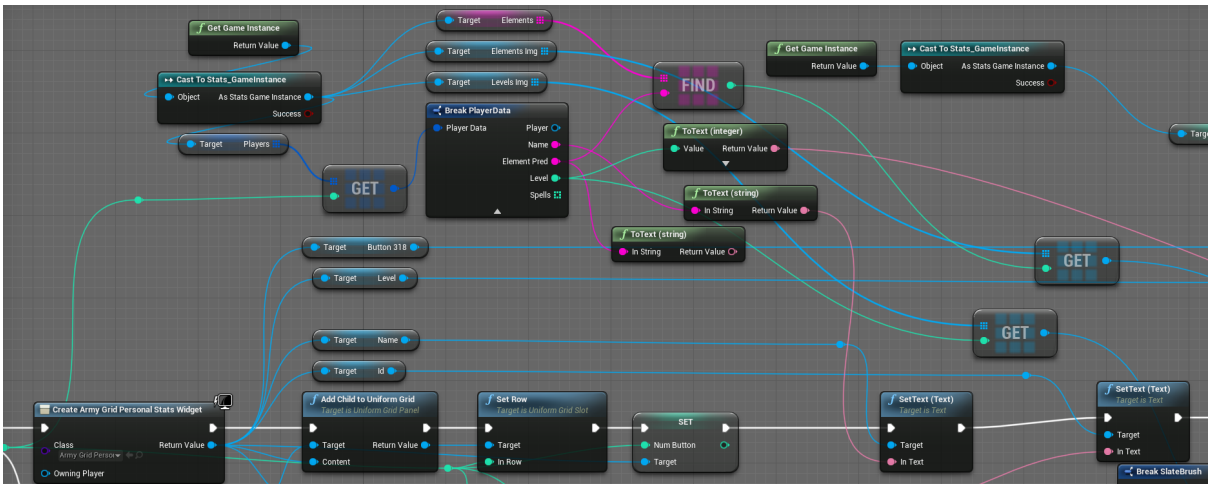


Figura 236: Captura de l'actor del gestor, la funció BeginPlay 2

- Per acabar, modificarem les variables extretes del widget ArmyGridPersonalStats obtingudes anteriorment. Per fer-ho utilitzarem els nodes "SetText" pels textos i "SetBrush" per les imatges i si es clicable utilitzant el node "SetIsEnabled". Veure Figura 237.

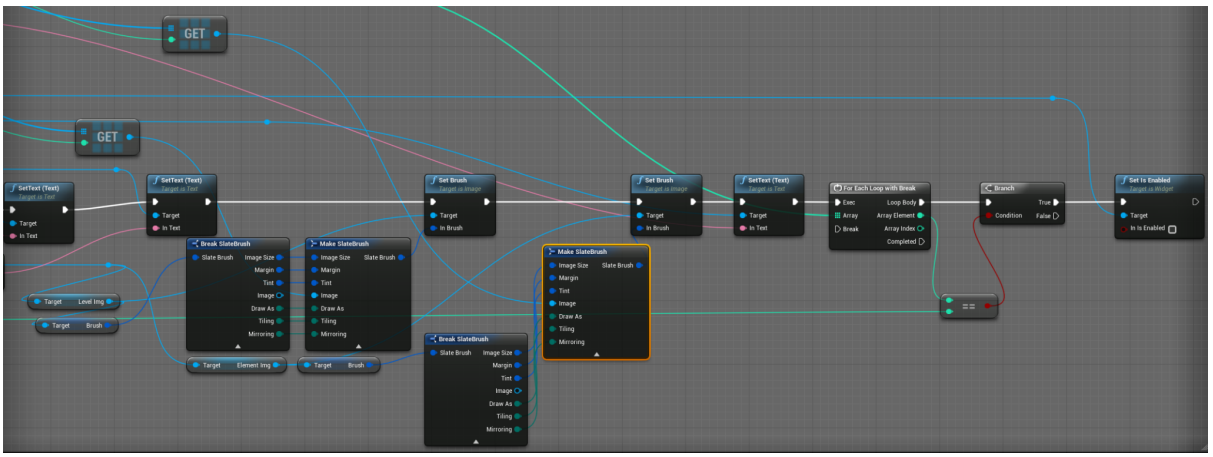


Figura 237: Captura de l'actor del gestor, la funció BeginPlay 3

6.9.5. Widget UserStats

Aquest widget (veure Figures 238 i 239) és un HUD que es mostra mentre el jugador està ingame i apuntant a un enemic.

