Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Disseny i Desenvolupament de Videojocs

Títol: Disseny i desenvolupament d'un videojoc de tipus "Boss Rush"

Document: Memòria

Alumne: Juan Torres Marí

Tutor: Gustavo Patow
Departament: Informàtica, Matemàtica Aplicada i Estadística
Àrea: Llenguatges i Sistemes Informàtics

Convocatòria (mes/any) Setembre / 2022

Contingut

1. Introducció i objectius

1.1 Introducció

Actualment, els videojocs amb mecàniques de combat acostumen a dividir l'espai de joc en zones, on el jugador primer es troba amb enemics bàsics i després al final de la zona es troba amb un boss que el jugador ha de matar per poder avançar a la següent zona. Però també existeix un gènere de jocs molt menys explotat, on ens saltem els enemics bàsics i anem directament a els combats contra bosses, que es caracteritzen per una dificultat elevada. Aquest gènere és el que es coneix com a "Boss Rush".

L'objectiu d'aquest projecte és dissenyar i desenvolupar un videojoc 2d de tipus "Boss Rush", on l'objectiu del joc és matar tots els bosses, calibrant la dificultat per a que sigui un repte, però assequible pel jugador.

1.2 Motivacions

A nivell professional, com hem mencionat al punt anterior, la principal motivació consisteix en desenvolupar un videojoc d'aquest gènere tan poc explotat, oferint una alternativa en 2d per a tots el jugadors als que li agraden aquest tipus de jocs amb dificultat elevada.

Per altra banda, com a jugador i desenvolupador, el projecte també a sorgit de les següents motivacions personals:

- Poder dissenyar Inteligencies Artificials relativament complexes desde zero.
- Aprofundir en l'ús i funcionament dels behaviour trees.
- Posar en pràctica el que he après al grau en un projecte real.

1.3 Propòsit i objectius del projecte

El propòsit d'aquest projecte és dissenyar i desenvolupar un videojoc 2d de tipus "Boss Rush", on l'objectiu del joc és matar tots els bosses, calibrant la dificultat per a que sigui un repte, però assequible pel jugador.

Concretament, els objectius són:

- Aprofundir en l'ús i funcionament dels behaviour trees.
- Estudiar el desenvolupament de videojocs 2d de combat amb Unity.
- Dissenyar intel·ligències artificials amb comportaments interessants per als bosses.
- Millorar el meu nivell de programació.

1.4 Distribució de tasques

A causa que l'objectiu principal del projecte és el desenvolupament d'intel·ligències artificials interessants, hem decidit enfocarnos sobretot en la part més técnica del desenvolupament del videojoc.

Estètica	10%
Narrativa	5%
Mecàniques	30%
Tecnologia	55%

El percentatge referent a l'estètica es refereix a el disseny de diferents efectes de partícules i d'algunes animacions dels enemics per tal de millorar el feedback cap al jugador.

La part narrativa fa referència a la petita història que hem pensat per donar un sentit general i una ambientació al joc, sense arribar a aprofundir-hi molt.

Pel que fa a les mecàniques seria tot el disseny conceptual tant de les accions que pot realitzar el jugador, com de les accions que poden realitzar els enemics, i com aquestes interactuen entre si.

Finalment, l'apartat de tecnologia fa referència a tota la implementació a Unity, desde el disseny del codi fins a la implementació de les inteligencies artificials.

2. Estudi de viabilitat

Abans de començar un projecte, cal valorar la viabilitat d'aquest, per determinar si disposem dels recursos necessaris per completar el desenvolupament del projecte sense problemes i per no malgastar recursos i temps en el desenvolupament d'un videojoc sense mercat o amb un mercat ja molt explotat.

Per tal de fer aquesta valoració, en aquest apartat calcularem el cost de desenvolupament, realitzarem un estudi de mercat i definirem el públic objectiu del nostre joc.

2.1 Recursos necessaris i viabilitat

2.1.1 Recursos tècnics

Tots els recursos utilitzats per al desenvolupament d'aquest projecte son gratuits o ja els haviem adquirirt abans de començar el projecte.

A nivell de hardware hem utilitzat:

- Portatil
    - CPU: Intel-i7
    - Ram: 16gb
    - Gràfica: NVIDIA RTX 2060
- Comandament Xbox series X.

A nivell de software hem utilitzat:

- Sistema operatiu: Windows 10.
- Motor de jocs: Unity v2021.3.1f1
- IDE: Visual Studio 2020
- Programa art: Aseprite v1.2.17
- Inkscape

2.1.2 Recursos humans

Com a norma general, els videojocs solen ser desenvolupats per equips de persones on cada un te un rol i unes responsabilitats concretes. Aquests rols poden ser:

- **Dissenyador de joc:** encarregat de definir el joc, pensar la història, els personatges i els objectius.
- **Artista:** encarregat de tota la part visual del joc seguint les pautes indicades pel dissenyador de joc.
- **Enginyer de so:** encarregat de tots els efectes de so i la música del joc.
- **Programador:** encarregat de implementar tot el codi necessari per a fer funcionar els sistemes pensats pel dissenyador de joc.
- **Dissenyador de nivells:** encarregat de crear nivells interessants que explotin al màxim les mecàniques del joc.
- **Testers:** encarregats de provar el joc, donar feedback als desenvolupadors i reportar qualsevol tipus de bug que trobin.

Per al desenvolupament d'aquest prototipus, al ser un equip de només una persona, aquesta persona assumirà totes les tasques adients a cada rol, a excepció del testing, ja que donarem el joc a amics i familiars per obtenir feedback.

2.1.3 Viabilitat econòmica

Com ja hem comentat a l'apartat 2.1.1, tots els recursos utilitzats per al desenvolupament d'aquest projecte son gratuïts o ja els haviem adquirirt abans de començar el projecte, per tant el cost del projecte és de 0€, però si haguéssim de fer una estimació dels costos seria la següent:

Estimació cost recursos tècnics en base a payscale.com:

RECURS	**COST**
Portatil	2000€
Comandament	60€
Sistema Operatiu	100€
Software artístic	20€
Altres	0€
TOTAL	2180€

Estimació cost recursos humans:

- **Dissenyador de joc:** 16 €/hora.
- **Artista:** 15 €/hora.
- **Enginyer de so:** 12 €/hora.
- **Programador:** 15 €/hora.
- **Dissenyador de nivells:** 16 €/hora.
- **Testers:** 11 €/hora.

2.2 Estudi de Mercat

Abans de començar a desenvolupar un videojoc, és important analitzar i estudiar altres videojocs del mateix gènere o amb característiques similars per tal de treure els punts forts i punts millorables, i no fer simplement una versió pitjor d'un joc que ja existeix, sino agafar els aspectes millorables d'altres jocs i crear un nou producte ben diferenciat d'aquests.

Els principals videojocs que hem analitzat prèviament al desenvolupament del joc son els següents:

Titan Souls

Titan Souls és un joc d'acció en 2D amb vista top-down i estil pixel art. L'objectiu del jugador és matar als 19 bosses, anomenats "Titans". El que ens ha semblat més interessant d'aquest joc son les mecàniques de combat extremadament senzilles, el jugador només te una flecha i un punt de vida, però combinant aquestes mecàniques amb comportaments interessants dels bosses, crea un joc on l'objectiu principal del jugador és entendre el comportament dels bosses i trobar el seu punt feble per tal de poder matar-los. Aquest és el joc en que més ens hem inspirat a nivell jugable.

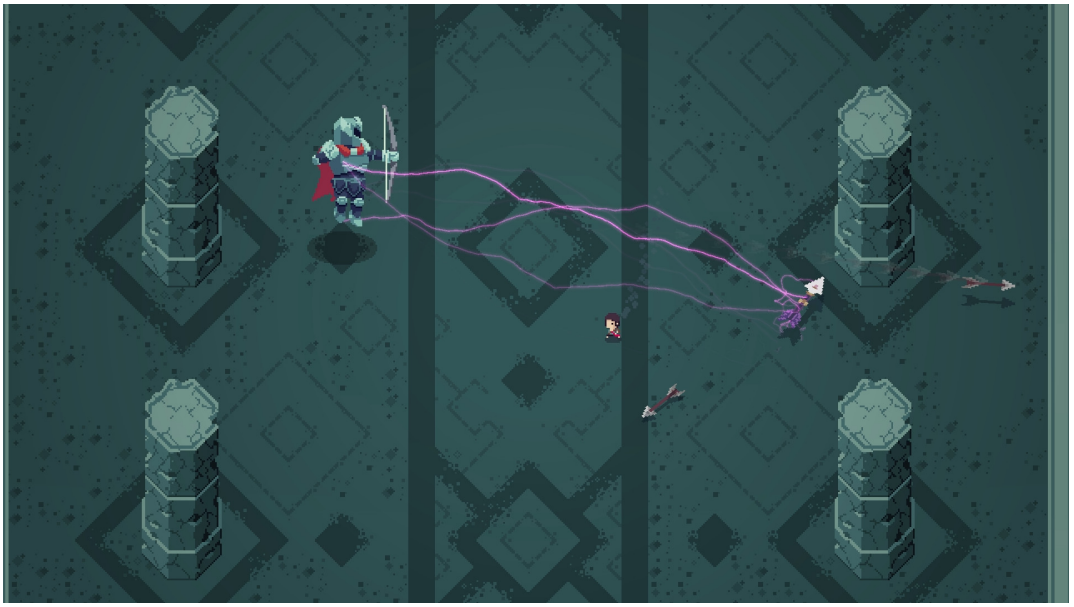

Figura 2.2.1: Captura del joc Titan Souls

Enter the Gungeon

Enter the Gungeon és un roguelite d'acció amb vista top-down on el jugador a de anar explorant la mazmorra anomenda "Gungeon" derrotant diferents enemics i bosses per poder avançar. D'aquest joc hem analitzat principalment l'estètica i el comportament dels bosses de tipus "bullet hell" i algunes mecàniques de combat, com per exemple la mecanica d'esquivar.

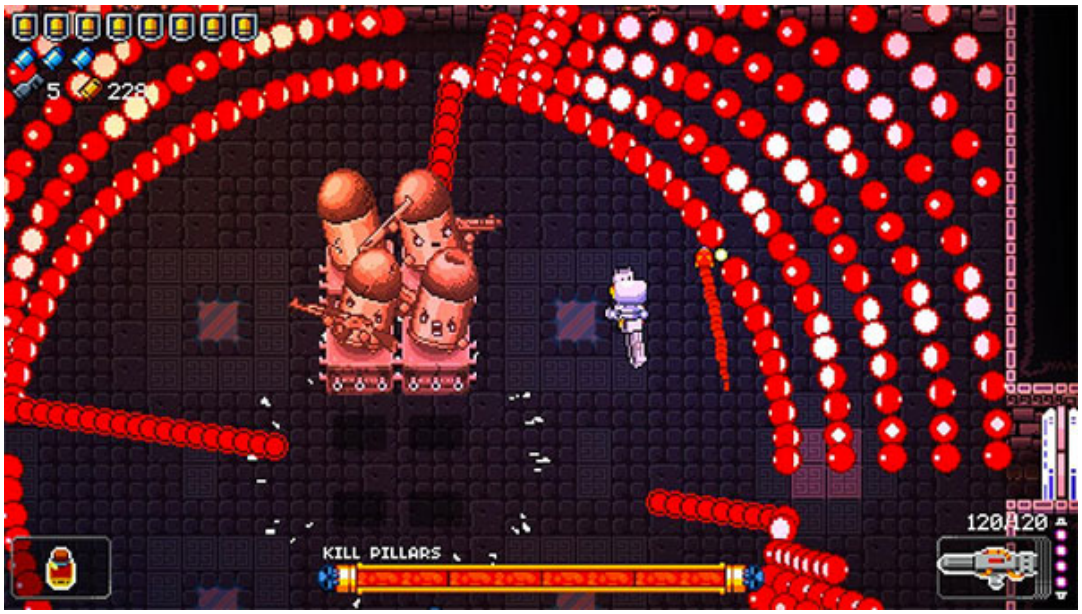

Figura 2.2.2: Captura del joc Enter the Gungeon

Hades

Hades és un roguelite de rol amb vista isomètrica on el jugador ha de aconseguir escapar de l'infern derrotant diferents enemics i bosses per poder avançar. D'aquest joc hem analitzat principalment el comportament dels bosses de combat a curta distància.

Figura 2.2.3: Captura del joc Hades

2.3 Públic objectiu i tipologia de jugador

El públic objectiu del joc son jugadors, en general joves, als que els hi agraden els jocs d'acció i que busquen la satisfacció obtinguda després de superar un repte difícil a base de intent i error.

3. Planificació

3.1 Diagrama de Gantt

	05/22				06/22				07/22				08/22			
Pluja d'idees																
Implementació de les mecàniques principals																
Buscar assets																
Disseny del primer boss																
Implementació del primer boss																
Disseny del segon boss																
Implementació del segon boss																
Disseny del tercer boss																
Implementació del tercer boss																
Proves i balanç																
Disseny i implementació de menus																
Últims detalls																
Documentació																

3.2 Eines de treball i programari

3.2.1 Motor de joc

Per al desenvolupament del joc vam barallar dues opcions, Unity i Unreal Engine, pero al ser un videojoc en 2d ens vam acabar decantant per el motor de Unity 2021, ja que compta amb un apartat especialitzat en el desenvolupament de videojocs en 2d, a més és el motor amb el que tenim més experiencia.

3.2.2 Editors de codi

Com a editor de codi hem utilitzat Visual Studio 2019, ja que és l'editor de codi que ve predeterminat amb Unity i permet debugar els projectes en temps real. A més ja tenim experiència amb aquest editor.

3.2.3 Programari artístic

Com a programari artístic hem utilitzat principalment Aseprite, una eina especialitzada per a la creació de sprites i animacions en pixel art. És una eina de pagament, pero ja comptavem amb una llicència abans de començar el projecte.

També hem utilitzat inkscape, una eina de disseny vectorial per crear els menús i alguns elements del UI.

3.2.4 Altres

Per la realització de la memòria hem utilitzat l'editor de text Google Docs.

Per tenir un historial de versions i mantenir una copia de seguretat al núvol hem utilitzat GitHub Desktop.

4. Marc de treball i conceptes previs

Abans de continuar, cal definir alguns conceptes que és clau entendre bé per poder seguir el projecte sense problemes.

- Concepte **"boss"**: boss és "cap" en anglès i es refereix a la persona que ocupa un càrrec elevat. Al mon dels videojocs, els bosses es caracteritzen per la seva dificultat elevada i una inteligencia artificial complexa. Alguns exemples de bosses molt icònics podrien ser el Bowser del videojocs de Mario, o el Ganondorf de la saga Zelda.

Figura 4.1: Captura del joc Super Mario 64

Figura 4.2: Captura del joc Zelda Breath of the Wild

- Concepte **“boss rush”**: és el gènere del nostre joc. Es basa principalment en saltar tota la part d’enemics bàsics i anar directament als combats contra bosses. És a dir, quan es mata a un boss, es passa directament al següent. Dos bons exemples de jocs basats principalment en aquest concepte son Titan Souls i Cuphead.

Figura 4.3: Captura del joc Titan Souls

Figura 4.4: Captura del joc Cuphead

Finalment cal comentar que l'equip del projecte està format per una sola persona.

5. Disseny del videojoc

5.1 Mecàniques

5.1.1 Espai de joc

Tenint en compte que el videojoc és de tipus boss rush, el disseny de l'espai de joc es va basar principalment en el disseny de les àrees on el jugador es barallarà amb els diferents bosses, amb l'objectiu de crear zones interessants amb elements que el jugador pugui utilitzar a favor seu per defensar-se de diferents atacs dels enemics.

Per al primer boss vam optar per un espai sense obstacles, ja que, al ser un boss molt agressiu de combat a curta distància i el jugador ha d'estar constantment fugint d'ell, no vam voler posar obstacles on el jugador es pogués quedar encallat.

Per altra banda, per al segon i tercer boss vam optar per posar algunes columnes per a què el jugador es pogués cobrir dels diferents atacs a distància que tenen aquests bosses. Veure les figures 5.1.1.1 i 5.1.1.2.

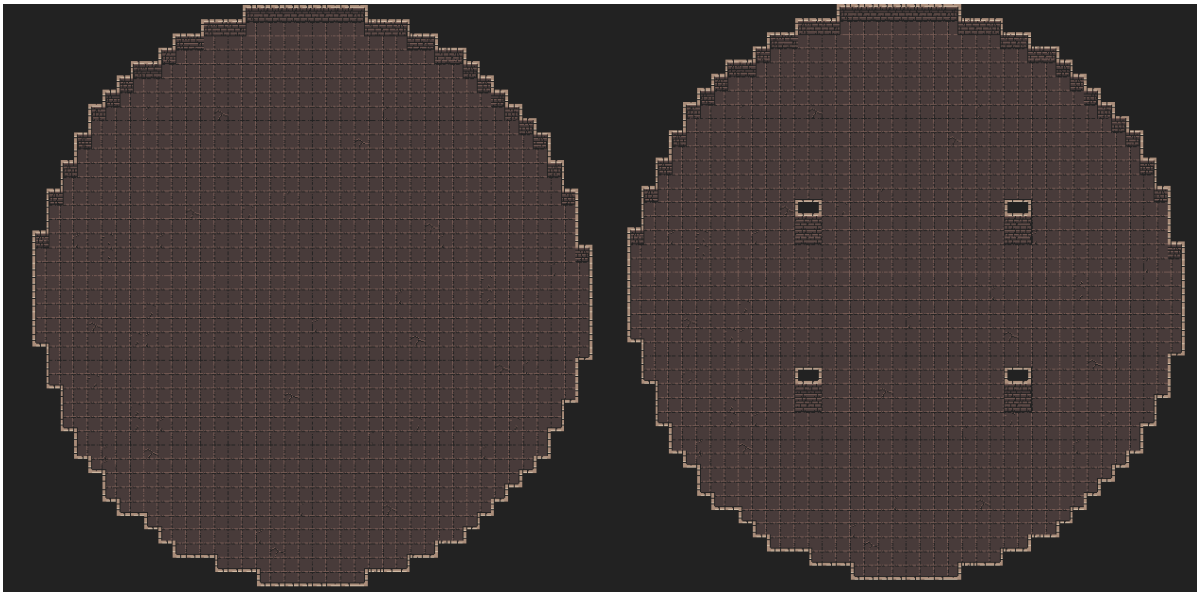

Figura 5.1.1.1: Espai sense obstacles

Figura 5.1.1.2: Espai amb columnes

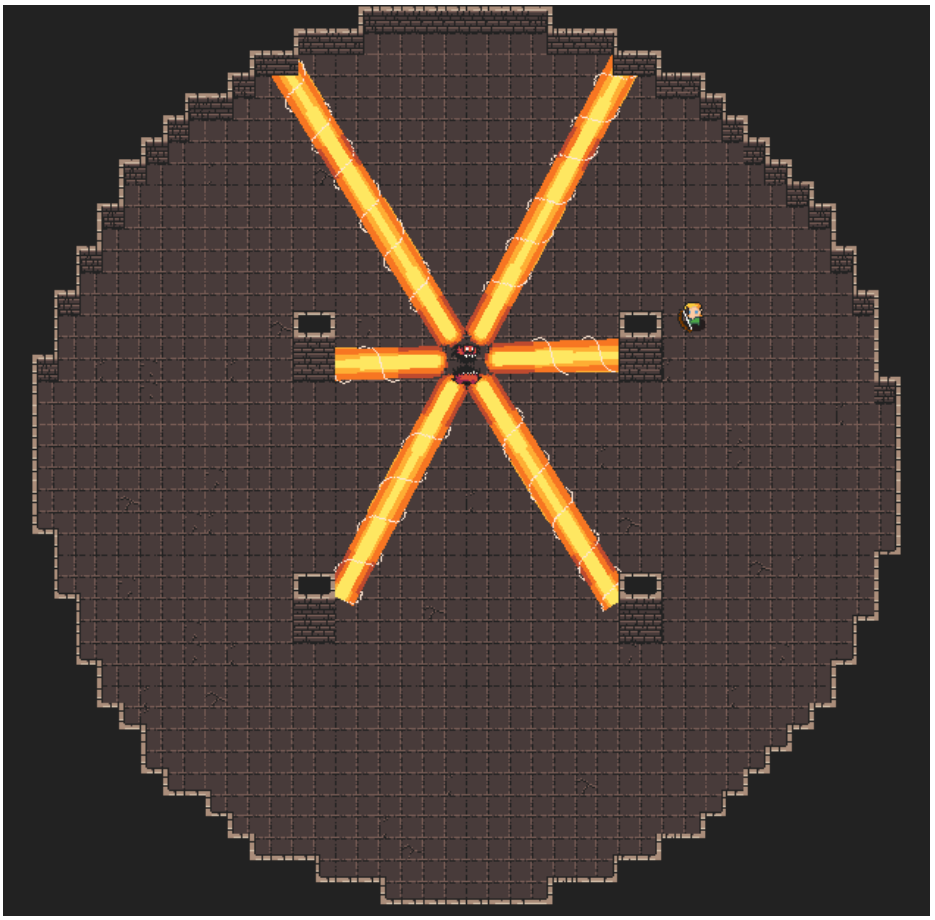

Figura 5.1.1.3: Jjugador utilitzant les columnes per cobrirse

5.1.2 Mecàniques del joc, reptes i accions

Com que volíem centrar la dificultat del joc en el comportament dels bosses, vam decidir crear un sistema de combat senzill i fàcil d'entendre amb poques mecàniques per a què el jugador les pugui aprendre ràpidament i així centrar tota l'atenció al comportament dels bosses i pensar estratègies per derrotar-los.

