Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Disseny i Desenvolupament de Videojocs

Títol: Desenvolupament d'un joc amb Unity d'estil roguelike-like

Document: Memòria

Alumne: Quim Mayoral Prats

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

Convocatòria (mes/any) Septembre/2022

Índex

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

1.1 Introducció

Des dels inicis de la indústria dels videojocs aquests s'han classificat en diferents gèneres o conceptes. Desde acció a puzzles, passant per estratègia i al gènere on ens centrarem, anomenat Rogue like. D'aquest gènere s'ha dit molt, però durant aquests últims anys es parla d'una pèrdua de significat a causa de la proliferació de l'ús de la paraula per definir diferents productes que es desvíen de la definició original i innoven a la seva manera.

Figura 1: The Binding of Isaac (2011)

Però tots aquests productes encara comparteixen característiques bàsiques, una d'aquestes és la generació procedural, en la qual ens centrarem per poder produir diferents escenaris al nostre joc i així assegurar la rejugabilitat, a més a més de fer un joc on el jugador mai tindrà la sensació de repetir escenari.
Un altre tema important en aquesta classe de jocs és la Inteligencia artificial (IA), ja que sense aquesta tindriam un escenari buit amb enemics arcaics i avorrits. En aquest projecte intentarem produir una IA que dificulti l'experiencia del jugador utilitzant la idea de Behaviour Trees (la idea de representar gràficament part del codi).

1.2 Motivació

Basant-nos en l'apartat anterior, podem definir la motivació principal d'aquest projecte en crear un joc d'estil rogue-like on la generació procedural i la creació d'una intel·ligència artificial siguin clau en el desenvolupament, ja que des d'una mirada personal són dos temes els qual em fascinen.

Tant la capacitat d'un codi per milers d'escenaris amb casi nula aportació d'un operador humà i oferint diversos resultat final com la construcció de comportament i accions a través d'un sistema semi-gràfic (Behaviour trees) per simplificar la creació d'aquests tipus de sistemes així reduint la càrrega de feina i simplificant tot el procés de producció.

També no es pot passar per alt l'oportunitat que suposa aquest projecte per desenvolupar i crear un videojoc amb tots els coneixements que hem acumulat durant aquests anys de formació a la politécnica.

1.3 Propòsit i objectius del projecte

El propòsit d'aquest projecte, com hem mencionat a l'apartat anterior, és la creació d'un joc estil rogue like on la generació procedural i la IA juguin un paper important.

Els objectius del projecte són els següents:

- Utilitzar un sistema de generació procedural en la creació del mapa, el qual permet la modificació de diverses característiques per crear un "End Product" diferent i personalitzat.

- Utilització del concepte Behaviour Tree per a la implementació dels enemics

- Millorar la capacitat de solucionar i localitzar tant errors de codi com la capacitat de testejar els elements creats.

- Intentar crear una estética unificada.

- Aprofundir i millorar els meus coneixements de el motor Unity.

1.4 Distribució de tasques

Donat el meu perfil tècnic, la distribució de les diferents tasques del projecte sera la següent:

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

Com hem explicat en els apartats anteriors, aquest joc es centrara sobretot en les mecàniques típiques d'aquest gènere i la tecnologia necessària per aconseguir un producte final satisfactori als nostres objectius.

Tot i que també tenim que recordar que la creació de nivells i la interfície d'usuari recauen sobre l'apartat d'estètica, per tant caldria invertir-hi temps de desenvolupament.

Finalment la narrativa tindrà un pes molt baix i casi nul en aquest projecte ja que, tot i ser part integra d'aquest gènere, en aquest projecte no ens hi centrarem.

2. Estudi de viabilitat

2.1 Recursos necessaris i viabilitat

2.1.1 Recursos tècnics

Aquest projecte s'ha realitzar amb el següent hardware:

Ordinador portàtil model **Asus Astrix GL753VD**:

- Targeta gràfica (Graphics Processor Unit o GPU): NVIDIA GeForce RTX 1050
- Processador (Central Processor Unit o CPU): Intel Core i7-7700HQ 2.80GHz
- Memòria (Random Acces Memory o RAM): 16GB DDR 4

2.1.2 Recursos humans

Darrera cada videojoc desenvolupat hi ha un equip format per varis individus amb rols i responsabilitats diferents, els quals es poden categoritzar en diversos perfils:

- **Dissenyador:** l'encarregat de definir el joc, els objectius, narrativa i qualsevol element que ha d'aparèixer.