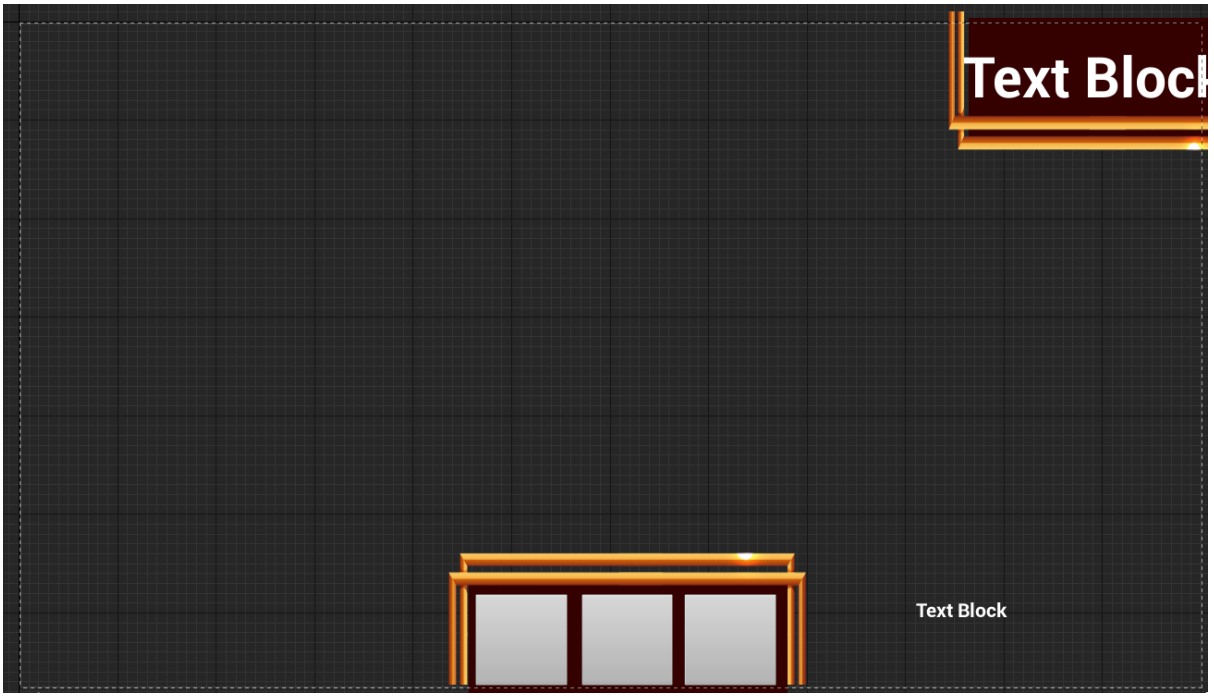


Figura 238: Captura del widget UserStats

Figura 239: Captura de les variables del widget UserStats

Aquest widget disposa de diferents apartats a comentar, aquests són:

- Text Block 1

  El Text Block de la cantonada superior esquerra de la Figura 238, s'encarrega de mostrar per pantalla el percentatge d'encertar un atac. Per aconseguir-ho, vam fer una funció dins del widget UserStats que agafés la variable PercentatgeHit i la mostrés. Veure Figura 240.

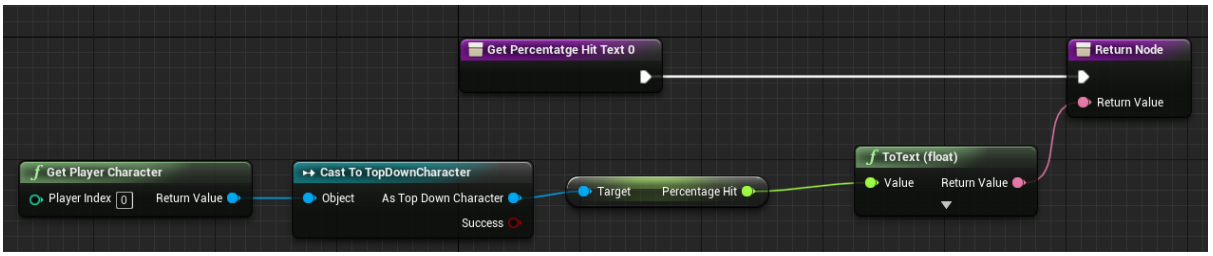


Figura 240: Captura del Graph de la funció vinculada al text 1 del widget UserStats

- Text Block 2

  El Text Block de la cantonada inferior esquerra de la Figura 238, s'encarrega de mostrar per pantalla les accions fetes. Per aconseguir-ho, vam fer una funció dins del widget UserStats que agafés el personatge que el jugador està utilitzant del gestor, i del seu actor agafarem la variable referent a les accions fetes per aquest personatge. Veure Figura 241.