La mecànica principal és la de disparar amb l'arc. Funciona com s'esperaria d'un arc, com més es carrega el tir, més ràpida surt la fletxa i més mal fa. Això ho combinem amb un dash que el jugador pot utilitzar per desplaçar-se més ràpidament i així esquivar els atacs dels enemics.

5.1.3 Objectes, recursos i interaccions

Des que vam pensar la idea principal del joc, sabíem que no volíem objectes, ja que com hem comentat al punt anterior, no volem sobrecarregar al jugador d'informació ni mecàniques innecessàries, sinó simplificar tot al màxim i que el jugador es pugui centrar únicament en el comportament dels bosses.

5.1.4 Economia

Seguint amb l'objectiu de fer un joc senzill, no hi ha economia interna dins del joc, ja que el jugador comença el joc amb tots els recursos necessaris per completar-lo.

5.2 Disseny de personatges

5.2.1 Estil

L'estil artístic utilitzat es caracteritza principalment per:

- La vista "top down" (vista cenital des d'adalt), la qual vam escollir perquè permet mostrar tot el terreny de joc de forma molt clara, aspecte molt important als videojocs d'acció.

- Gràfics píxel art 2D (inspirats en Enter the Gungeon, veure figura 5.2.1.1). Vam escollir l'estil píxel art principalment per preferència personal, però també perquè, al ser un estil simple, dona molta claredat al jugador, de manera que li serà més fàcil identificar els diferents elements del joc.

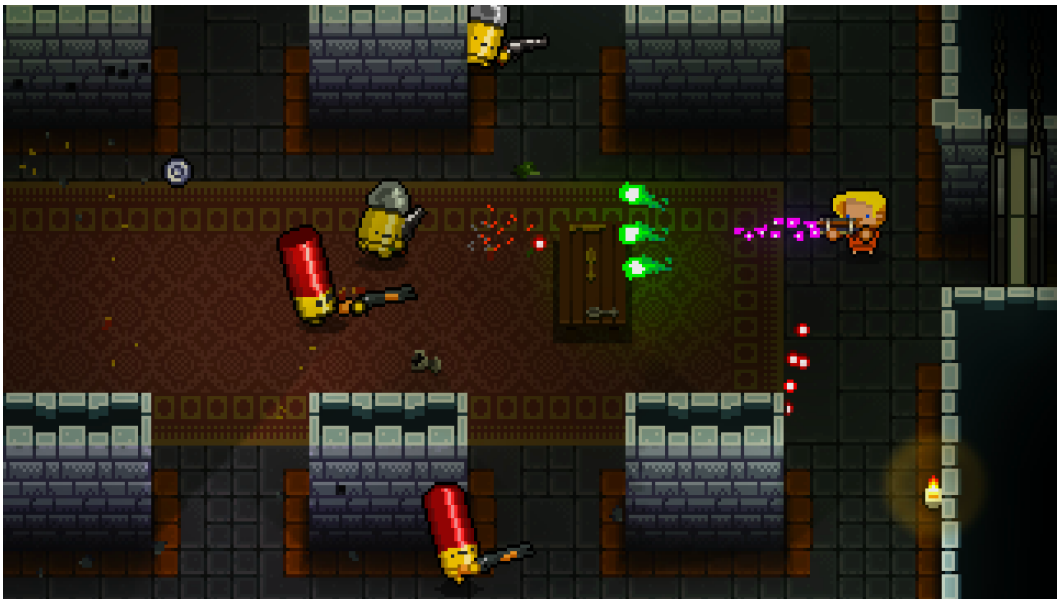

Figura 5.2.1.1: Captura del joc Enter the Gungeon

5.2.2 Assets utilitzats

Com que aquest projecte està orientat més a la part tècnica que no pas la artística, hem optat per utilitzar assets gratuïts. Concretament, hem utilitzat el pack "Dungeon Tileset II" de itch.io, ja que s'adaptava molt bé a la visió que teníem del projecte. Tot i això, sí que hem fet algunes petites animacions i efectes de partícules que necessitàvem per donar un millor feedback al jugador. Aprofundirem més en aquest aspecte als apartats 5.2.3, 5.2.4 i 5.6.

5.2.3 Personatge principal

El personatge principal és un elf amb un arc. Els sprites de l'elf són del pack mencionat al punt 5.2.2, mentre que les animacions i sprites de l'arc les hem fet nosaltres. Veure figures 5.2.3.1 i 5.2.3.2.

Figura 5.2.3.1: Personatge principal

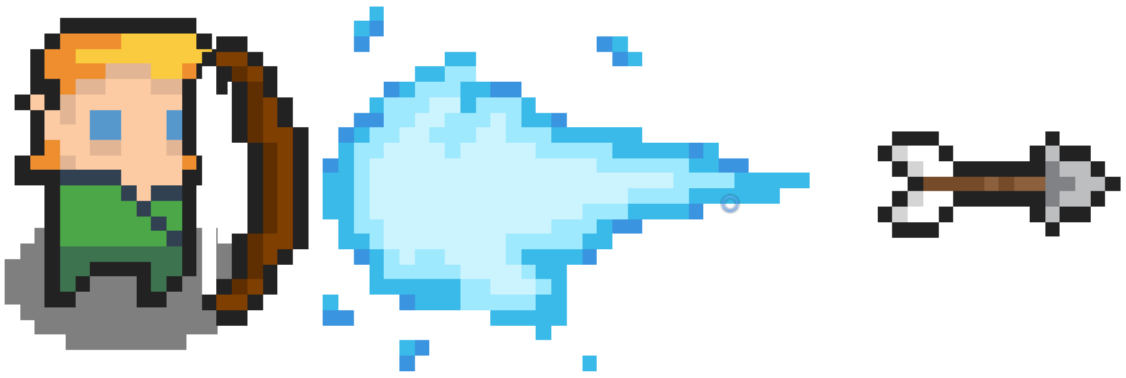

Figura 5.2.3.2: Animació dispar carregat al màxim.

5.2.4 Els bosses

Ogre

Figura 5.2.4.1: Sprite Ogre

El primer boss és un Ogre (veure Figura 5.2.4.1) amb un comportament molt agressiu. Es centra única i exclusivament en utilitzar atacs a curta distància, cosa facilita un poc el combat, ja que els atacs del personatge principal son a distancia, i deixa al jugador espai per aprendre els controls i les mecàniques del joc durant el combat. El comportament d'aquest boss es divideix en dues fases, una primera fase que, al ser el primer boss, busca ensenyar les mecàniques del joc, i una segona que posa a prova el que ha après el jugador pujant la dificultat i afegint un nou atac.

Atacs primera fase:

1. Cop cos a cos amb el bastó. Aquest atac té l'objectiu de motivar al jugador a mantenir la distància per aprofitar al màxim l'arc.
2. Intentar saltar sobre el jugador. Aquest atac està pensat per ensenyar al jugador a utilitzar la mecanica d'esquivar, ja que el boss salta i intenta caure sobre el jugador, que si està a curta-mitja distància, es veu forçat a esquivar.

Atacs segona fase:

1. Cop cos a cos amb el bastó. Aquest atac es manté de la fase anterior, pero és més ràpid.
2. Intentar saltar sobre el jugador. Aquest atac es manté de la fase anterior, pero és més ràpid i salta més lluny.
3. Dash cap al jugador + cop amb el bastó.

Dimoni

Figura 5.2.4.2: Sprite Dimoni

El segon boss és un Dimoni. A diferencia del boss anterior, aquest és centra únicament en utilitzar atacs a distància. Veure figura 5.2.4.2.

El comportament d'aquest boss es divideix en dues fases.

Atacs primera fase:

1. Disparar 2 onades de 4 projectils.
2. Laser cap al jugador.
3. Bomba.

Atacs segona fase:

1. Disparar 3 onades de 4 projectils.
2. Laser cap al jugador.
3. 3 bombes.
4. 6 lasers en totes direccions.

Nigromant

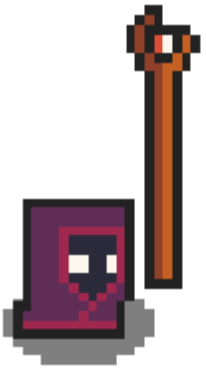

Figura 5.2.4.3: Sprite Nigromant

El tercer boss és un Nigromant. Aquest boss pot invocar súbdits, i els utilitza per a lluitar contra el jugador. Pot invocar esquelets, xamans i cavallers. Veure Figures 5.2.4.3 i 5.2.4.4.

Figura 5.2.4.4: Sprites invocacions del nigromant

El comportament d'aquest boss es divideix en tres fases a les quals invoca diferents súbdits. El Nigromant no pot rebre mal mentre els súbdits estan vius.

Primera fase. Invoca 5 esquelets, són els enemics més febles que pot invocar el nigromant, persegueixen al jugador i intenten fer-li mal per contacte.

Segona fase. Invoca 4 xamans, aquests enemics ataquen al jugador a distància amb boles de foc constants.

Tercera fase. Invoca 3 cavallers, aquests enemics son els que més vida tenen i ataquen cos a cos. A més, durant aquesta fase, el Nigromant comença a utilitzar el bastó per llançar atacs a distància al jugador mentre els cavallers ataquen.

5.3 Narrativa

A causa de l'enfocament més tècnic del projecte, hem utilitzat la narrativa principalment per donar un sentit general i una ambientació al joc, sense arribar a aprofundir-hi molt. La ambientació del joc està basada en una epoca medieval fantàstica, on habiten diferents races com per exemple el personatge principal, que és un elf.

Hem escollit aquesta ambientació per tenir gran llibertat a l'hora de crear els bosses i els atacs.

5.4 Interfícies

Al ser un videojoc d'acció, vam voler mantenir la pantalla lo més neta possible per a que el jugador pugui veure clarament els atacs dels enemics per tant, les úniques interfícies que el jugador pot veure mentres juga són:

- Els seus punts de vida, representats amb cors a dalt a l'esquerra, on cada cor son 2 punts de vida (6 punts de vida en total).

- La vida del boss, representada amb una barra vermella a baix.

Veure Figura 5.4.1.

Figura 5.4.1: Captura del joc

Per als textos dels menús hem escollit una tipografia amb estil pixel art, que encaixés bé amb l'estil artístic del joc. Concretament hem utilitzat la font "Dogica". Veure Figura 5.4.2.

Figura 5.4.2: Mostra de la font "Dogica"

5.5 Game layout chart

L'estructura del joc la mostrem a la Figura 5.5.1, es tracta d'una estructura totalment lineal on el jugador pasa de nivell al matar un boss i quan mor torna a començar el combat contra el mateix boss que l'ha matat. Vam decidir fer-ho així ja que, un cop el jugador ha superat el repte de derrotar a un boss, no té sentit obligar a que torni a lluitar contra ell.

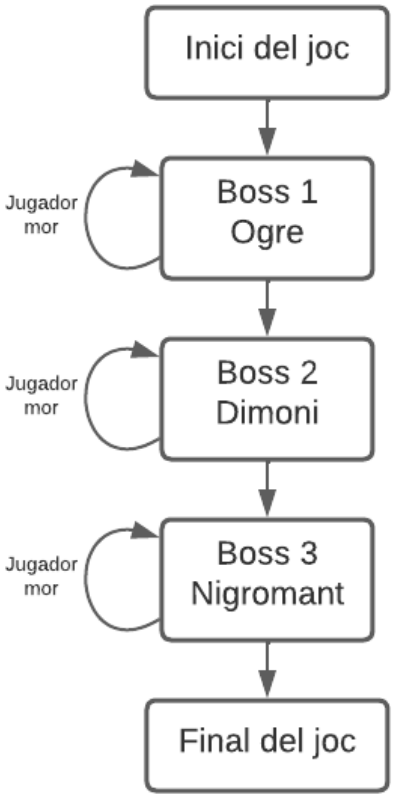

Figura 5.5.1: Game layout chart

5.6 VFX

5.6.1 Explosions

Impacte de projectils, que s'il·lustra a la Figura 5.6.1.1.

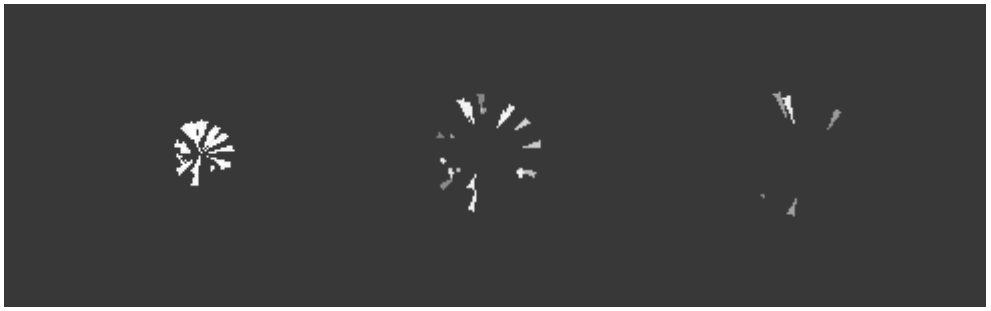

Figura 5.6.1.1: Impacte de projectils

Explosió granada, que podem veure a la Figura 5.6.1.2.

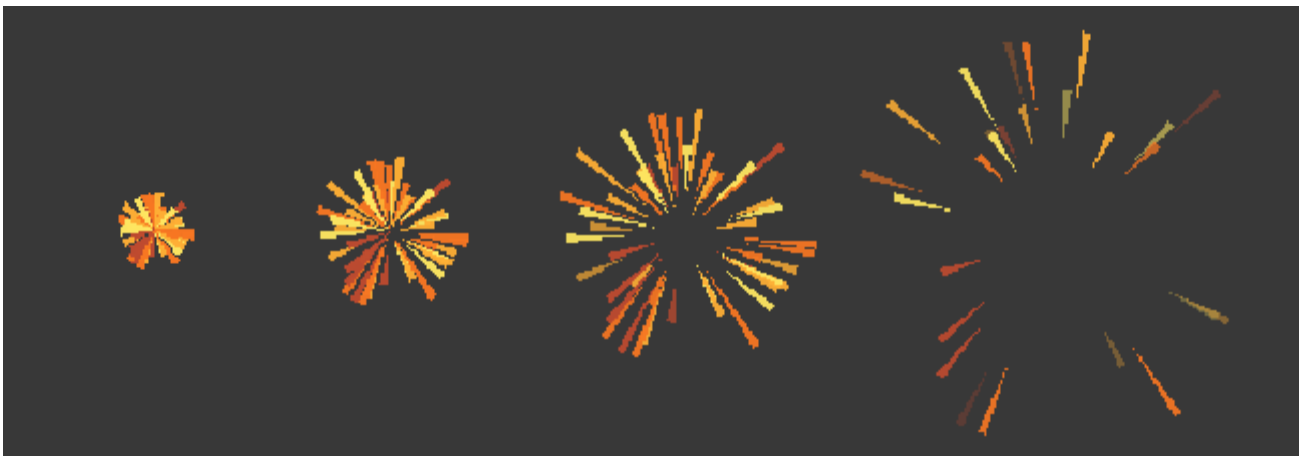

Figura 5.6.1.2: Impacte de projectils

Ambdues explosions han estat creades amb el unity particle system.

5.6.2 Efectes especials de bosses

Trencar el terra.
Alguns dels bosses tenen atacs potents amb els que poden arribar a trencar el terra. Per això vam crear aquesta animació amb els mateixos colors que el terra per poder superposar-la en qualsevol moment. Veure Figures 5.6.2.1 i 5.6.2.2.

Figura 5.6.2.1: Animació trencar el terra

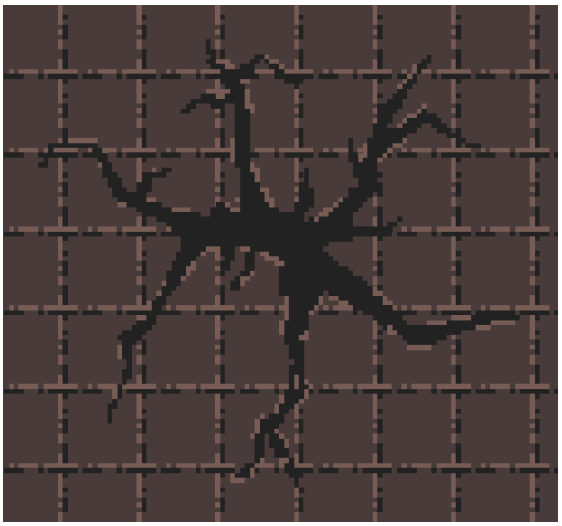

Figura 5.6.2.2: Mostra d'ús de la animació

Escut màgic.