- **Programador:** encarregat de la implementació del joc, desenvolupament del codi i dels aspectes més tècnics del projecte.

- **Artista:** l'encarregat de dissenyar i dibuixar els components visuals del joc.

- **Dissenyador de so:** encarregat de dissenyar la música i els efectes de so del projecte.

Ja que en el nostre cas només treballara un individu, haurem de prioritzar diferents tasques i buscar contingut extern (art, sons, …) per suplir el déficit de personal i temps.

2.1.3 Recursos tecnològics

El software utilitzat per tal de desenvolupar aquest joc ha estat:

- Sistema operatiu: Windows 10 Home
- Motor de videojocs: Unity versió 2020.3.18f1 amb IDE Visual Studio 2020

2.1.4 Viabilitat econòmica

Ja que disposem del material necessari per desenvolupar aquest projecte, no ens farà falta afegir el preu de la maquinaria ni de les llicències ja que els programes utilitzats són gratuïts. Per tant només ens faltara calcular els salaris hipotetics del equip que desenvoluparia el videojoc (En cas que fos necessari serien aproximadament 1100 de hardware més 100 de software fent un total aproximat de 1200€):

- Dissenyador de joc: 16 €/hora
- Programador: 13 €/hora
- Artista: 13 €/hora
- Dissenyador de so: 12 €/hora

A traves de la web www.payscale.com obtindrem el salari que reben els hipotetics treballadors. Com que no hem necessitat contractar a personal, el cost real de recursos humans del projecte és 0€.

2.2 Estudi de mercat

Com hem comentat anteriorment, el gènere rogue like inclou moltes definicions i interpretacions, però es podria dividir en dos subapartats: gràfics i mecàniques:

-Gràfics: Es poden dividir en tres subapartats

-Purista: Els rogue like han de fer servir caracters de text per representar el món.

-Neutral: Els rogue like tenen uns gràfics basats en "tiles".

-Radical: Els rogue like no tenen uns gràfics predefinits.

-Mecàniques:

-Purista: Els rogue like són "turn based dungeon crawler" i utilitzen sistemes de generació procedural amb mort permanent.

-Neutral: Els rogue like utilitzen sistemes de generació procedural amb mort permanent.

-Radical: Els rogue like no tenen unes mecàniques predefinides.

2.3 Públic objectiu

El públic objectiu per al videojoc que desenvoluparem en aquest projecte és aquell que gaudeix d'un estil de joc ràpid, dinàmic, casual amb llibertat de moviment.
La seva prioritat és combatre els diferents enemics i bosses, aprendre les diverses maneres per superar els diferents reptes i escollir la més apropiada al seu estil de joc i gaudir de les mecàniques de joc associades a aquest gènere, deixant de banda el pes de la narrativa usual dels altres gèneres.

Assumirem que el públic objectiu ja esta familiaritzat amb jocs i mecàniques similars a aquest gènere a causa que busca experiències semblants.

3. Planificació

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

3.1. Tasques a realitzar

- Planificació del projecte.
- Redacció i preparació de la memòria.
- Implementació de la generació procedural de nivells.
- Disseny dels prefabs que conformen el nivell.
- Modelatge/Obtenció d'objectes/escenaris.
- Animacions dels personatges.
- Disseny de mecàniques, reptes i accions.
- Estudi de viabilitat.
- Selecció/creació d'efectes de so.
- Depuració d'errors.
- Identificació del públic objectiu.
- Disseny/Obtenció d'objectes/escenaris.
- Disseny de la interfície.
- Selecció de l'entorn de treball.
- Implementació de mecàniques.
- Implementació de la interfície.

- Implementació de la IA dels enemics.
- Testing.