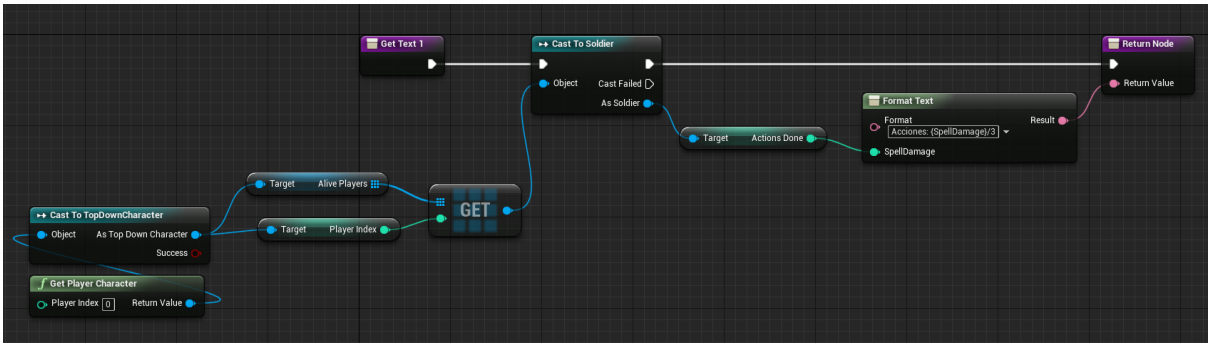


Figura 241: Captura del Graph de la funció vinculada al text 2 del widget UserStats

- Botons

Els botons (veure Figura 242) que es veuen a la Figura 238, s'encarreguen d'activar l'atac en ser clicats. Depenent del botó, s'activa un Spell o un altre. Aquestes funcions fan:

- Cridar la funció del mateix Blueprint anomenada CastSpell passant-li l'id del Spell.

- Aquesta funció agafa el personatge que està apuntant, busca el Spell que ha utilitzat, agafant com a índex el idSpell, obtenint la posició de l'array de SpellData.

- Seguit, agafem l'array de SpellsData i obtenim el Spell necesari a partir del valor anterior. Aquest Spell, ens el guardem a la variable CastingSpell.

- Cridem la funció de SoldierAttack al gestor.