Per mostrar al jugador clarament que no pot fer mal al Nigromante fins que no mata a tots els súbdits, hem fet una animació d'una "connexió magica" entre els súbdits i el Nigromante que dona un escut a aquest últim. Veure Figura 5.6.3.1.

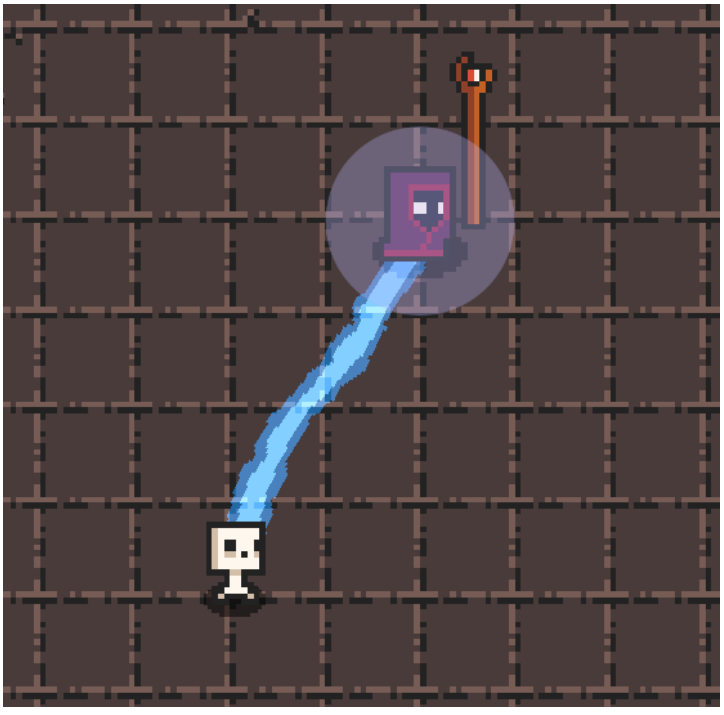

Figura 5.6.3.1: Escut màgic

Laser.
Un dels atacs del Dimoni. El podem veure a la Figura 5.6.4.1.

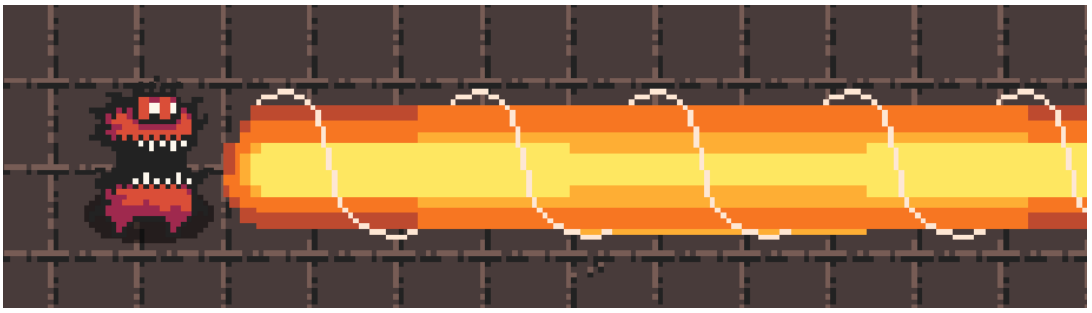

Figura 5.6.4.1: Laser

6. Implementació i proves

6.1 Implementació dels nivells

6.1.1 Estructura de les escenes

El joc està format per les 5 escenes mostrades a la Figura 6.1.1.1.

La escena “MainMenu” correspon al menú principal, i es la primera cosa que veu el jugador a l’obrir el joc.

Les escenes “OgreBoss”, “DemonBoss” i “NecromancerBoss” son les escenes corresponents a les diferents bosses del joc, col·locades en ordre cronològic.

Finalment la escena “EndScreen” es la pantalla que es mostra al completar el joc.

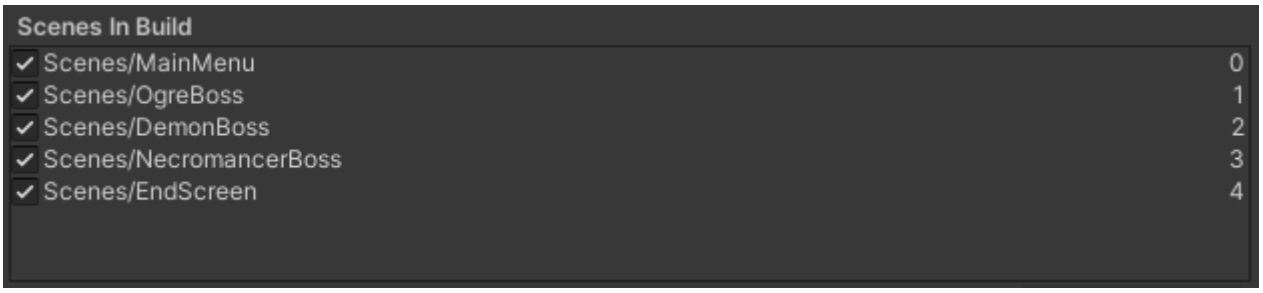

Figura 6.1.1.1: Llistat d’escenes

6.1.2 Creació de nivells

Per la creació dels nivells hem fet servir els components Tilemap i TilemapRenderer, dividint tots els elements del nivell en tres grups, cadascun amb aquests dos components i un component “NavMeshModifier” per definir per on poden caminar els personatges, com es pot veure a la Figura 6.1.2.1.

El grup de “Floor” es per tots els tiles corresponents al terra i que no han de colisionar amb res, simplement es un fons fet amb tiles.

El grups de “WallsTop” i “WallsBottom” son per tots els obstacle, i a més dels components mencionats anteriorment, també tenen els components Rigidbody 2D, Tilemap Collider 2D i Composite Collider per gestionar tot el tema de les collisions. Hem dividit les parets en dos grups per poder diferenciar quines s’han de mostrar per sobre dels personatges i quines s’han de mostrar per sota. Això es veu clarament a la Figura 6.1.2.2.

Figura 6.1.2.1: Estructura tilemaps

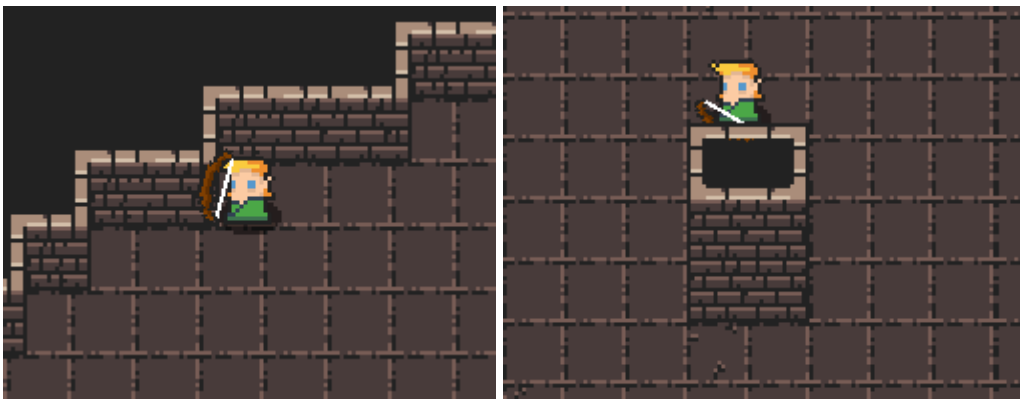

Figura 6.1.2.2: Exemple mostrant orde de capes

6.2 Implementació de la càmera

Per a la camara hem fet servir el package Cinemachine de Unity, amb el que hem creat un component CinemachineVirtualCamera que hem associat a la càmera principal de la escena afegint un component CinemachineBrain. Un cop fet això, simplement hem assignat el component Player al follow de la camera virtual per a que la càmera segueixi al jugador. A més hem modificat alguns dels paràmetres de follow de la camera virtual per fer que el moviment de la camera sigui més suau i per definir una petita "zona morta" al voltant del jugador i així només moure la camera quan el jugador surt d'aquella zona, com podem veure a la Figura 6.2.1.

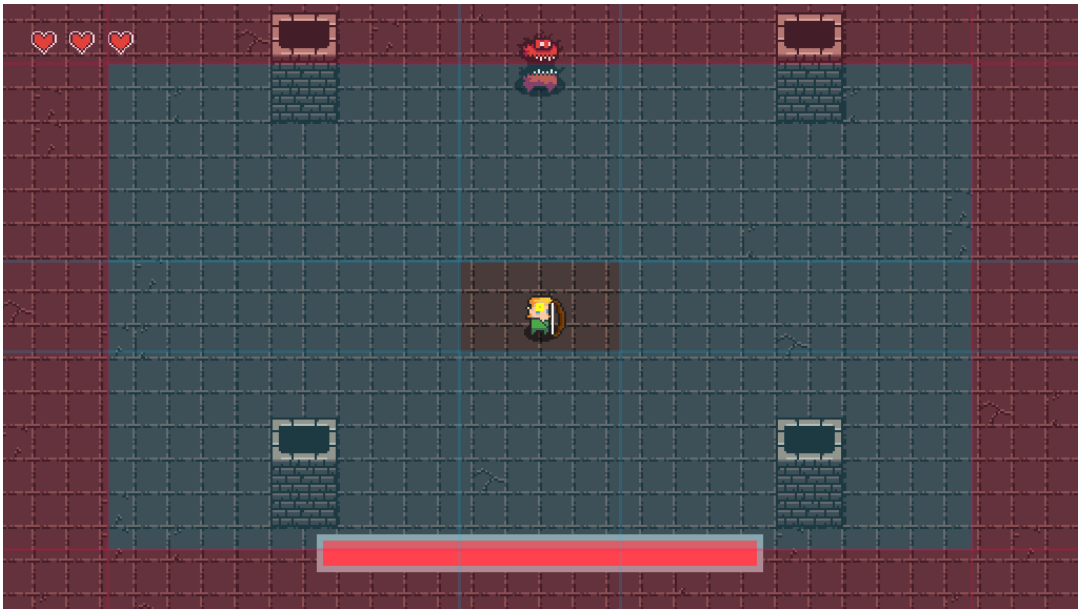

Figura 6.2.1: Deadzone de la càmera

També hem afegit un component Cinemachine Confiner 2D a la camera virtual per evitar que la cámara surti de el mapa.

6.3 Implementació de les interfícies

6.3.1 PlayerHealth

Per la vida del jugador hem afegit 3 imatges al canvas, una per cada cor que té el jugador (veure Figura 6.3.1.1).

Figura 6.3.1.1: Vida del personatge

Per posar les imatges correctes per mostrar la vida del personatge, hem creat un script "PlayerHealth".

Aquest script té una funció "takeHit()" (veure Figura 6.3.1.2) que es crida sempre que el jugador rep un impacte. La funció actualitza la vida actual, modifica les images dels cors i si la vida és 0 crida la funció "manageDeath()", que gestiona la mort del personatge. A més quan es crida la funció "takeHit()" s'emet un event "OnHit()", al que altres scripts es poden subscriure si és necessari, aprofundirem més en aquest tema a l'Apartat 6.6.

```
4 referencias
public void takeHit()
{
    if (mainCharacterController.isDashing) return;
    OnHit();
    currentHP -= 1;
    for (int i = 0; i < heartImages.Length; i++) {
        if(i*2 < currentHP) {
            if(i*2 + 1 < currentHP) {
                heartImages[i].sprite = fullHeart;
            }
            else {
                heartImages[i].sprite = halfHeart;
            }

        }
        else {
            heartImages[i].sprite = emptyHeart;
        }
    }

    if (currentHP <= 0) manageDeath();
}
```

Figura 6.3.1.2: Funció takeHit()

6.3.2 Health Bar

Per la barra de vida dels bosses simplement hem afegit una imatge de tipus Filled al canvas, la qual té un paràmetre fill amount que igualem al percentatge de vida de l'enemic. Veure Figura 6.3.2.1.

Figura 6.3.2.1: Barra de vida dels bosses

6.3.3 Menú principal

Escena amb un fons, el títol del joc i dos botons, un per començar la partida i un altre per tancar el joc. Veure Figura 6.3.3.1.

Figura 6.3.3.1: Menú principal

6.3.4 Pantalla final

Escena amb un fons, un text i dos botons, un per tornar al menú principal i un altre per tancar el joc. Veure Figura 6.3.4.1.

Figura 6.3.5.1: Pantalla final

6.4 Implementació del jugador

6.4.1 Input

Per als inputs voliem permetre al jugador poder jugar tant amb teclat i ratolí, com amb un comandament. Per tant vam decidir utilitzar el nou input system de unity, que permet crear diferents accions per als jugadors i canviar dinámicamente entre comandament i teclat i ratolí en funció de que esta utilitzant el jugador. Veure Figura 6.4.1.1.

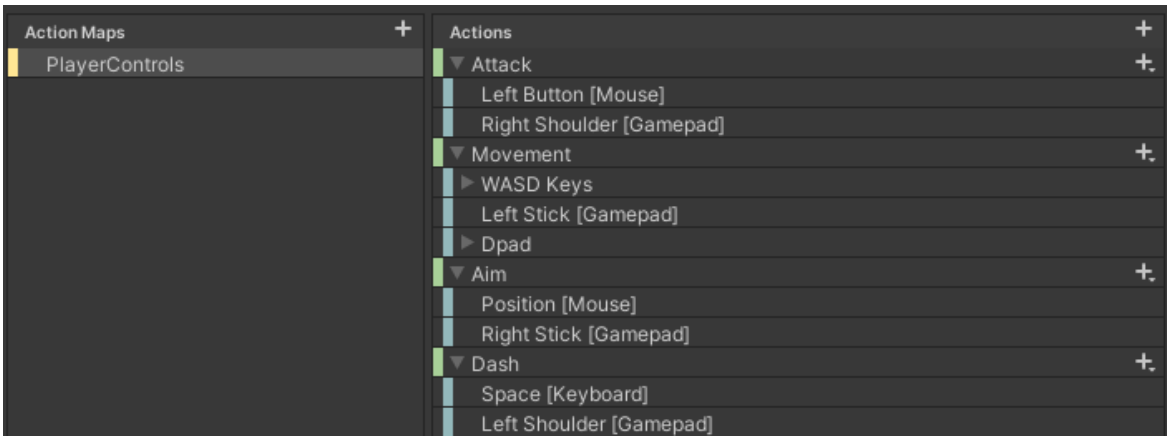

Figura 6.4.1.1: Accions del jugador + botons associats

Un cop definides les accions, creem funcions públiques que podem associar als events que s'executen quan el jugador fa una acció.

6.4.2 Moviment

Per al moviment tenim 2 mecàniques:

- La primera és la de **caminar/correr**. Per aquesta mecànica hem assignat al personatge un Rigidbody2D dinàmic, per a que s'encarregui de totes les col·lisions. Per moure aquest rigidbody agafem el Vector2 que ve donat per la acció "Movement", el normalitzem per obtenir el vector de moviment, el multipliquem per la constant velocitat i assignem aquesta velocitat a la velocitat del rigidbody. Veure Figura 6.4.2.1.

- La segona mecànica és la del **dash**. Aquesta mecànica funciona de forma similar a la anterior pero afegim un input extra, que és el botó que ha de pulsar el jugador per fer el dash. Només es pot fer cada cert temps (és a dir, no es pot fer un nou dash immediatament després d'acabar un dash) i un cop començat el dash, aquest va en la mateixa direcció fins que acaba el dash. Veure Figura 6.4.2.1.

```
1 referencia
private void Move()
{
    // Walk
    if (!isDashing)
    {
        rb.velocity = inputMovement * spd;
        if(inputMovement != Vector2.zero) lastInputMovement = inputMovement;
    }
    //Dash
    else if (isDashing) rb.velocity = lastInputMovement * dashSpd;
}

1 referencia
private void Dash()
{
    if (Time.time > lastDashTime + dashDuration) isDashing = false;
    if (Time.time > lastDashTime + dashCooldown) canDash = true;
    if (dashPressed && canDash)
    {
        dashPressed = false;
        isDashing = true;
        canDash = false;
        lastDashTime = Time.time;
    }
}
```

Figura 6.4.2.1: Funcions moviment

6.4.3 Arc

L'arc és posem com a fill del player i conté dos elements: L'sprite i el punt on volem instanciar les fletxes. Veure Figura 6.4.3.1.

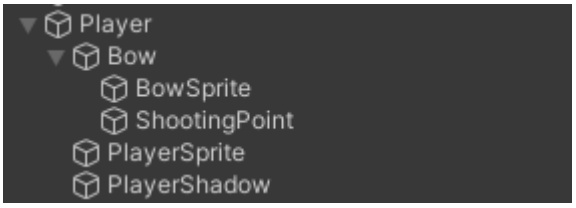

Figura 6.4.3.1: Objecte bow

El primer que fem és rotar l'sprite de l'arc per a que apunta cap a la posició del ratolí o cap a la direcció del joystick dret del comandament (depenent de que estigui utilitzant el jugador en aquell moment). Veure Figures 6.4.3.2 i 6.4.3.3.

```
//Input
1 referencia
private void AimInput()
{
    if (playerInput.currentControlScheme == "Gamepad")
    {
        aimDir = playerInput.actions["Aim"].ReadValue<Vector2>();
    }
    else if (playerInput.currentControlScheme == "Keyboard and Mouse")
    {
        mousePos = cam.ScreenToWorldPoint(playerInput.actions["Aim"].ReadValue<Vector2>());
        aimDir = mousePos - rb.position;
    }
}
```

Figura 6.4.3.2: Funció AimInput script MainCharacterController.cs

```
1 referencia
private void RotateWeapon()
{
    float angle = Mathf.Atan2(mainCharacterController.aimDir.y, mainCharacterController.aimDir.x) * Mathf.Rad2Deg;
    transform.rotation = Quaternion.AngleAxis(angle, Vector3.forward);
}
```

Figura 6.4.3.3: Funció RotateWeapon script Weapon.cs

Per al dispar primer llegim el moment en que el jugador prem el botó i quan el deixa de prémer instanciem el projectil en la direcció que està apuntant. Un cop instanciat, assignem diferents mal i velocitat al projectil en funció de el temps que el jugador ha mantingut el botó premut. Veure Figura 6.4.3.4.

```
private void Shoot()
{
    if (shootButtonJustReleased && Time.time > nextFire)
    {
        shootButtonJustReleased = false;
        nextFire = Time.time + shotsPerSec;

        //Instantiate projectile
        float angle = Mathf.Atan2(mainCharacterController.aimDir.y, mainCharacterController.aimDir.x) * Mathf.Rad2Deg - 90f;
        GameObject projectileInstance = Instantiate<GameObject>(projectile, shootingPoint.transform.position, Quaternion.Euler(new Vector3(0, 0, angle)));
        Projectile projectileScriptRef = projectileInstance.GetComponent<Projectile>();
        float timeCharged = Time.time - chargeShotTime;

        //Set DMG and Speed
        //Level 1 charge
        if (timeCharged >= maxChargeTime)
        {
            projectileScriptRef.spd = maxChargeSpd;
            projectileScriptRef.dmg = maxChargeDmg;
            animator.Play("BowShoot");
        }
        //Level 2 charge
        else if(timeCharged >= mediumChargeTime)
        {
            projectileScriptRef.spd = mediumProjectileSpd;
            projectileScriptRef.dmg = mediumProjectileDmg;
            animator.Play("BowNormalShot");
        }
        //Max charge
        else
        {
            projectileScriptRef.spd = minProjectileSpd;
            projectileScriptRef.dmg = minProjectileDmg;
            animator.Play("BowIdle");
        }
    }
}
```

Figura 6.4.3.4: Funció Shoot script Weapon.cs

6.4.4 Animacions

Per les animacions del personatge hem fet un script PlayerAnimation on fem que el jugador miri en la direcció cap a la que està apuntant en tot moment i canviem entre les animacions Idle i Run en funció del si el personatge s'està movent. Veure Figura 6.4.4.1.

```
void Update() {
    if (mainCharacterController.aimDir.x > 0 && !isFacingRight) Flip();
    else if (mainCharacterController.aimDir.x < 0 && isFacingRight) Flip();

    if (Mathf.Abs(rb.velocity.magnitude) <= 0.1f) {
        ChangeAnimationState(PLAYER_IDLE);
    }
    else {
        ChangeAnimationState(PLAYER_RUN);
    }
}
```

Figura 6.4.4.1: Update script PlayerAnimation

6.5 Implementació dels enemics

6.5.1 Bosses

El comportament dels bosses està fet amb Behaviour Trees. Per facilitar el procés de creació dels BehaviourTrees hem utilitzat un package gratuït fet per “TheKiwiCoder” que proporciona un entorn visual on construir els arbres, i els nodes bàsics pel funcionament del mateix.