3.2.Paquets de treball

Documentació:

- Tasques:
    - Redacció i preparació de la memòria.
- Temporització:
    - Aquesta tasca es realitza al llarg de tot el projecte.
- Fita:
    - Document.

Anàlisi de viabilitat i estudi previ

- Tasques:
    - Identificació del públic objectiu.
    - Estudi de viabilitat.
    - Planificació del projecte.
    - Selecció de l'entorn de treball.
- Temporització:
    - Setmana 1 Juny 2022 – Setmana 1 Setembre 2022
- Fita:
    - Document.

Mecànica

- Tasques:
    - Disseny dels prefabs que conformen el nivell.
    - Disseny de mecàniques, reptes i accions.
- Temporització:
    - Setmana 2 de Juny 2022 – Setmana 1 Agost 2022
- Fita:
    - Document.
    - Esquemes d'alt nivell.
    - Entorn dels nivells

Estètica

- Tasques:
    - Disseny/Obtenció d'objectes.
    - Disseny de la interfície.
    - Modelatge/Obtenció d'objectes.

    - Animacions dels personatges.
    - Selecció/creació d'efectes de so.
- Temporització:
    - Setmana 1 de Juliol 2022 – Setmana 1 d'Agost 2022
- Fita:
    - Objectes 2D.
    - Animacions 2D.
    - Arxius de so.
    - Imatges/icones d'interfície.

Tecnologia

- Tasques:
    - Implementació de mecàniques.
    - Implementació de la interfície.
    - Implementació de la IA dels enemics.
    - Implementació de la generació procedural de nivells.
- Temporització:
    - Setmana 1 Juny 2022 – Setmana 1 d'Agost 2022
- Fita:
    - Assets d'Unity
    - Projecte d'Unity.

Depuració i feedback

- Tasques:
    - Testing.
    - Depuració d'errors.
- Temporització:
    - Setmana 2 de Juny 2022 – Setmana 1 de Setembre 2022
- Fita:
    - Captures i gravacions de pantalla.
    - Esquemes de la implementació.