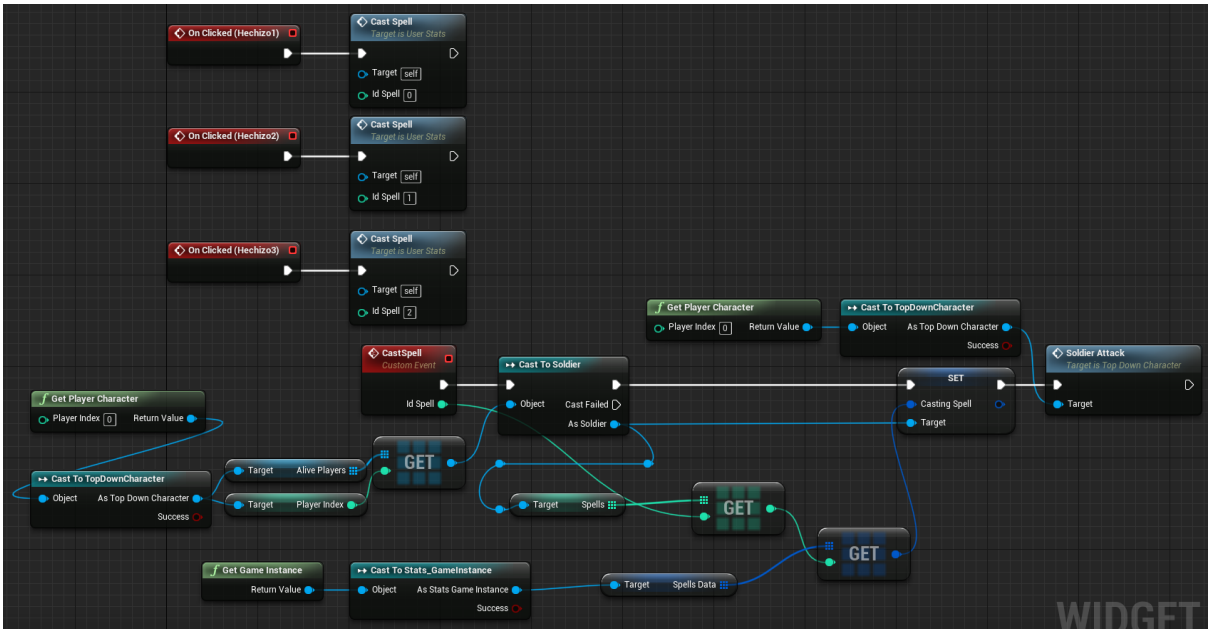


Figura 242: Captura del Graph de les funcions dels botons del widget UserStats

6.9.6. Altres Widgets

A part dels widgets comentats, altres widgets menys importants són:

- Menú principal

  Aquest widget es mostra al menú principal. Les seves funcions són accedir a l'editor de l'equip i sortir del joc. Veure Figures 243 i 244.

Figura 243: Captura del widget MainMenu

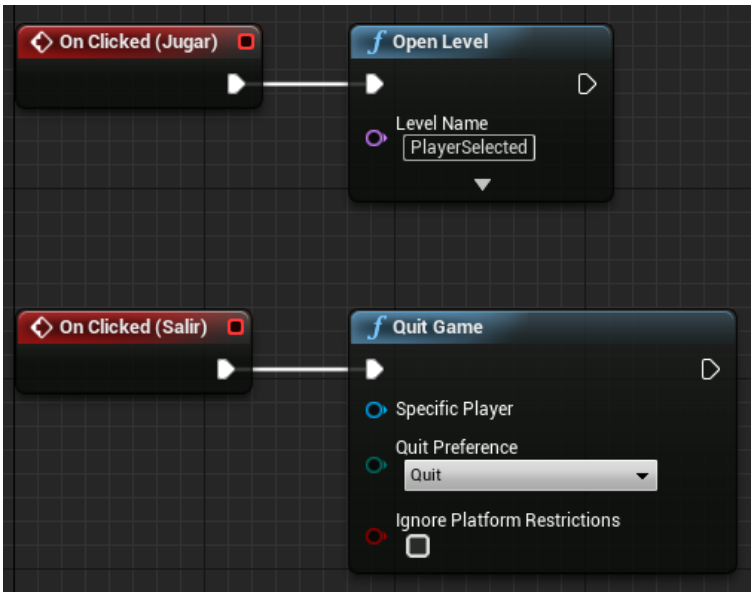


Figura 244: Captura del Graph del widget MainMenu

- Menú victoria

Aquest widget es mostra quan el jugador supera un nivell. La seva funcion és enviar al jugador de nou al menú principal. Veure Figures 245 i 246.

Figura 245: Captura del widget WinMenu

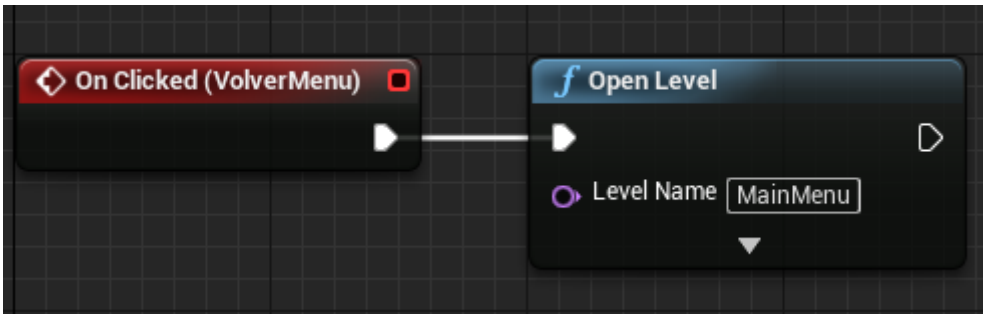


Figura 246: Captura del Graph del widget WinMenu

- Menú mort del jugador

  Aquest widget es mostra quan el jugador és eliminat en un nivell. La seva funció és enviar al jugador de nou al menú principal. Veure Figures 247 i 248.