Nodes del package que hem utilitzat:

- **Log**: imprimeix un missatge per consola.
- **Wait**: retorna Running durant un temps entrat per paràmetre, Success un cop ha passat el temps.
- **Selector**: executa els nodes fills d'esquerra a dreta fins que un retorna Success. Si no retorna Failure.
- **RandomSelector**: executa un node fill aleatori.
- **Sequencer**: executa els fills d'esquerra a dreta fins que un retorna Failure. Si no retorna Success.
- **Failure**: sempre retorna Failure independentment dels fills.
- **Repeat**: executa el fill en en bucle.

6.5.2 Nodes comuns

En aquest apartat explicarem el funcionament dels nodes creats per aquest projecte utilitzats per més d'un dels bosses, mentre que els nodes específics de cada boss i els Behaviour Trees que connecten tots els nodes s'explicaran als apartats 6.5.3, 6.5.4 i 6.5.5.

Nodes decoradors:

ExecuteIfPhase
Té un paràmetre fase, i només executa el node fill si la fase actual del boss és igual a la fase entrada al paràmetre, en cas contrari retorna Success.

FailIfPhase
Té un paràmetre fase, i només executa el node fill si la fase actual del boss és diferent a la fase entrada al paràmetre, en cas contrari retorna Failure.

RandomTimeout
Calcula un temps random entre dos paràmetres i executa el node fill fins que s'acaba el temps. Quan s'acaba el temps retorna Failure. Veure Figura 6.5.2.1.

```
protected override void OnStart() {
    startTime = Time.time;
    duration = Random.Range(minDuration, maxDuration);
}

45 referencias
protected override void OnStop() {
}

46 referencias
protected override State OnUpdate() {
    if (Time.time - startTime > duration) {
        return State.Failure;
    }

    return child.Update();
}
```

Figura 6.5.2.1: Node RandomTimeout

<u>Nodes d'acció</u>:

CheckCurrentPhase
Retorna Success si la fase actual és igual al paràmetre fase, Failure en cas contrari.

CheckHPBelow
Retorna Success si la vida actual és inferior al paràmetre entrat, Failure en cas contrari.

FacePlayer
Rota l'sprite de l'enemic perquè miri cap al personatge. Sempre retorna Success.

MoveTowardsPlayer
Mou l'enemic cap al personatge utilitzant l'AI agent de unity i un navmesh que determina per on es pot caminar. Retorna success si el path és vàlid, Failure si és invàlid.

NextPhase
Suma 1 a la fase actual. Sempre retorna success.

PlayAnimAndWait
Atura el moviment de l'enemic resetejant el path, i reprodueix l'animació amb el nom entrat. Mentre l'animació s'està reproduint, retorna Running i quan acaba de reproduir-se retorna Success. Veure Figura 6.5.2.2

```
public string animationName;
private bool playedAnimation = false;

44 referencias
protected override void OnStart() {
    context.agent.ResetPath();
    playedAnimation = false;
}

44 referencias
protected override void OnStop() {
}

45 referencias
protected override State OnUpdate() {
    if (!playedAnimation) {
        playedAnimation = true;
        context.animator.Play(animationName);
        return State.Running;
    }
    else if (context.animator.GetCurrentAnimatorStateInfo(0).IsName(animationName)
        && context.animator.GetCurrentAnimatorStateInfo(0).normalizedTime <= 1.0f) {
        return State.Running;
    }
    else {
        return State.Success;
    }
}
```

Figura 6.5.2.2: Node PlayAnimAndWait

RandomWait
Retorna Running durant un temps aleatori entre dos paràmetres, Success quan ja ha passat el temps.

6.5.3 Ogre

A continuació explicarem el Behaviour Tree de l'Ogre, així com els nodes específics desenvolupats. Veure Figura 6.5.3.1.

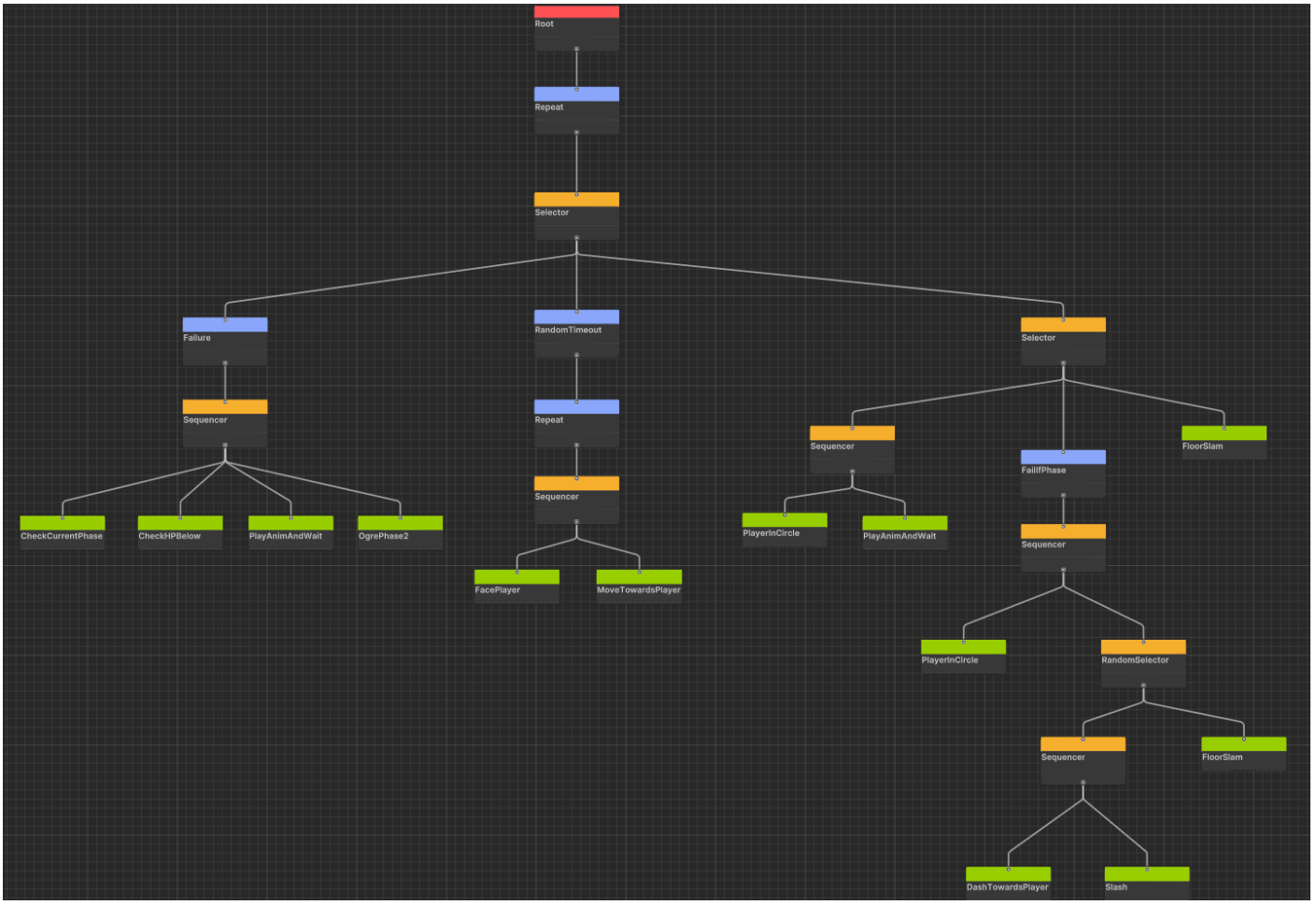

Figura 6.5.3.1: Behaviour Tree Ogre

Comencem amb un node Repeat per a que l'arbre no s'executi una sola vegada, sinó que ho faci en bucle, amb un node Selector a sota que divideix l'arbre en 3 branques.

La primera branca (veure Figura 6.5.3.2) utilitza un node Sequencer que comprova la fase actual amb el node CheckCurrentPhase i, si està en primera fase, comprova si la vida és inferior al 50% amb el node CheckHPBelow. Si es compleixen aquestes dues condicions, reprodueix l'animació de canvi de fase amb el node PlayAnimAndWait, i quan l'animació acaba executem el node OgrePhase2. Tot això amb un node Failure a dalt de tot per a què, independentment del resultat del Sequencer, s'executi la següent branca.

OgrePhase2
Aquest node canvia alguns atributs de l'ogre, com la velocitat de moviment i la velocitat d'algunes animacions per fer els atacs més difícils.

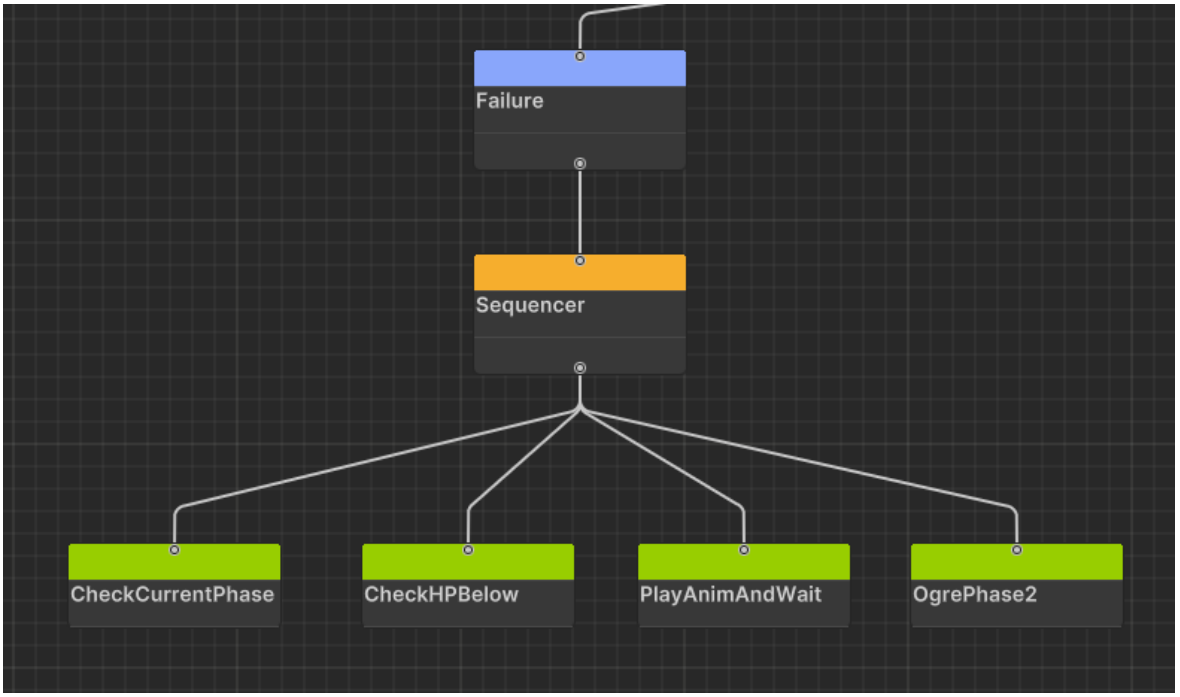

Figura 6.5.3.2: Primera branca Behaviour Tree Ogre

La segona branca (veure Figura 6.5.3.3) s'encarrega de fer que l'enemic miri al jugador i camini cap a ell (nodes FacePlayer i MoveTowardsPlayer) durant una estona aleatòria entre 2 valors (node RandomFailure). Aquesta branca ens serveix per donar temps de respirar al jugador entre atac i atac.

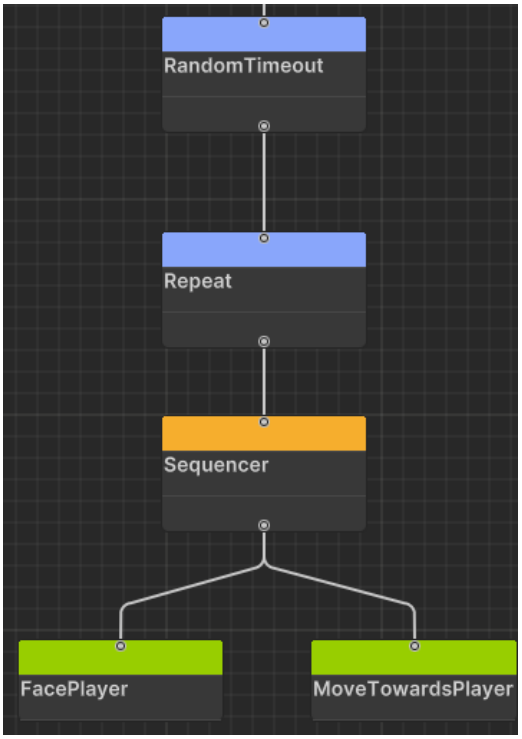

Figura 6.5.3.3: Segona branca Behaviour Tree Ogre

Finalment, la tercera branca (veure Figura 6.5.3.4), s'encarrega de escollir quin atac farà l'enemic en cada moment. Aquesta branca és divideix en tres sub-branques. La primera comprova si el jugador està dintre de l'area d'atac a melee amb el node PlayerInCircle, i si esta dintre de l'àrea executa el node PlayAnimAndWait que fa l'atac a melee. Si el jugador no està dintre del cercle, passem a la segona sub-branca.

La segona sub-branca només s'executa si el boss està en segona fase (node FaillfPhase) i a una distancia determinada del jugador (node PlayerInCircle). Si es compleixen aquestes 2 condicions executa un atac aleatori entre:

- Dash + cop cos a cos (nodes DashTowardsPlayer + Slash)
- Saltar sobre el personatge (node FloorSlam)

Finalment si arriba a la tercera subbranca simplement executa el node FloorSlam.

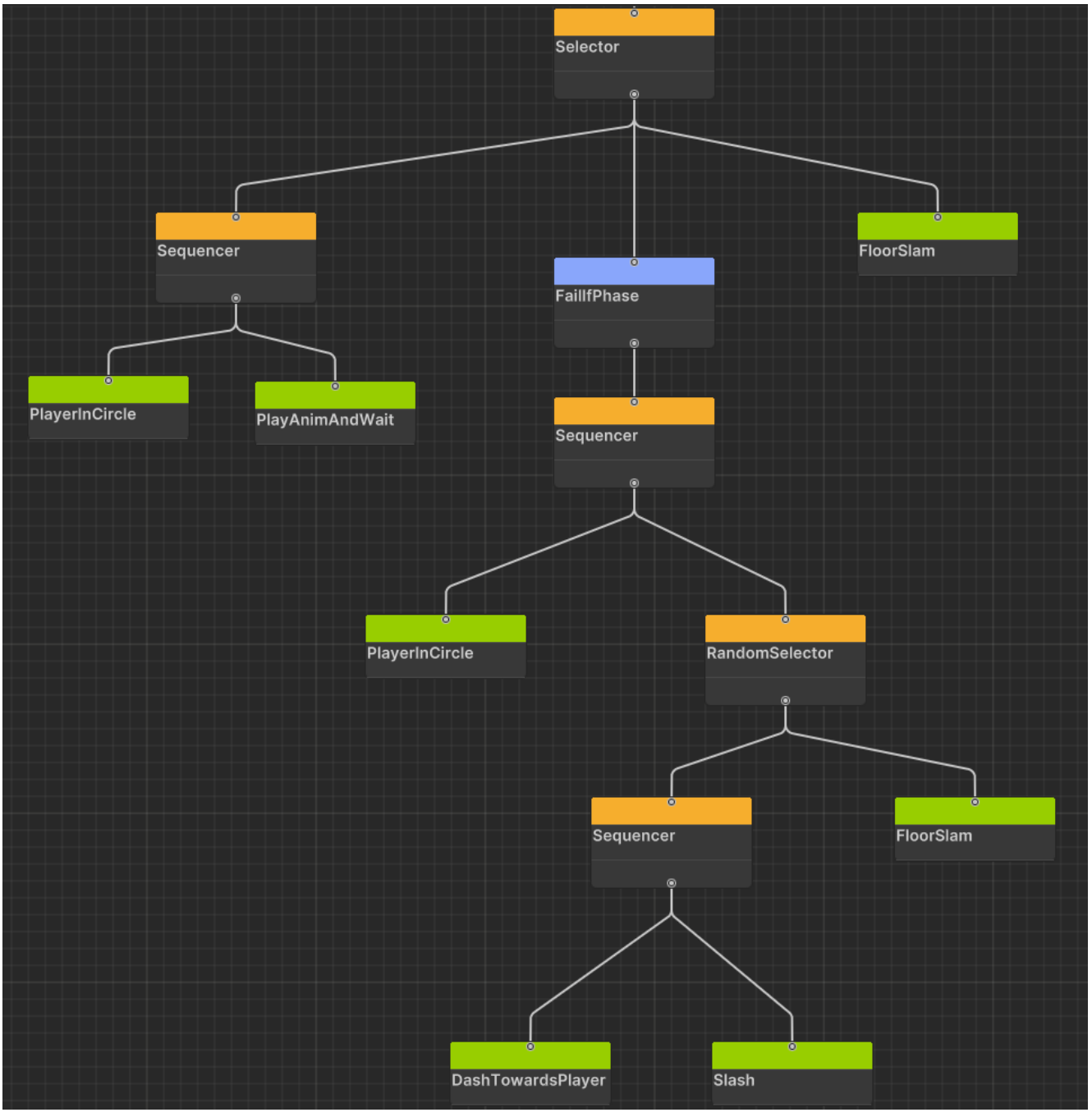

Figura 6.5.3.4: Tercera branca Behaviour Tree Ogre

PlayerInCircle

Comprova si el jugador està dintre d'una area determinada. Retorna Success si el jugador està dintre del cercle, Failure en cas contrari. Veure Figura 6.5.3.5.