3.3.Esquema d'arbre dels paquets de treball

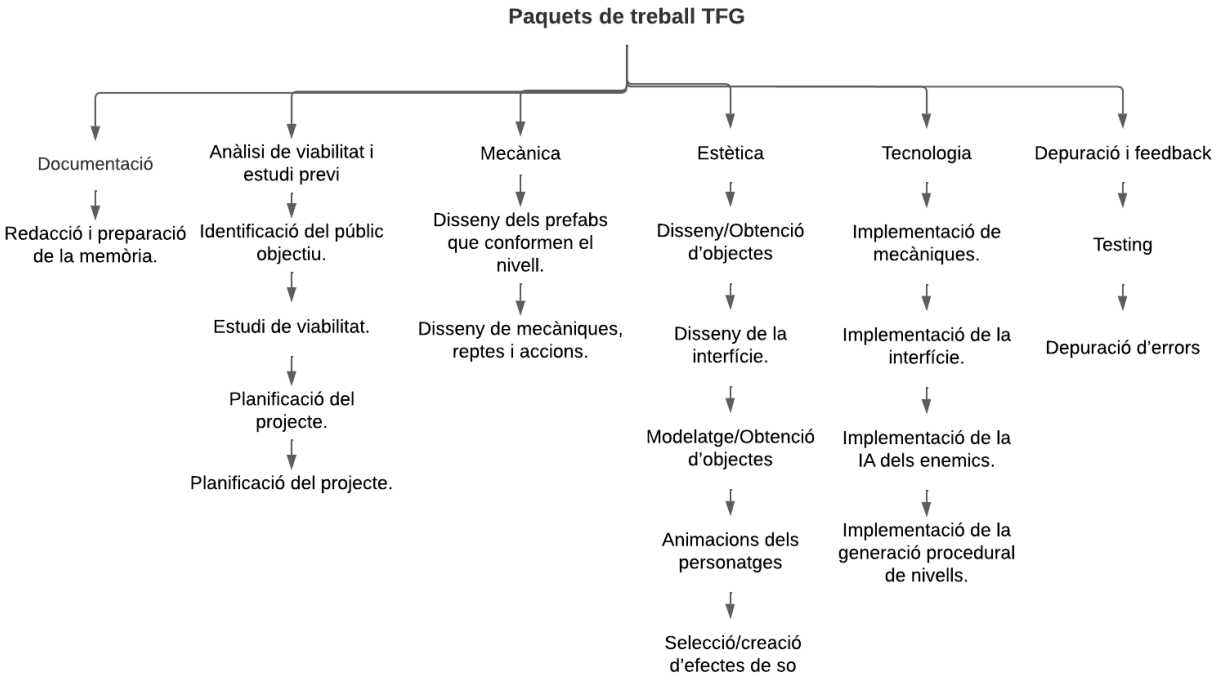


Figura 2: Esquema d'arbre dels paquets de treball i les seves respectives tasques.

3.4.Cronograma del projecte

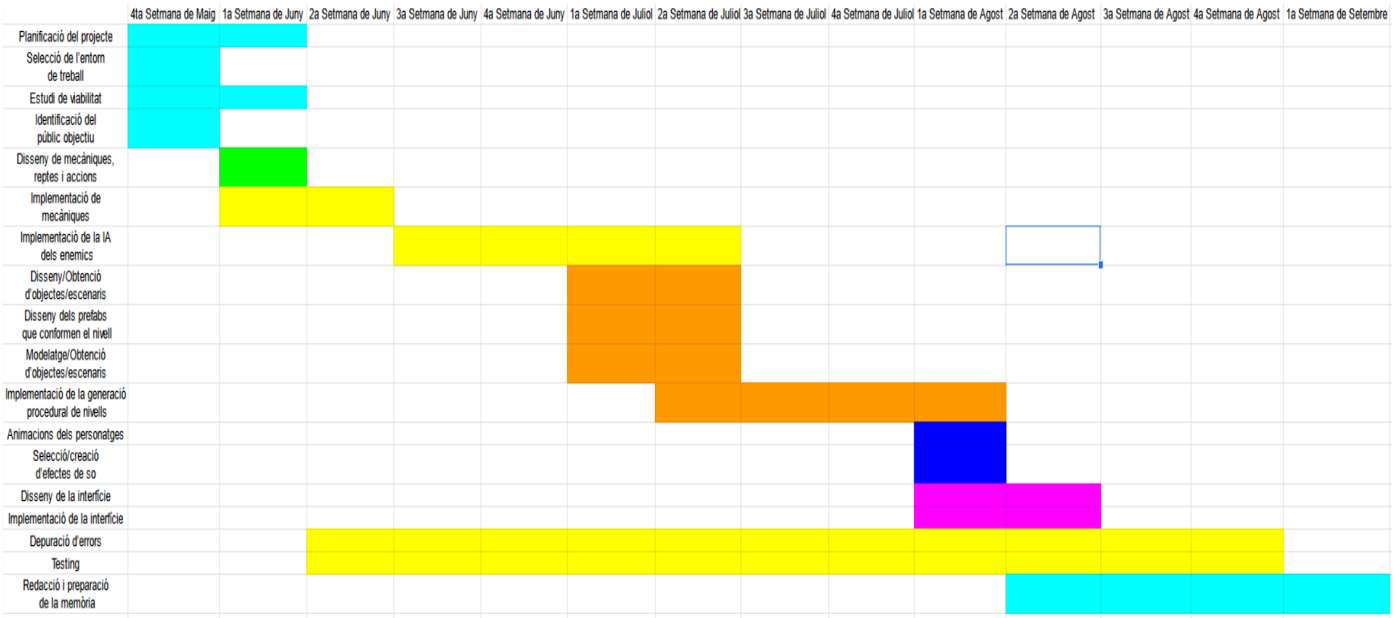


Figura 3: Diagrama de Gantt amb el cronograma estimat.