Figura 247: Captura del widget LoseMenu

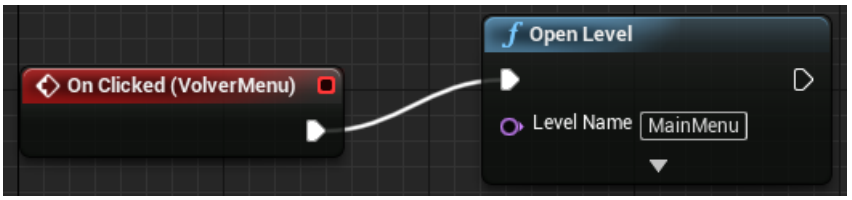


Figura 248: Captura del Graph del widget LoseMenu

- Menú Pausa:

  Aquest widget es mostra quan el jugador és eliminat en un nivell. Les seves funcions són enviar al jugador de nou al menú principal i reprendre la partida. Veure Figures 249 i 250.

Figura 249: Captura del widget PauseMenu

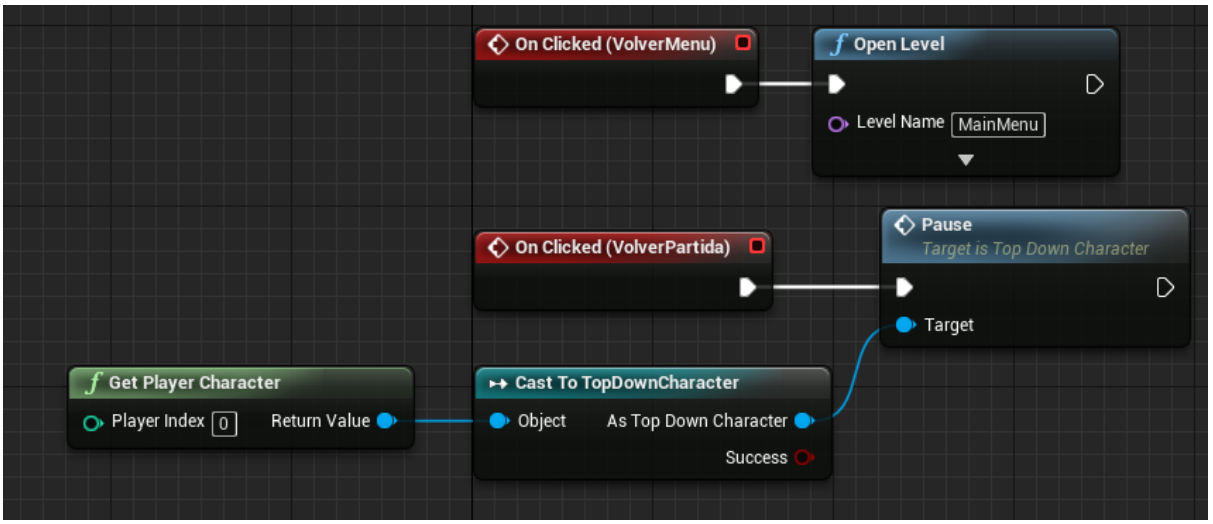


Figura 250: Captura del Graph del widget PauseMenu

7. Resultats

7.1. Legislació i normativa vigent

El prototipus desenvolupat no presenta cap irregularitat en cap aspecte legislatiu. El joc no guarda cap informació de caràcter personal del jugador, per la qual cosa no s'aplica la LOPD (Llei Orgànica de Protecció de Dades) i, a causa que el joc no comporta cap activitat econòmica, tampoc s'aplica la LSSICE (Llei de Serveis de la Societat de la Informació i Comerç Electrònic).

Per una altra banda, els problemes relacionats amb els drets d'autor tampoc suposen un problema. El prototipus utilitza recursos externs com són algunes animacions de Mixamo i alguns paquets donats a Unreal MarketPlace, però tot el contingut d'aquestes pàgines és gratuït i totalment lliure.

Finalment, si el projecte continués endavant i es decidís posar a la venda a algun mercat digital, Epic Games, creadora d'Unreal Engine, imposaria uns royalties del 5% dels beneficis si arribéssim al milió de dòlars de beneficis totals.

7.2. Classificació PEGI

Per obtenir una classificació d'edat i uns descriptors de contingut oficials a Europa, s'utilitza el sistema PEGI (Pan European Game Information). Aquest sistema classifica els jocs per edat recomanada i també informa l'usuari del contingut que pot trobar al joc. Per una altra banda, aquesta classificació no és de caràcter obligatori, sinó que té un motiu purament informatiu i de recomanació (no exclou la possibilitat de jugar de cap perfil de jugador).