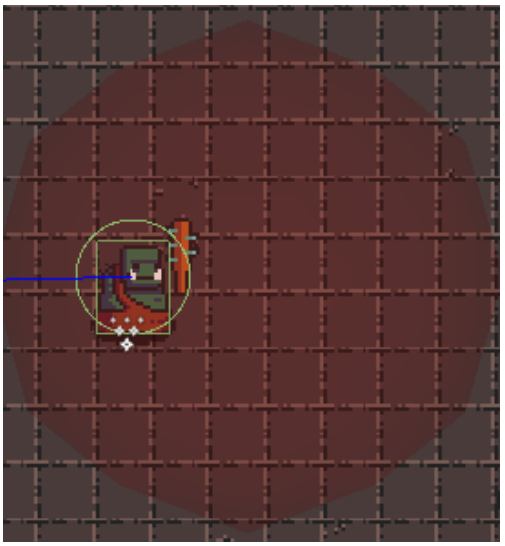

Figura 6.5.3.5: Àrea PlayerInCircle

DashTowardsPlayer

Fa un dash des de la seva posició inicial fins a la posició del jugador al moment de començar el dash. Retorna Failure si el path és invàlid, Running mentre fa el dash i Success quan arriba a la posició final. També rota l'arma de l'enemic per a que miri cap al jugador. Veure Figura 6.5.3.6.

```
protected override State OnUpdate() {

    RotateWeapon();
    if (context.agent.pathPending) {
        return State.Running;
    }
    if (context.agent.pathStatus == UnityEngine.AI.NavMeshPathStatus.PathInvalid) {
        context.customBlackboard.dust.Stop();
        return State.Failure;
    }
    if (context.agent.remainingDistance < tolerance) {
        context.customBlackboard.dust.Stop();
        context.agent.ResetPath();
        return State.Success;
    }
    return State.Running;
}

2 referencias
private void RotateWeapon() {
    dashDir = context.customBlackboard.playerPosition.position - context.transform.position;
    dashDir = dashDir.normalized;
    float angle = Mathf.Atan2(dashDir.y, dashDir.x) * Mathf.Rad2Deg;
    context.customBlackboard.weaponRotationCenter.rotation = Quaternion.AngleAxis(angle, Vector3.forward);
}
```

Figura 6.5.3.6: Codi DashTowardsPlayer

Slash

Reprodueix l'animació de l'atac Slash. Retorna Running mentre s'executa i quan acaba reseteja la rotació de l'arma (modificada al node DashTowardsPlayer) i retorna Success.

FloorSlam
Reprodueix l'animació de salt, canvia la velocitat de l'enemic a la velocitat de salt. Retorna Running mentre s'executa i, quan acaba l'animació, reseteja la velocitat de l'enemic a la velocitat base i retorna Success.

6.5.4 Dimoni

A continuació explicarem el Behaviour Tree del Dimoni, així com els nodes específics desenvolupats. Veure Figura 6.5.4.1.

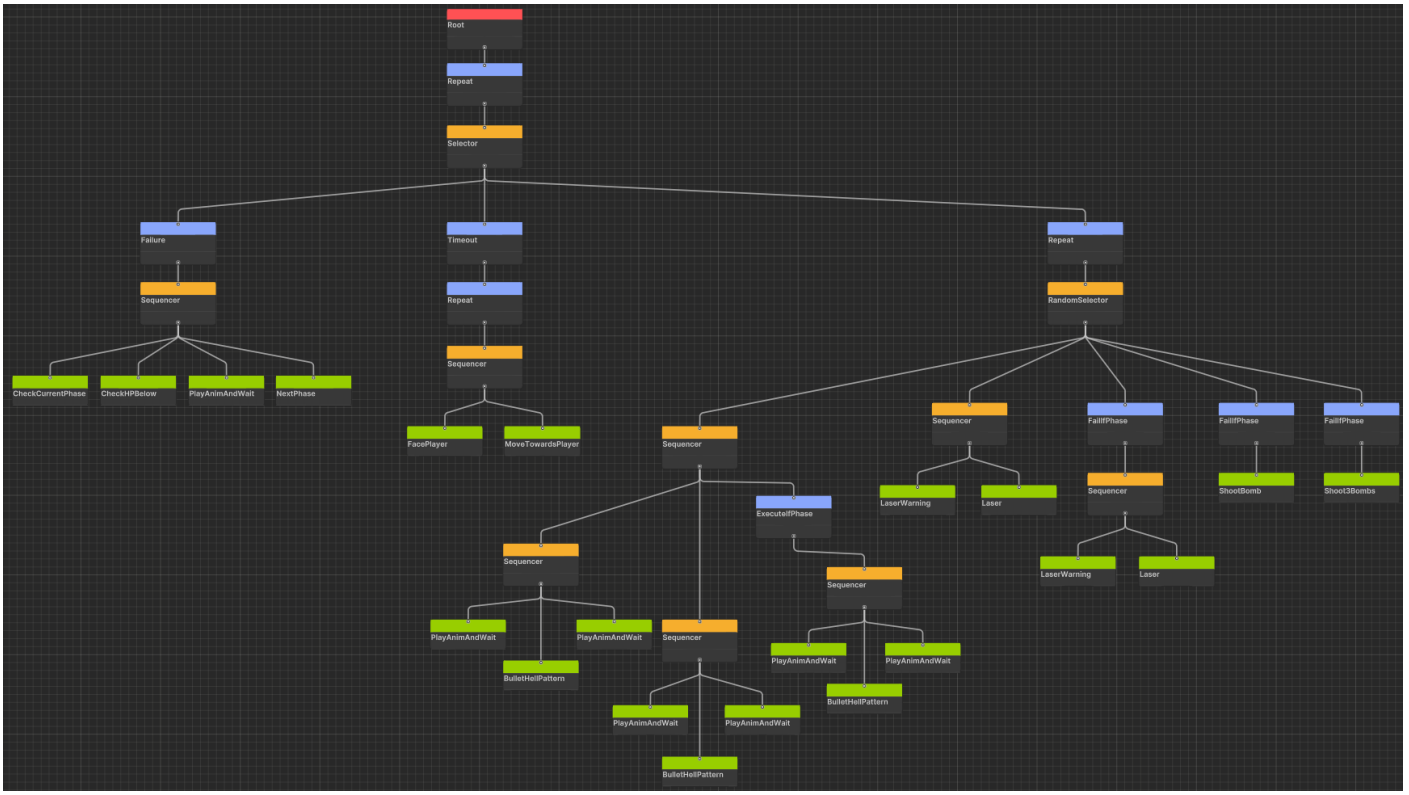

Figura 6.5.4.1: Behaviour Tree Dimoni

Les dos primeres branques d'aquest boss son molt similars a les comentades a l'apartat anterior. La única diferencia en temes de nodes es troba a la primera branca, on canviem el node OgrePhase2, que era específic per l'ogre, pel node NextPhase, ja que a aquest boss no necessitem canviar atributs al canviar de fase, simplement sumar 1 a la fase actual. Per altra banda, al node MoveTowardsPlayer de la segona branca li hem pujat la distancia a la que l'enemic deixa de seguir al personatge, ja que al ser un enemic a distancia no volem que s'apropi tant al jugador.

La tercera branca gestiona els atacs. Utilitzem un Random Selector que separa els 5 atacs posibles combinat amb un repeat (configurat per a que només repeteixi on Failure) que repeteix la selecció random en cas de que l'atac escollit no sigui vàlid a la fase actual.

El primer atac està construït combinant Sequencers amb els nodes PlayAnimAndWait i BulletHellPattern, de manera que s'executa Animació -> Disparar projectils -> Animació 2 cops (3 si el boss està en segona fase). Veure Figura 6.5.4.2.

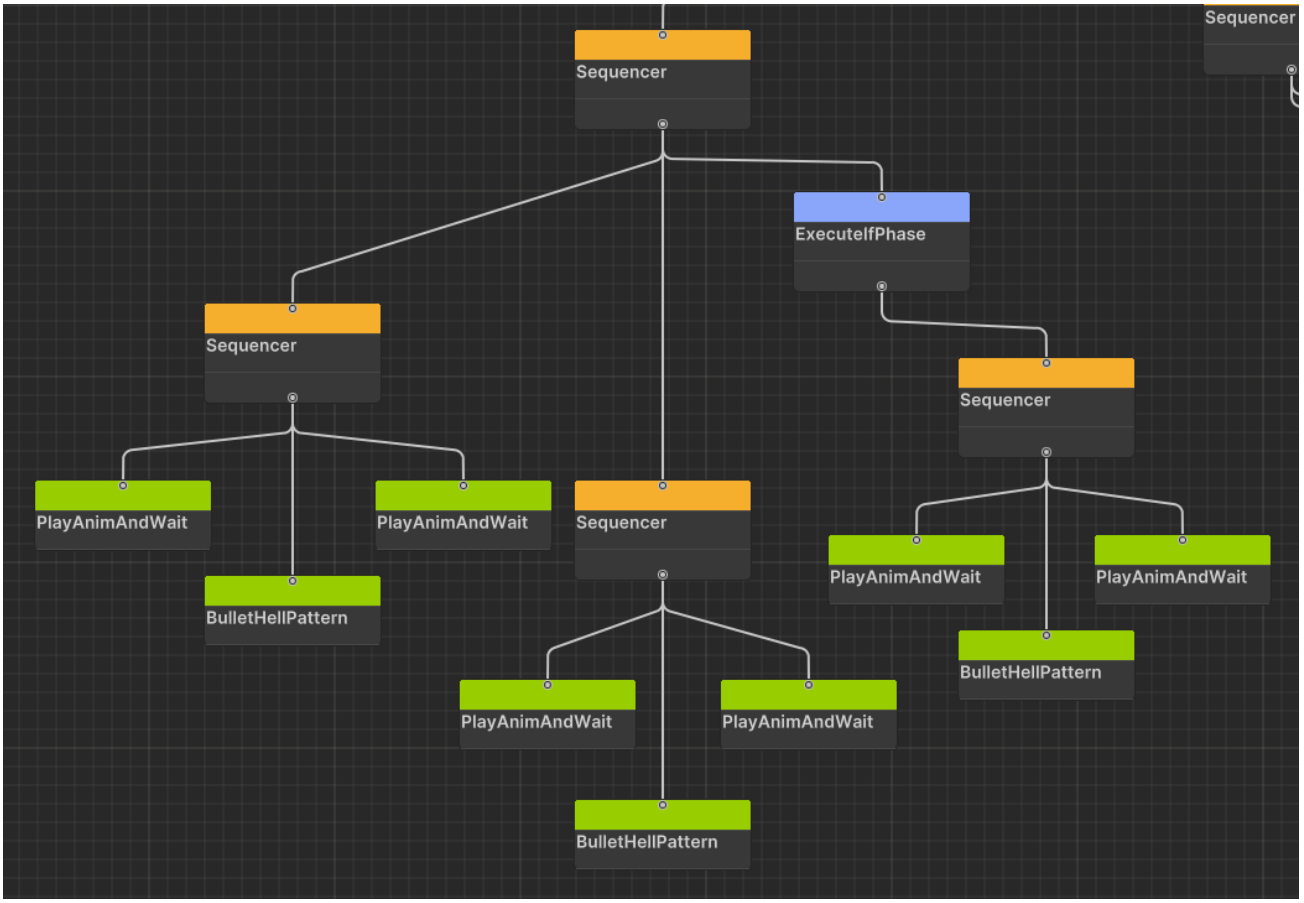

Figura 6.5.4.2: Arbre primer atac

BulletHellPattern

Aquest node ens permet disparar projectils amb patrons tipus bullet hell (veure Figura 6.5.4.3) entrant el nombre de projectils que volem disparar, l'angle inicial del primer projectil i un prefab del projectil. Sempre retorna Success

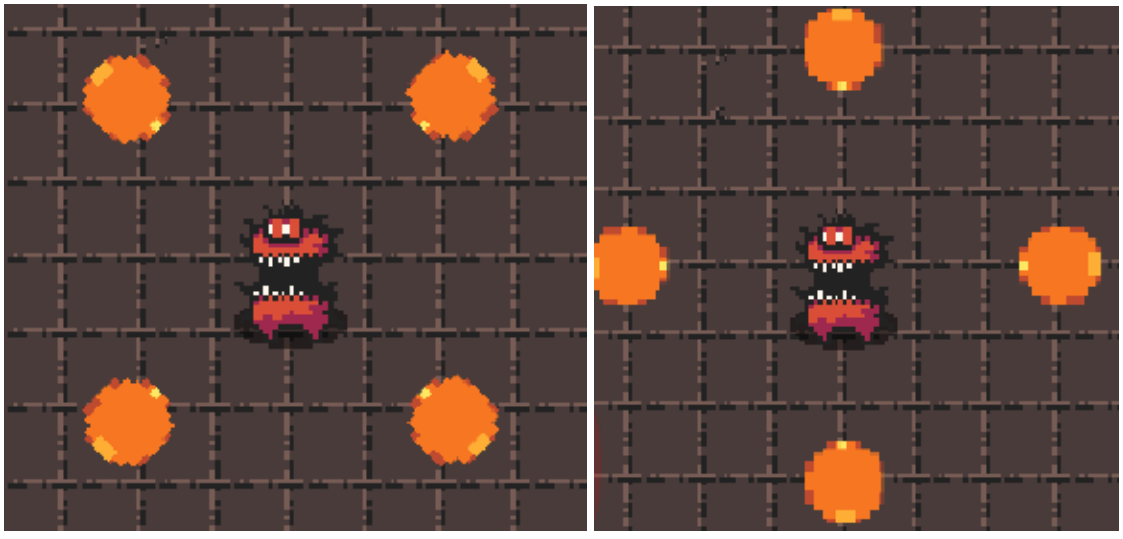

Figura 6.5.4.3: Primer atac

El segon atac mostra primer un làser vermell semi transparent (node LaserWarning) que apunta cap al jugador per avisar de que a continuació ve un làser, i després d'un temps determinat dispara el làser (node Laser). Veure Figura 6.5.4.4

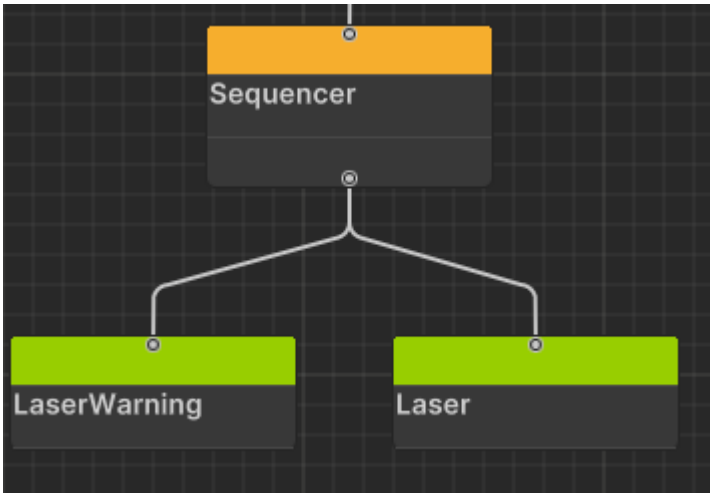

Figura 6.5.4.4: Segon atac

LaserWarning i Laser

Expliquem aquests 2 nodes junts perquè funcionen de forma bastant similar. Primer podem especificar el nombre de lasers que volem disparar i si volem que el primer láser apunti cap al jugador o no. Amb això obtenim l'angle al que hem de instanciar el primer laser (si hem d'instanciar més d'un làser calculem els angles a partir d'aquest angle inicial i el nombre de lasers). Un cop hem instanciat els lasers, en el cas del LaserWarning, si tenim marcat que el laser ha de seguir al jugador, recalculem l'angle a cada frame per a que apunti al jugador, mentre que al Laser només ho fem al primer frame. A més, en el cas del Laser podem especificar una velocitat de rotació dels lasers al voltant de l'enemic. Veure Figura 6.5.4.5.

El tercer atac és una variació del segon (veure Figura 6.5.4.6), i només s'executa en segona fase. Aquest atac instancia 6 LaserWarning i després 6 Lasers que van rotant al voltant de l'enemic.

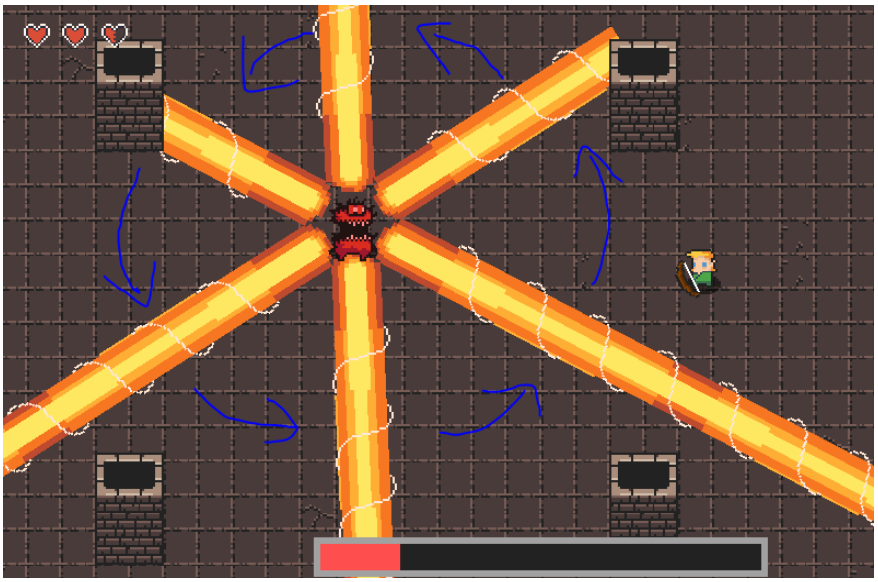

Figura 6.5.4.5: Exemple rotació lasers

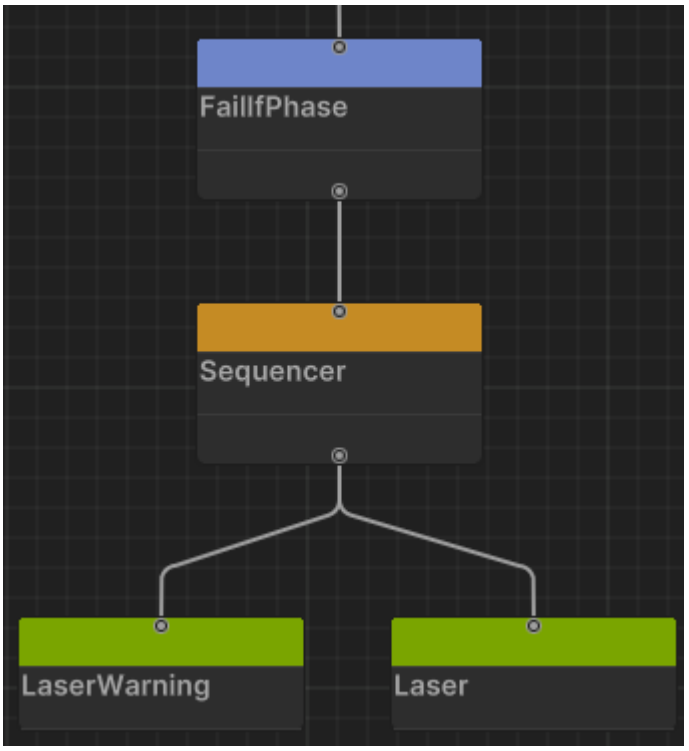

Figura 6.5.4.6: Tercer atac

El quart atac només s'executa en primera fase i dispara un prefab de projectil tipus bomba cap al jugador. Aquest projectil té associat un script que va reduint la velocitat fins arribar a 0, i quan arriba a zero instancia un prefab bomba i s'autodestrueix. El prefab bomba reprodueix una animació que mostra que està apunt de explotar i, quan explota, instancia partícules i detecta col·lisions dintre d'una area circular per fer mal al jugador. Veure Figura 6.5.4.7.