3.5.Metodologia de treball

Abans de començar a desenvolupar el projecte hem de decidir com ens organitzarem i de quina manera treballarem per dur endavant aquest videojoc. Tot seguit descriurem les claus principals de la metodologia utilitzada en aquest projecte:

- Es dividiran els diferents mesos en diferents tasques generals a realitzar (Ex: Mes d'agost documentació i debug) i les subsequents setmanes s'especificaran d'una manera més concreta (Ex: Primera setmana d'agost: recerca del format de la memòria i escriptura de la introducció), així assignant diversos temps a diferents recursos necessaris per desenvolupar la tasca mensual.

- Es faran una llista de prioritats segons tasca general, després s'assigna la dificultat estimada i el subsequent temps a dedicar, en cas de que no es completi la tasca es planteja deixar cada mes un temps de marge per a tasques no acabades o pendents de optimització

- El treball es divideix en formats de dues setmanes de duració (el temps entre tutories). Abans de començar el període de temps entre tutories decidirem què s'implementarà durant aquest i la prioritat que té cada subapartat.

- A les tutories s'expondra la feina feta i es debatirà si fa falta retallar subtasques per assegurar una tasca principal completa.

4. Marc de treball i conceptes previs

En aquesta secció detallarem quins són els conceptes que cal entendre prèviament a la lectura de la memòria i quin programari s'ha utilitzat durant el desenvolupament del projecte. També mencionarem diversos videojocs que presenten característiques similars al joc que volem desenvolupar i en aquest apartat els analitzarem

Hotline Miami - Hotline Miami 2:

Hotline Miami és un videojoc d'acció independent estil shoot 'em up desenvolupat per Dennaton Games, equip compost per Jonatan Söderström i Dennis Wedin. El més interessant d'aquest joces que, en la seqüela, hi havia un editor de nivells on l'usuari podia crear i configurar els nivells i elements com el volgués.

Figura 4: Imatge de Hotline Miami.

Nuclear Throne:

Nuclear Throne és un videojoc d'estil bullet hell, roguelike desenvolupat per Vlambeer. En aquest títol l'estructura dels escenaris de joc no té cap element predeterminat, només el tipus d'enemics i bosses als que s'haurà de fer front a cada etapa de la partida. També podem remarcar que, a diferència d'altres títols, les armes del jugador no utilitzen munició:

en lloc d'haver de gestionar aquest recurs obtenint bales, el personatge ha de controlar que la seva arma no se sobreescalfi.

Figura 5: Imatge de Nuclear Throne.

The Binding of Isaac:

The Binding of Isaac és un videojoc roguelike independent dissenyat per Edmund McMillen i programat per Florian Himsl. En aquest títol les sales són predeterminades, el que canvia és la distribució: estan connectades les unes a les altres paret amb paret. Com a curiositat, el jugador només pot disparar en quatre direccions, per tal de disparar a un enemic que es troba en diagonal, s'ha de desplaçar i fer punteria.

Figura 6: Imatge de The Binding of Isaac.

4.1 Conceptes previs

- **Generació procedural:** Es un mètode de creació de continguts a través d'algorismes, en oposició a un mètode de creació manual. En videojocs, la generació procedural s'utilitza per a que el contingut es generi a l'ordinador que els conté, en temps real, i no de manera prèvia i renderitzat en paquets gràfics predefinits. Se l'usa principalment per a la generació d'ambients i mapes, encara que també s'aplica per a intel·ligència artificial i jugabilitat.

- **Behaviour Trees:** Un arbre de comportament (o Behaviour Tree) és un model matemàtic d'execució de plans utilitzat en informàtica, robòtica, sistemes de control i videojocs. Descriuen els canvis entre un conjunt finit de tasques de manera modular. La seva força prové de la seva capacitat per crear tasques molt complexes compostes de tasques senzilles, sense preocupar-se de com s'executen les tasques senzilles.