Com es pot veure a la Figura 626, PEGI classifica els jocs en 5 trams d'edat (3, 7, 12, 16 i 18 anys) i atorga diferents descriptors de contingut, com, per exemple, la presència de drogues, sexe, violència o discriminació.

En el cas del joc desenvolupant, l'etiqueta d'edat hauria de ser la PEGI 16. Els principals motius pels quals considerem que cal una edat superior a 16 anys és perquè aquest joc conté un combat amb eliminacions de personatges. No obstant això, no hi ha presència de sang en cap moment, per tant, impedeix categoritzar el joc com a PEGI 18.

7.3. Resultat final

En aquest apartat es mostren múltiples captures del joc en diferents situacions i menús.

Figura 251: Captura del menú principal del joc

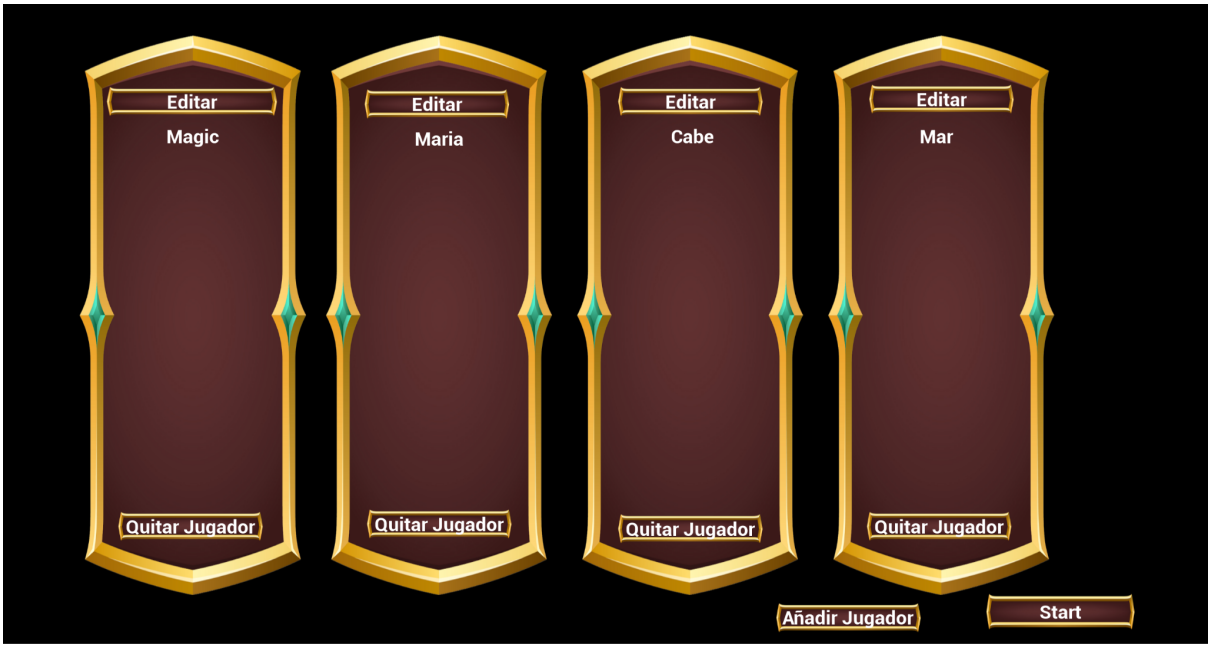


Figura 252: Captura del menú de selecció de jugadors

Figura 253: Captura del menú per editar un personatge

Figura 254: Captura in-game

Figura 255: Captura in-game d'un atac

Figura 256: Captura in-game d'un atac

Figura 257: Captura in-game de la vista de l'escenari

Figura 257: Captura in-game de la vista de l'escenari

Figura 258: Captura in-game al rebre un atac

Figura 259: Captura in-game al rebre un atac

Figura 260: Captura in-game victoria jugador

Figura 261: Captura del menú de selecció de nivell un cop guanyat el primer nivell

Figura 262: Captura in-game de la vista de l’escenari

Figura 263: Captura in-game de la vista de l’escenari

Figura 264: Captura in-game del joc

Figura 265: Captura in-game d’un atac

8. Conclusions

8.1. Valoració final del projecte

Per la meva part, estic molt satisfet amb el resultat final del prototipus. Crec que els objectius i propòsits que es van determinar al principi del projecte han estat assolits completament, i que el resultat final del joc és testimoni del treball que s'ha portat a terme i de la dedicació que ha estat invertida durant els mesos de duració.