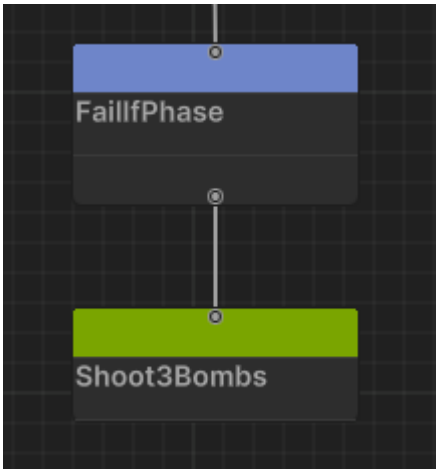

Figura 6.5.4.7: Quart atac

El cinquè i últim atac funciona igual que el quart pero dispara 3 projectils de tipus bomba en comptes de 1 i només s'executa en segona fase. Veure Figura 6.5.4.8.

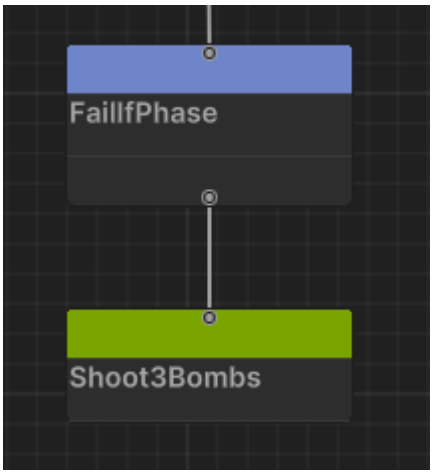

Figura 6.5.4.8: Cinquè atac

6.5.5 Nigromant

A continuació explicarem els Behaviour Trees del Nigromant i els enemics que invoca, així com els nodes específics desenvolupats. Veure Figura 6.5.5.1.

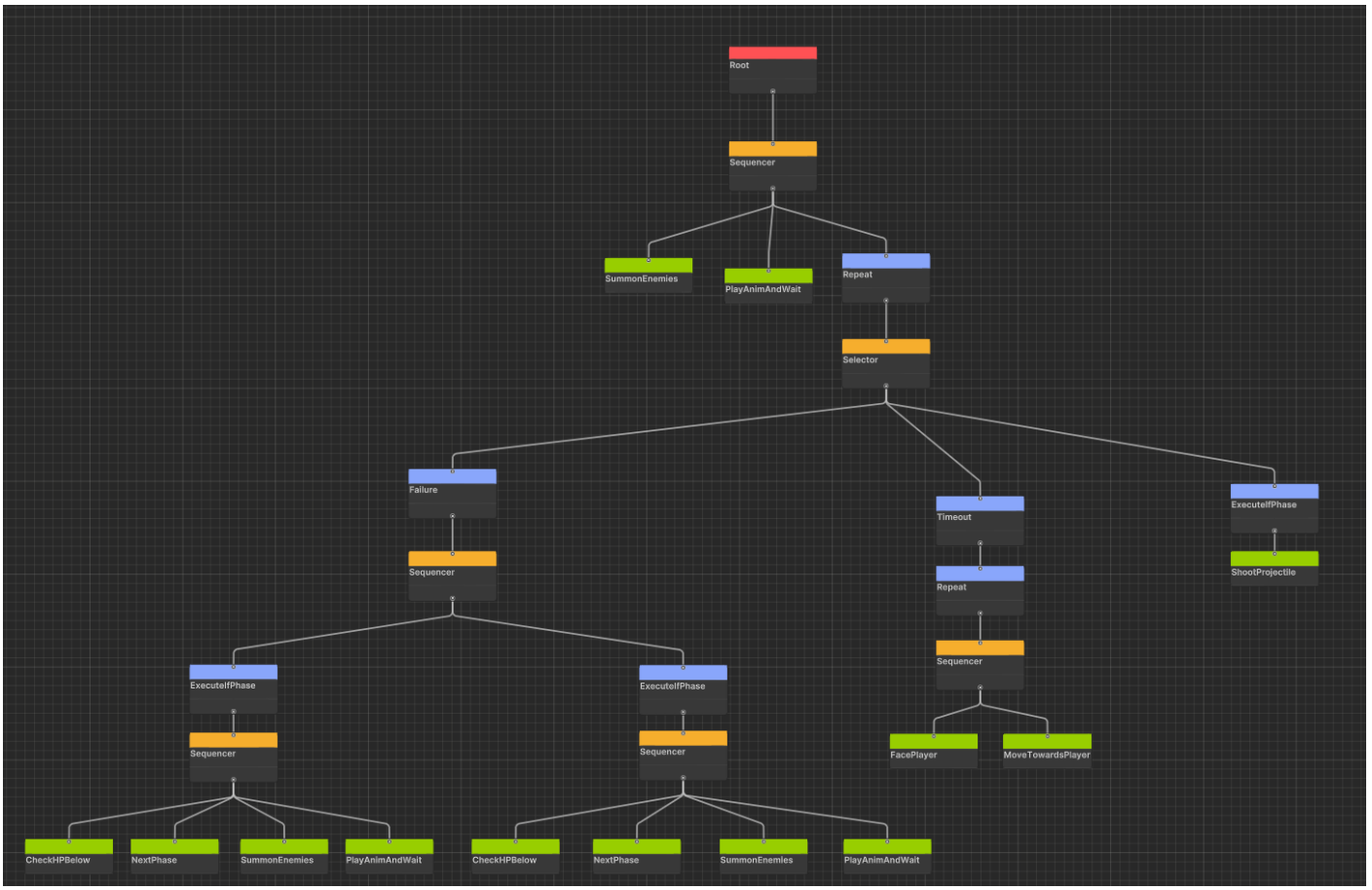

Figura 6.5.5.1: Behaviour Tree Nigromant

L'arbre del Nigromant te algunes diferències respecte als anteriors a l'inici i a la primera branca.

- Abans del repeat, al primer bucle de l'arbre, executem els nodes SummonEnemies i PlayAnimAndWait per invocar la primera olejada d'enemics.
- A la primera branca sota el repeat afegim un nou comprovar fase, ja que el Nigromant té 3 fases. La primera al 100% de vida on invoca esquelets, la segona al 66% de vida on invoca xamans i la tercera i última al 33% de vida on invoca cavallers.

La segona branca és igual a la del Dimoni, ja que a aquest boss també li interessa mantenir la distància.

Finalment la tercera branca només s'executa en tercera fase i consta simplement del node ShootProjectile, que dispara un prefab d'un projectil en direcció al jugador.

Enemics Invocats

Esquelet

L'esquelet és l'enemic més bàsic dels que pot invocar el Nigromant. Abans del repeat executa una animació inicial mirant cap al personatge (nodes FacePlayer i PlayAnimAndWait) on es veu a l'enemic sortint del terra, i al repeat mira cap (node FacePlayer) al personatge i camina cap a ell (MoveTowardsPlayer) per intentar fer-li mal per contacte. Veure Figura 6.5.5.2.

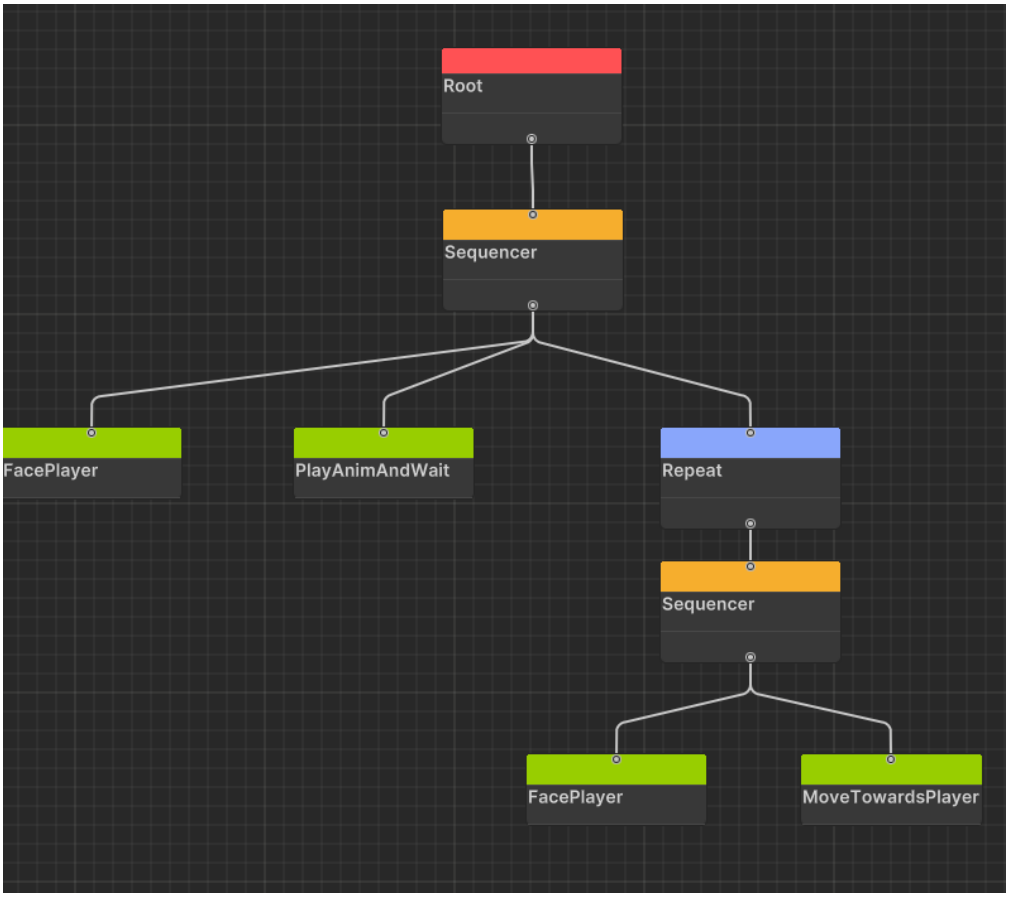

Figura 6.5.5.2: Behaviour Tree Esquelet

Xaman

El xaman funciona igual que l'esquelet al principi, pero al repeat va canviant entre perseguir (node MoveTowardsPlayer) al personatge i disparar (node ShootProjectile). Veure Figura 6.5.5.3.

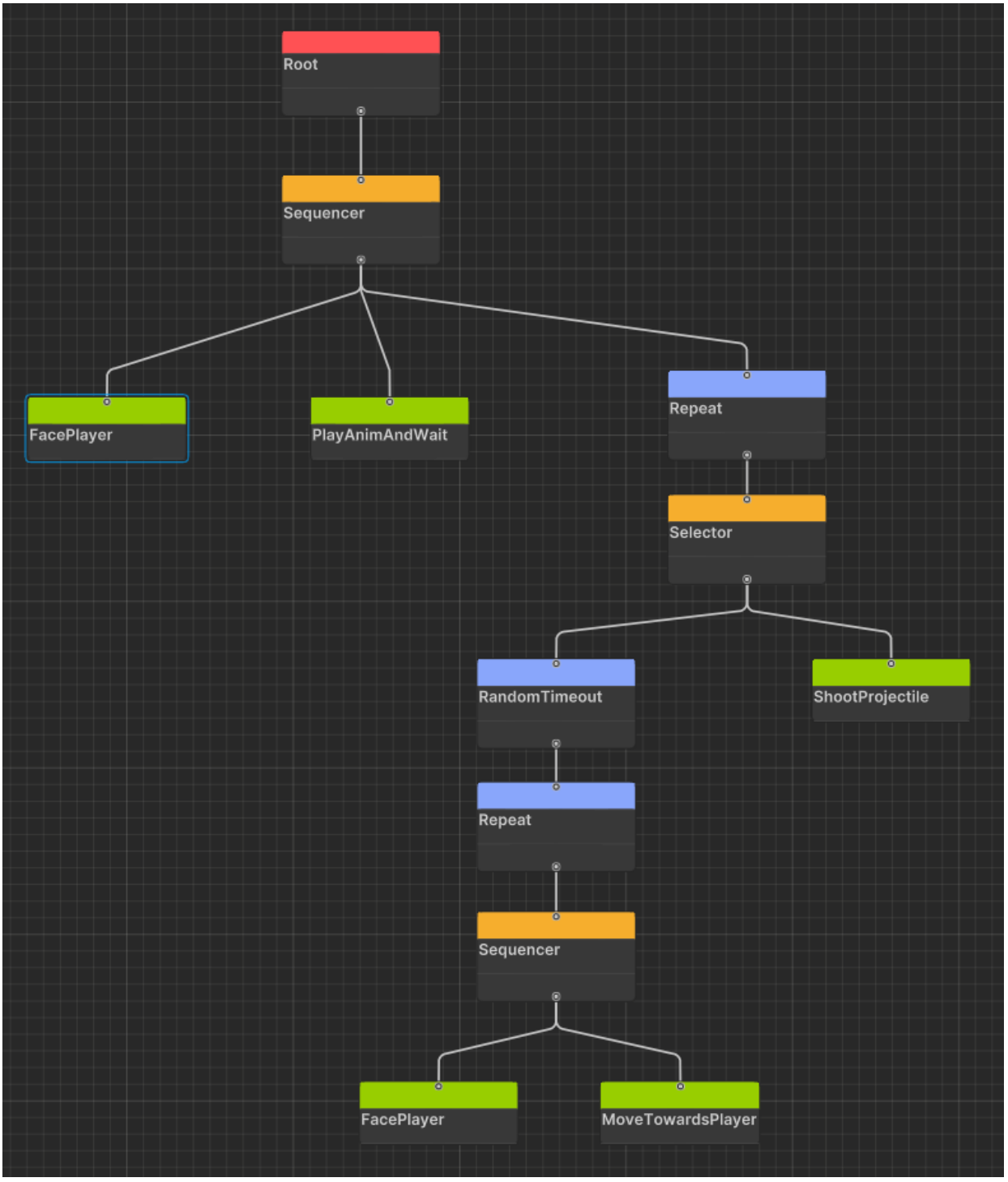

Figura 6.5.5.3: Behaviour Tree Xaman

Cavaller

El cavaller és una versió molt més ràpida que l'esquelet, pero en comptes de buscar fer mal per contacte, ataca amb l'espasa (PlayAnimAndWait) quan el jugador esta a rang (PlayerInCircle). Veure Figura 6.5.5.4.

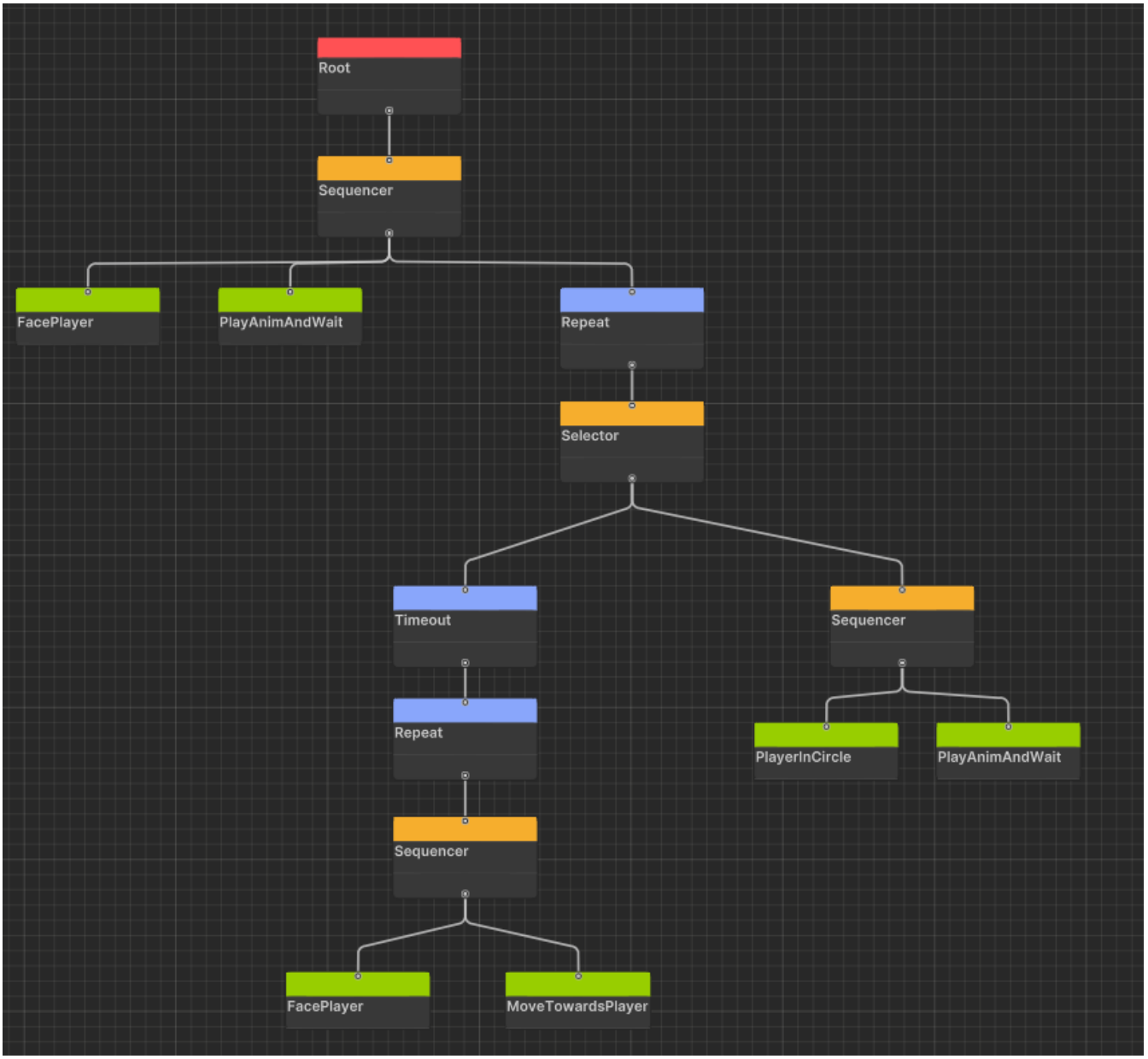

Figura 6.5.5.4: Behaviour Tree Cavaller

6.6 Implementació d'efectes visuals

6.6.1 Partícules

Per a la implementació de partícules hem utilitzat el component ParticleSystem de Unity, que proporciona una gran quantitat de paràmetres personalitzables amb els quals podem generar qualsevol effecte que necessitem.