El projecte m'ha fet aprendre moltíssim sobre el desenvolupament seriós del prototipus d'un joc certament ambiciós. Començant pels requisits d'un disseny que resulti prou atractiu, passant per una implementació complexa, amb molts elements que necessiten un desenvolupament simultani i paral·lel, fins a arribar a una fase de correcció d'errors o debugging que comporta una inversió de temps igual o superior a les altres dues.

He après a dissenyar i implementar un dels elements més importants i que més donen forma a un videojoc: els enemics. Per una banda, implementar un Behaviour Tree, i tot i tenir coneixements previs, ha estat una experiència complicada i laboriosa. Mantenir la constància i la dedicació requerida durant les parts més denses i dures del desenvolupament (debugging) ha estat un dels principals problemes, ja que passar una gran quantitat d'hores en les quals no es veuen resultats fructífers produeix certa desmotivació i falta de ganes de continuar amb el projecte.

El resultat, a més, és acceptablement bo donat l'àmbit del prototipus, oferint i proporcionant al jugador múltiples possibilitats per trobar un mètode d'acció que s'adapti a la col·locació dels enemics, i fent que els enemics, tot i seguir un patró, tinguin certa vida i intel·ligència.

També he après molt sobre els widgets i la facilitat que poden aportar al desenvolupament del joc si es coneixen els nodes necessaris.

Tot i tenir un cert nivell en Unreal Engine, he notat que la meva habilitat amb aquest motor ha crescut de forma notable. He après a utilitzar molts nodes i actors particulars del motor que enriqueixen la qualitat final de la implementació i, a la vegada, la simplifiquen. Encara que ja havia dut a terme algun projecte del cicle superior i del grau fent ús d'Unreal, no havia pogut aprofundir i explotar a aquesta escala les opcions que ofereix el motor i el seu llenguatge visual.

Finalment, considero que he après molt sobre el desenvolupament general d'un videojoc i, en concret, de les mecàniques. Per tant, considero que aquest projecte ha servit per experimentar una evolució en l'àmbit del disseny la programació molt positiva, i que el resultat ha estat molt satisfactori.

8.2. Desviacions de la planificació

Tot i que no vaig tenir cap desviació de la planificació inicial, sí que he notat que per culpa de diferents errors als quals els hi vaig dedicar molt temps, vaig haver de deixar fora diferents coses que des d'un principi volia implementar. La planificació final va patir els canvis que es poden veure al diagrama de Gantt de la Figura 266.

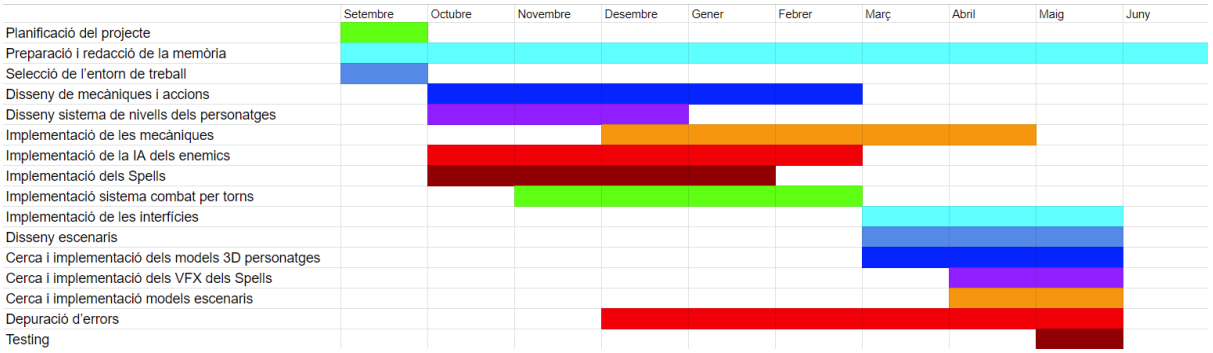


Figura 266: Diagrama de Gantt amb el cronograma estimat durant l'inici del projecte.

9. Treball futur

Amb el prototipus de joc implementat, s'ha pogut demostrar el potencial que tindria el joc finalitzat si aquest tingués un desenvolupament amb més temps i recursos. En qualsevol cas, si es pogués portar a terme aquesta continuació del desenvolupament, hi ha diferents aparts on el joc podria créixer i millorar:

- Afegir més nivells: Els nivells actuals trobo que són molt pocs per considerar-se un joc final, per tant, necesitaria un augment de nivells.