Les partícules que hem generat amb aquest sistema són les vistes anteriorment a l'apartat 5.6.1, aprofitant principalment la emissió de tipus burst, per simular una explosió.

6.6.2 Altres efectes

Dash

Per fer l'efecte del dash hem creat un prefab amb un sprite del personatge semi transparent amb una animació de fade out. Durant el dash es van instancian aquests prefabs cada cert temps, donant l'efecte vist a la Figura 6.6.2.1.

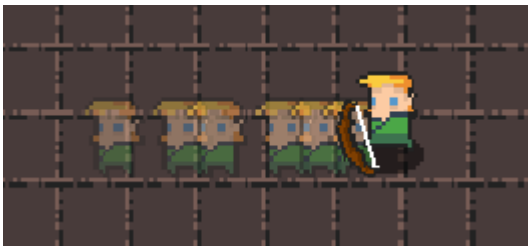

Figura 6.6.2.1: Dash

WhiteFlash

Per deixar ben clar al jugador quan el personatge principal o algun enemic rep mal hem creat un material completament blanc que apliquem als personatges amb un script WhiteFlash. A aquest script subscribim la funció Flash (veure figura 6.6.2.2) al event OnHit del sistema de vida.

```
2 referencias
public void Flash() {
    if (flashRoutine != null) {
        StopCoroutine(flashRoutine);
    }
    flashRoutine = StartCoroutine(FlashRoutine());
}

1 referencia
private IEnumerator FlashRoutine() {
    spriteRenderer.material = flashMaterial;
    yield return new WaitForSeconds(duration);
    spriteRenderer.material = originalMaterial;
    flashRoutine = null;
}
```

Figura 6.6.2.2: Funcio Flash

6.7 Proves realitzades

Al llarg de tot el desenvolupament hem anat realitzant proves contínuament per verificar el correcte funcionament de les diferents mecàniques i funcionalitats que anàvem implementant.

Principalment hem fet servir el mòdul “Debug” i la funció Debug.Log() combinat amb la pestanya de Debug. També hem fet servir la funció OnGizmosDraw() per mostrar diferents elements geomètrics a la pantalla com l'àrea de detecció dels enemics. Veure Figura 6.7.1.

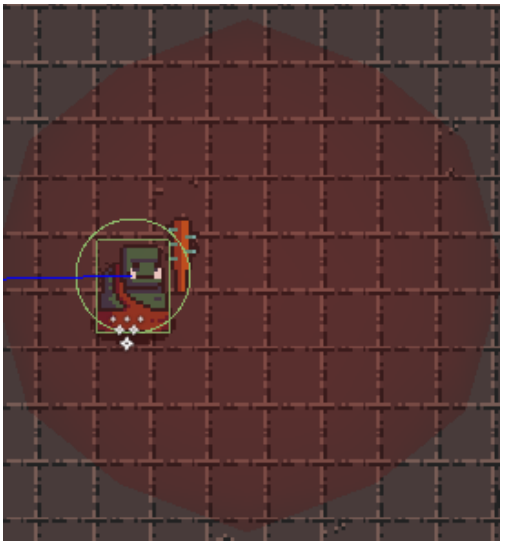

Figura 6.7.1: Area detecció ogre

7. Resultats

7.1 Legislació i normativa vigent

Pel que fa a problemes legislatius, el joc no te ningun problema, ja que en cap moment demanem ni guardem dades de caràcter personal del jugador, per tant no s’aplica la LOPD (Llei Orgànica de Protecció de Dades). Tampoc apliquem la LSSICE (Llei de Serveis de la Informació i Comerç Electrònic), ja que no es poden realitzar compres de ningun tipus al joc.

Tampoc hauríem de tenir problemes de Copyright, ja que tots els recursos utilitzats han estat creats per l’equip o son de lliure ús. Cal destacar que Unity només és gratuït mentres la persona o equip que utilitzin la eina guanyin menys de 100.000$ anuals amb els productes desenvolupats amb la mateixa.

7.2 Pegi

El PEGI s’utilitza per qualifica els videojocs mitjançant icones (veure figura 7.2.1), mostrant l’edat recomanada i descriure el contingut sensible que apareix en el joc, com ara la presència de violència, drogues o sexe. Són classificacions informatives, no de compliment obligatori.

Figura 7.2.1: Icones PEGI

En el cas del nostre projecte, al ser un joc de combat, aquest hauria de portar l'etiqueta de violencia, però al ser un tipus de violència molt poc gràfic i amb un estil poc realista, podem qualificar el joc com a PEGI 7.

7.3 Resultat final

En aquest apartat mostrarem diferents captures del joc per veure el resultat final. A més, en el següent enllaç podem veure un vídeo d'una partida completa del joc:
https://youtu.be/M9qXTsndFOE

Figura 7.3.1: Menú principal

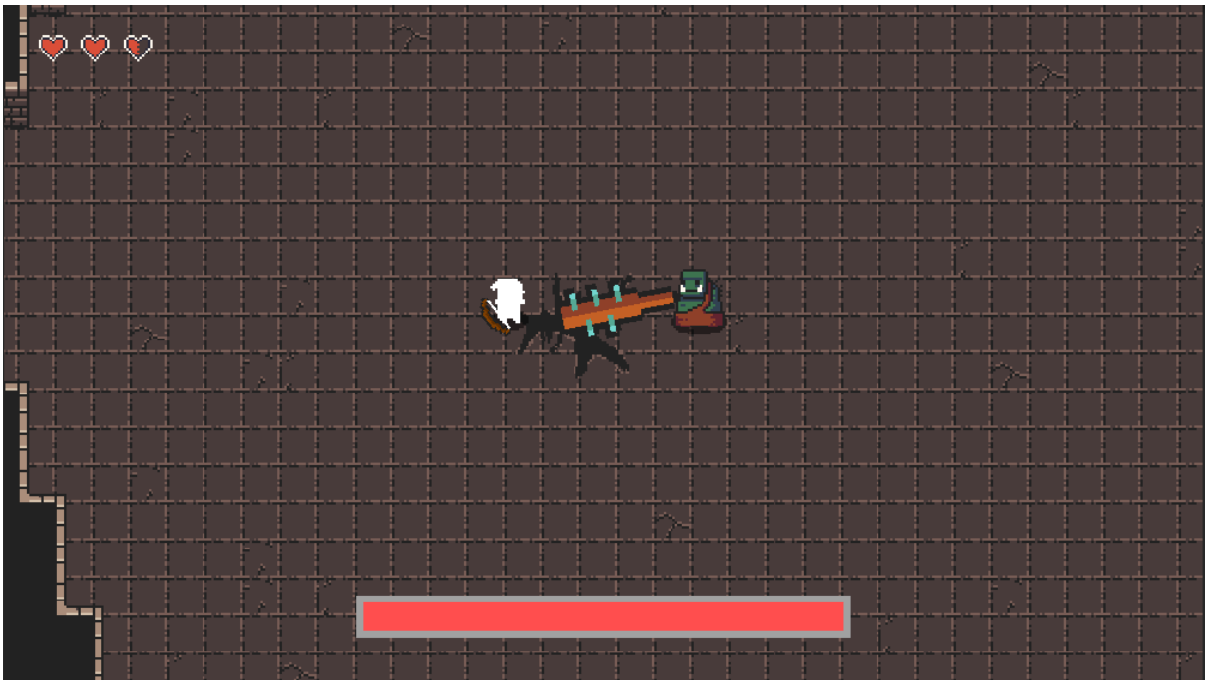

Figura 7.3.2: Ogre picant al jugador

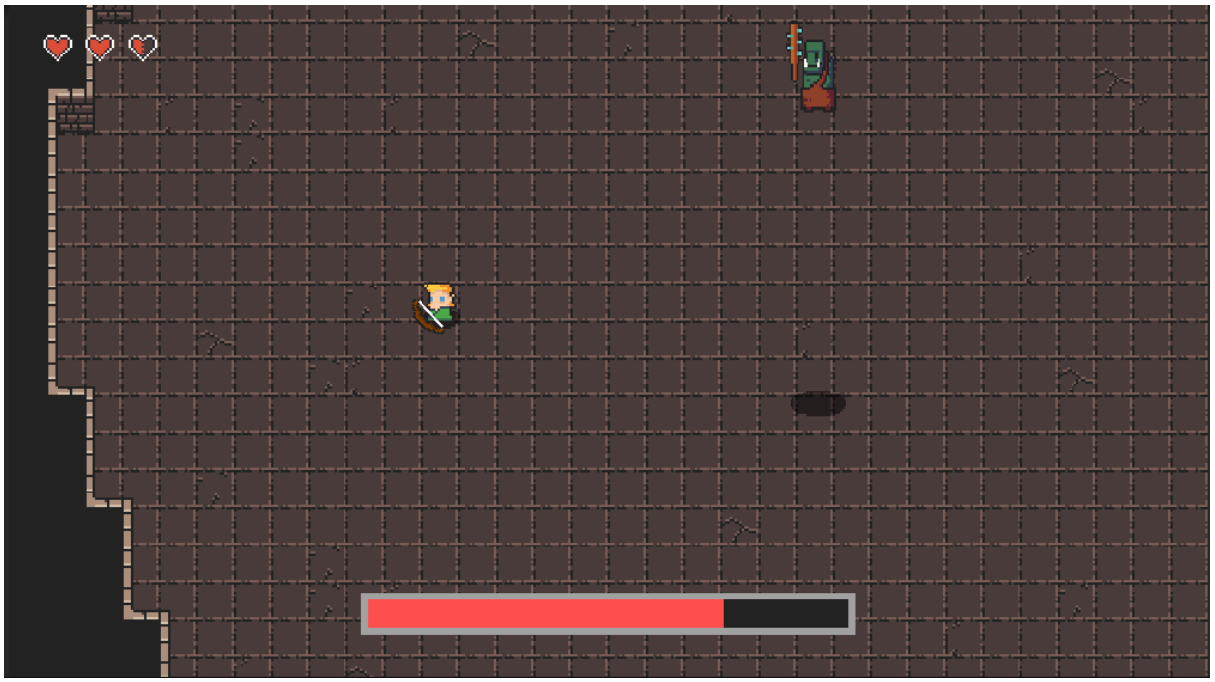

Figura 7.3.3: Ogre saltant cap al personatge

Figura 7.3.4: Ogre dash + cop

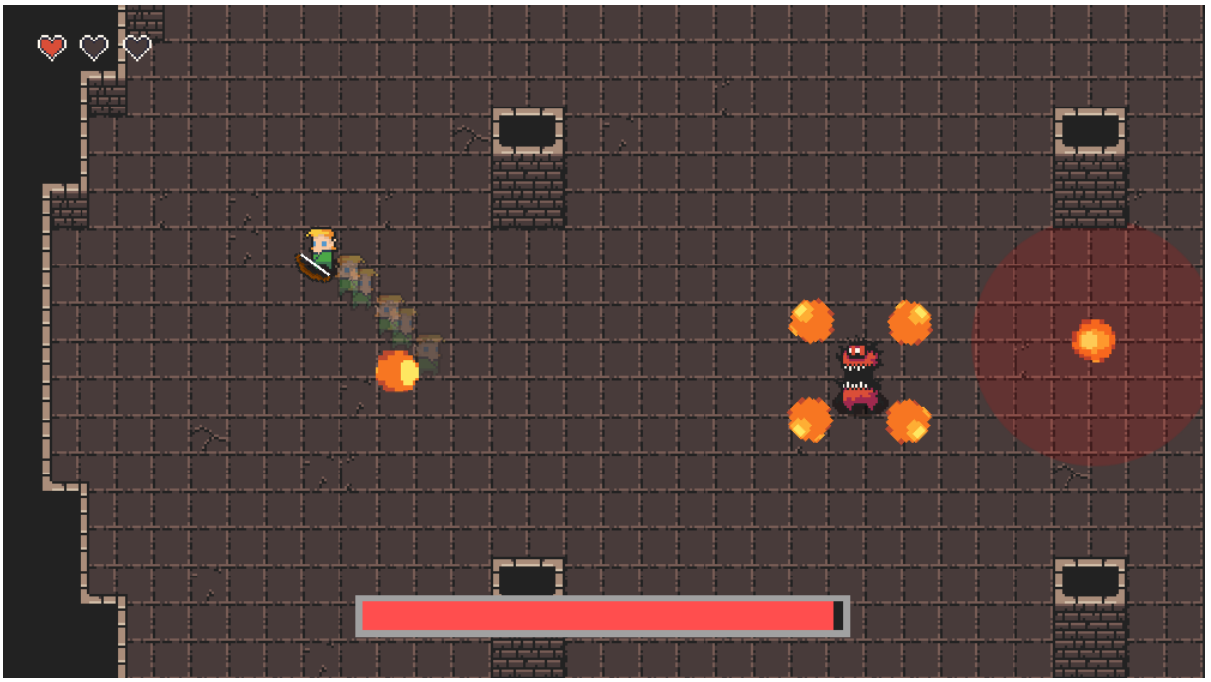

Figura 7.3.5: Jugador esquivant projectils

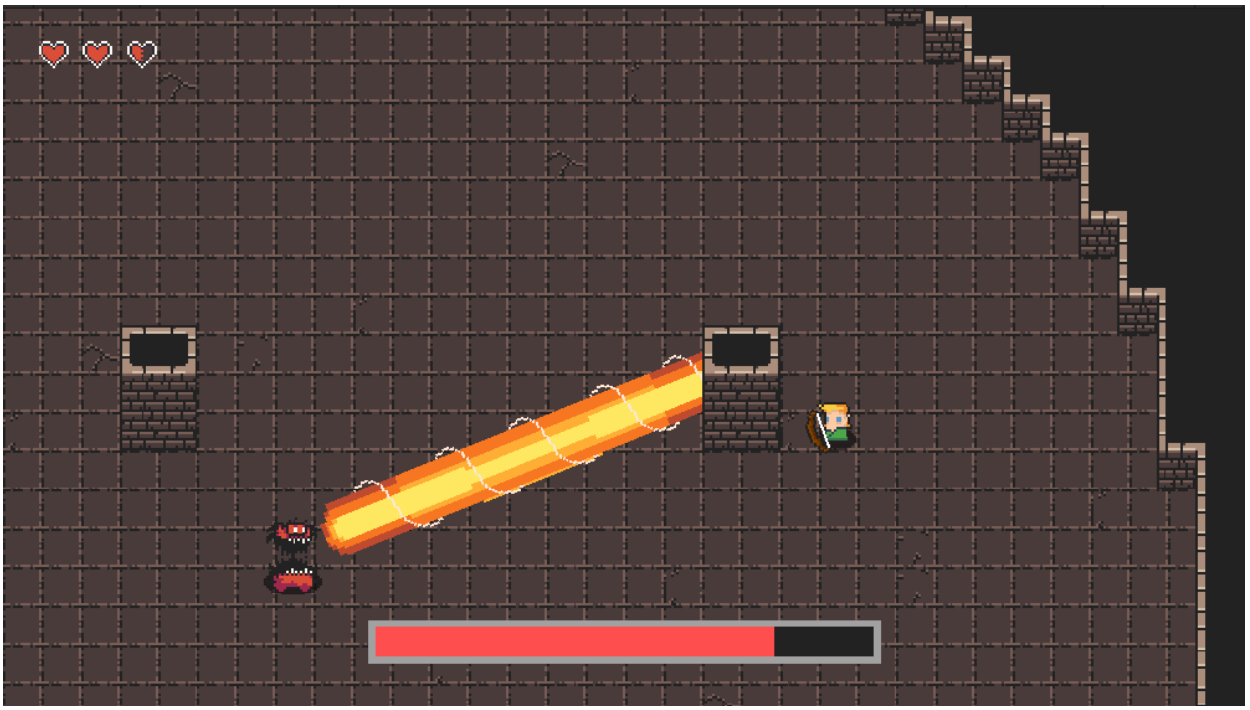

Figura 7.3.6: Jugador aprofitant el terreny per protegir-se

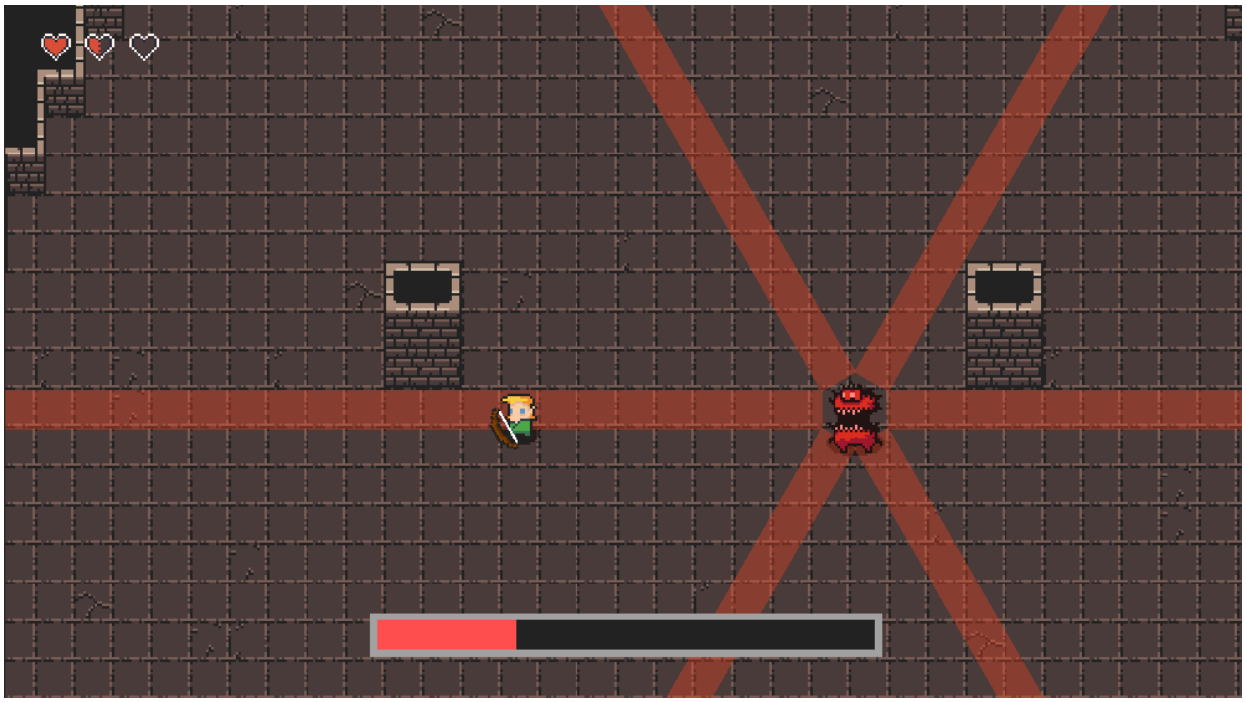

Figura 7.3.7: Avis atac 6 làsers.

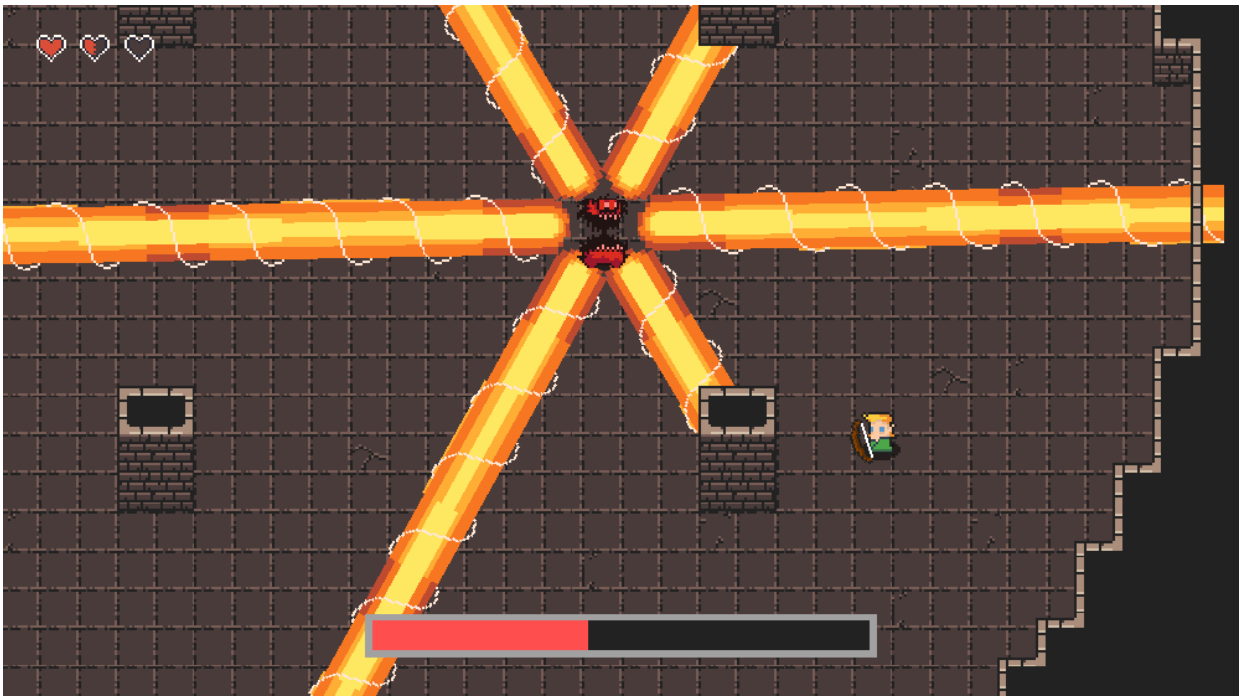

Figura 7.3.8: Atac 6 làsers.

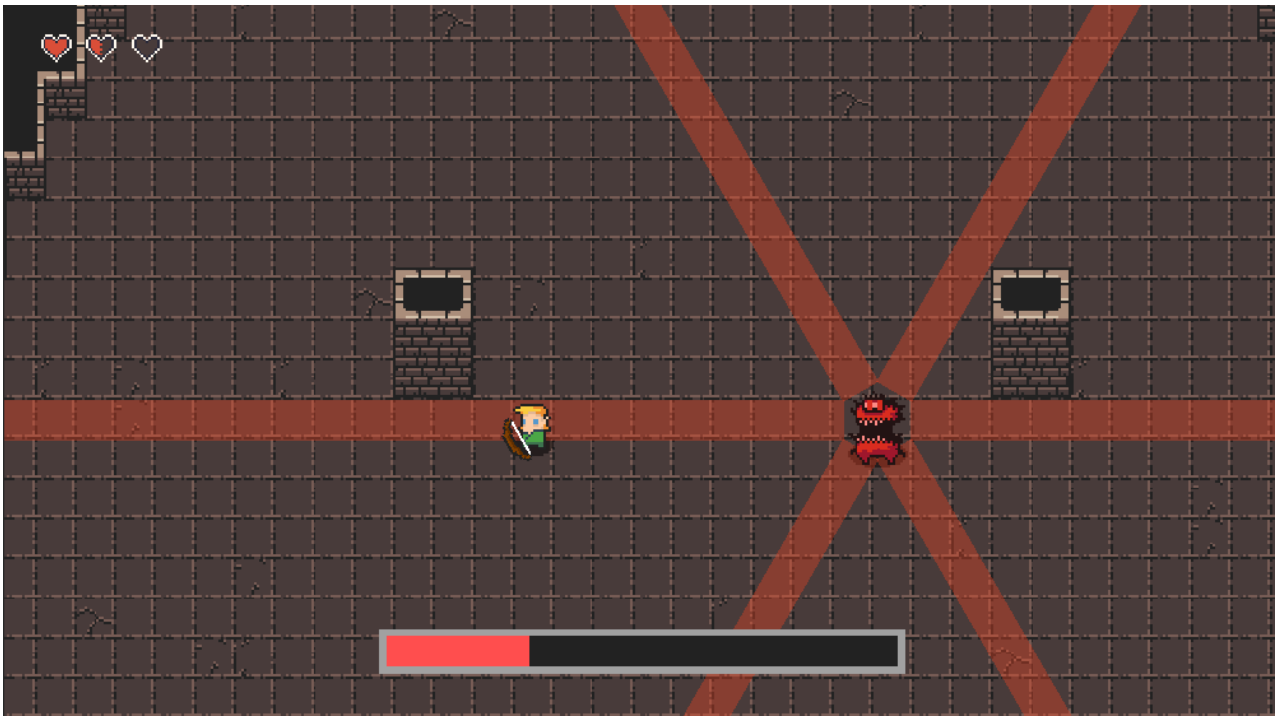

Figura 7.3.9:

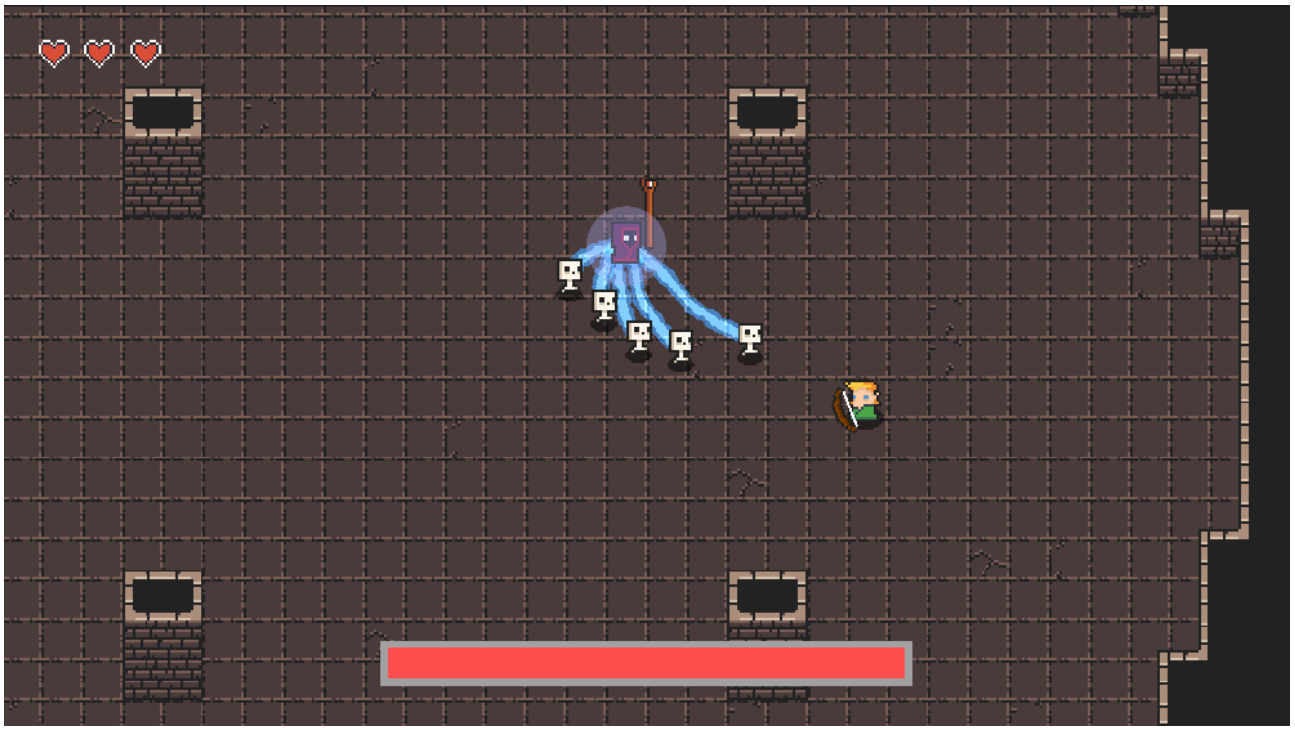

Figura 7.3.10: Nigromant amb esquelets

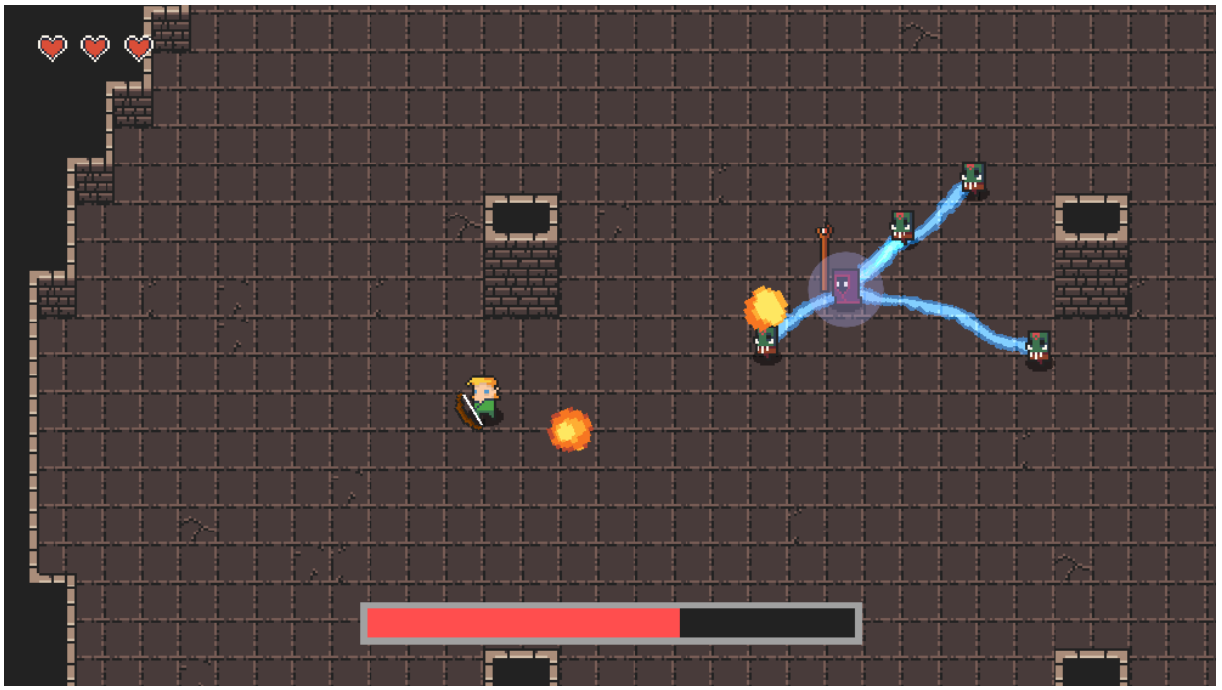

Figura 7.3.11: Nigromant amb xamans

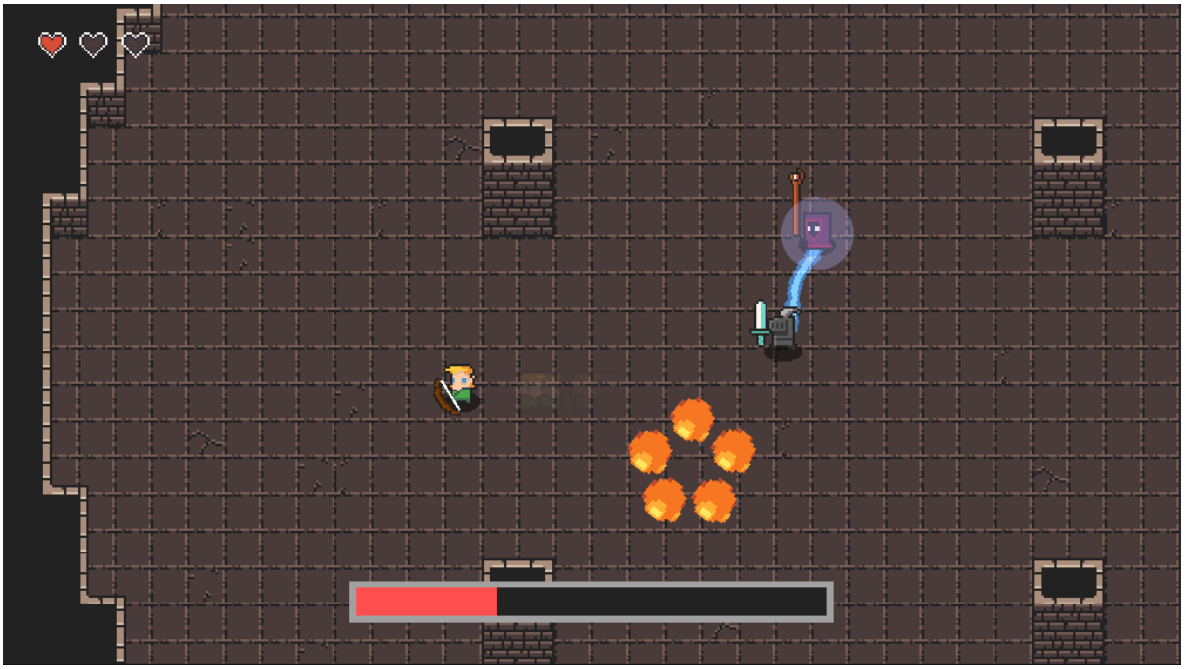

Figura 7.3.12: Nigromant amb cavallers

Figura 7.3.13: Pantalla final

8. Conclusions

8.1 Valoració del treball

Estic molt content amb el resultat final del projecte, ja que s'ha pogut complir amb pràcticament tots els objectius.

Ja havia realitzat alguns projectes personals i acadèmics utilitzant Behaviour Trees per fer la intel·ligència artificial, però aquest és el primer cop que feia un projecte complet des de zero centrat principalment en aquest aspecte, cosa que m'ha ajudat molt a entendre millor totes les etapes i tasques que conformen el desenvolupament d'un videojoc. A més, a l'haver centrat el projecte en Behaviour Trees, he après molt sobre com desenvolupar nodes reutilitzables i construir els arbres més complexos.

Pel que fa al motor de Unity, tot i ja haver fet diversos projectes amb ell, aquest ha estat el primer cop que realitzava un projecte tan gran de forma individual, cosa que m'ha fet aprofundir més en aspectes que mai havia tocat o que havia tocat molt per sobre.

8.2 Desviacions de la planificació original

Hem seguit la planificació original pràcticament al 100%, per tant no hem tingut grans desviacions més enllà de petits detalls que hem retocat/pulit un poc cap al final del projecte.

9. Treball futur

Tot i que el joc ara mateix és completament jugable, hi ha una serie de detalls que podem millorar/afegir per donar el joc per acabat al 100%.

- **Afegir opcions gràfiques:** Actualment, no es pot modificar la resolució del joc, ni si es pot posar en pantalla completa.

- **Afegir efectes de so:** Per donar més immersió i feedback al jugador.

- **Zones de combat temàtiques:** Donar zones de combats temàtiques a cada boss ajudaria molt a augmentar la immersió del jugador i donar vida al joc. Per exemple que el Dimoni estigui a una zona de foc.

- **Més bosses:** Tot i que el joc ara té 3 bosses ben diferenciats, guanyaria molt afegint més bosses interessants, per allargar la duració del joc.

10. Bibliografia

- Unity Technologies (2022). Unity Manual. https://docs.unity3d.com/Manual/index.html
- Unity Technologies (2022). Unity Forum. https://forum.unity.com
- Unity Technologies (2022). Unity Answers. https://answers.unity.com/index.html
- GitHub, Inc (2022). GitHub. https://github.com
- Stack Exchange, Inc (2022). https://stackoverflow.com
- TheKiwiCoder, canal de youtube “TheKiwiCoder”. https://www.youtube.com/TheKiwiCoder
- TheKiwiCoder (2022). TheKiwiCoder. https://thekiwicoder.com
- Fundación Wikimedia, Inc. Wikipedia. https://wikipedia.org
- Brackeys, canal de youtube “Brackeys”. https://www.youtube.com/Brackeys

11. Anexos

S’ha adjuntat l’executable del joc, i un enllaç al projecte sencer de Unity, amb tots els assets creats i tot el codi escrit. El projecte està organitzat per carpetes (escenes, scripts, animacions…). També hem adjuntat un video on mostrem una partida completa, i captures d’alta resolució on es poden veure els Behaviour Trees de tots els enemics.

12. Manual d’usuari

12.1 Iniciar el joc

Per a iniciar el joc, a la carpeta “Dungeon Rush”, hem de buscar i executar el fitxer executable “DungeonRush.exe”. Un cop executat s’obrirà el menú principal. Un cop aquí fem clic al botó “Play” i s’iniciarà el primer boss.

12.2 Controls teclat

- **Moviment:** Tecles WASD.
- **Apuntar:** Movent el ratolí.
- **Disparar:** Botó esquerre del ratolí. Mantenir pulsat per carregar el tir.
- **Dash:** Barra espaiadora. Fa el dash en direcció al ratolí.

12.3 Controls comandament

Veure figura 12.3.1.

Figura 12.3.1: Pantalla final