- Sistema de MP: aquest sistema tindria com a objectiu tenir una barra consumible que representés el manà. Això serviria per forçar al jugador a ser més prudent amb els atacs, ja que cadascun tindrìa un cert cost i no es podria tirar si et quedessis en negatiu, a part que si es falla un atac, es continua perdent el manà.

- Afegir diferents tipus d'enemics: Actualment, sí que és veritat que tinc diferents enemics, però aquests tenen la mateixa mesh i només varia el material. Ens agradaria augmentar els tipus diferents d'enemics.

- Combat cos a cos: Tot i que el sistema està fet per combats majoritàriament a distància, la majoria dels jocs també disposen del combat cos a cos, ja que permet al jugador jugar d'una manera diferent i augmenta les possibilitats.

- Afegir objectes i armadures: Continuant amb el punt anterior, aquests tipus de jocs acostumen a tenir diferents objectes. En el nostre cas, hem pensat en una manera d'aplicar una mecànica com aquesta per fabricar armadura o equipament que puguin portar els personatges.

- Eliminar personatges quan moren: Actualment, no es reflecteix el concepte que els personatges tenen una sola vida. Per aquesta raó, de cara al futur, seria necessari aplicar-ho.

- Afegir diferents models 3D i creació automàtica dels personatges: Actualment, els personatges ja estan creats i no apareixen més, i com en un futur s'eliminaran els jugadors al morir, s'ha d'implementar una altra manera d'aconseguir noves unitats.

- Afegir un personatge 3D al menú PlayerSelected: Com els personatges seran diferents físicament i tindran equipament, és necessari mostrar quin personatge s'ha seleccionat, ja que enrecordar-se per l'equipament i el model és molt més senzill a fer-ho només pel nom.

- Afegir sons: en el prototipus, no s'han implementat sons, per tant, un punt necessari ha de ser la implementació d'aquests.

- Afegir feedback: en el prototipus actual, el feedback que rep el jugador quan es canvia de torn, quan rep mal, etc, és mínim. De cara a la finalització, és necessari implementar més feedback.

10. Bibliografía

1. Epic Games, Inc. Unreal Engine 4.22 Documentation.

   https://forums.unrealengine.com/t/unreal-engine-4-4-22-offline-documentation/126935

2. Epic Games, Inc. Unreal Engine Forum.

   https://forums.unrealengine.com/

3. Epic Games, Inc. Canal de YouTube d'Unreal Engine.

   https://www.youtube.com/c/UnrealEngine

4. Adobe Systems, Inc. Mixamo.

   https://www.mixamo.com/

5. freepik.

   https://www.freepik.com

6. Pinterest.

   https://www.pinterest.es

11. Annexos

11.1. Projecte d'Unreal Engine

Com a Annex a aquest treball, s'adjunta el projecte sener d'Unreal Engine, juntament amb l'arxiu executable. Per poder obrir el projecte, cal descomprimir la carpeta "WizardsOfFate_TFG2023.zip" i, a dins de la carpeta "WizardsOfFate", hi ha l'arxiu "WizardsOfFate.uproject", el qual es pot obrir amb Unreal Engine. Respecte a l'executable, per poder obrir-lo, cal descomprimir la carpeta "WizardsOfFate_TFG2023_Ejecutable.zip", dins trobarem una altra carpeta anomenada "WindowsNoEditor", dins d'aquesta trobare l'executable "WizardsOfFate.exe", el qual amb un doble clic obrirà el joc.

Respecte als enllaços, el link del projecte es:
https://www.dropbox.com/s/8p6ijgnnz7ycfd8/WizardsOfFate_TFG2023.zip?dl=0

I el link de l'executable és:
https://drive.google.com/file/d/1nrWph-Gw7Yko5n-qoKTIgB9IPeg6wHOR/view?usp=sharing

Com a extra, hem gravat un gameplay del primer nivell, el link d'aquest és:
https://drive.google.com/file/d/1OpTEsRBkosImadNpaBc2PECtDUW9I2hh/view?usp=sharing

12. Manual d’usuari i instal·lació

El Joc només és jugable amb teclat i ratolí. Aquests són els controls:

- Moviment del jugador: Clic esquerre del ratolí
- Apuntar: Q
- Canviar de personatge: W
- Canviar d’objectiu (quan s’apunta): A / D.
- Atacar: Clic esquerre al botó del Spell escollit
- Menú Pausa: U