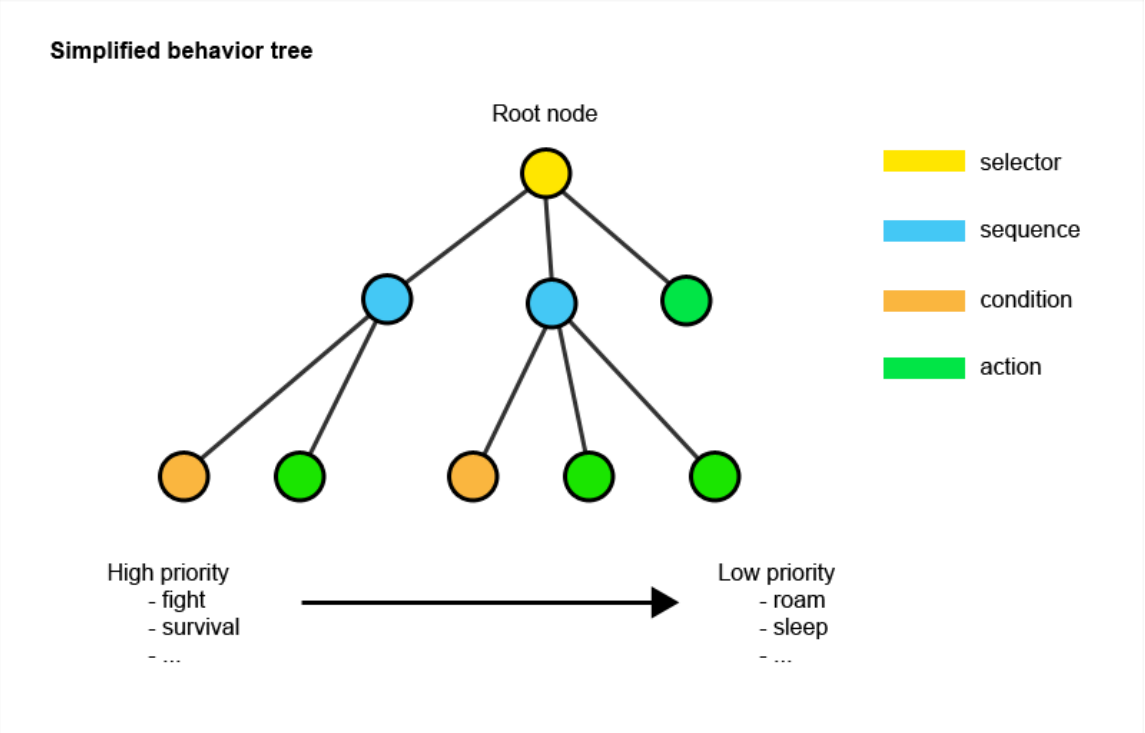


Figura 7: Exemple de Behaviour Tree

- **Blackboards:** Cada arbre de comportament manté una estructura de dades anomenada pissarra, que conté variables i valors rellevants per a un agent d'IA. Es pot considerar com la memòria de l'agent d'IA i s'utilitza per desacoblar les dades utilitzades en un arbre de comportament de la seva lògica.

4.2 Entorns de treball

Abans de iniciar el desenvolupament del projecte hem de decidir quin software farem servir per a cada tasca.

4.2.1. Motor gràfic i entorn de programació

A l'hora de decidir el motor gràfic pel prototipús cal tenir en compte les diverses necessitats que es tindran durant el desenvolupament. Tot seguit enumerarem unes quantes:

- Joc 2.5D amb càmera en tercera persona.

- Es vol utilitzar un sistema de generació procedural, a ser possible amb gran capacitat de modificació i configuració d'elements i el procés.

- Es vol dissenyar una interfície d'usuari senzilla.

- Es necessita una IA, la qual es podria implementar idealment amb un arbre de comportament, o Behaviour Tree.

- En termes de contingut, es tracta d'un projecte que ha de ser fàcilment mantingut i escalable per poder afegir complexitat a la generació d'escenaris i optimització/millora de la IA.

Després d'analitzar les necessitats del projecte i observar els diferents motors gràfics a l'actualitat, decidim utilitzar Unity Engine amb la versió 2020.3.18f1.

Ens hem decidit per aquest sistema ja que un dels objectius d'aquest projecte era millorar el nostre us del motor Unity; a més a més, tot i que Unity no disposa de un sistema natiu d'us de Behaviour Trees ni elements de generació procedural, hem estat capaços d'obtenir dos Plug-Ins. Els quals ens donaran una estructura per poder generar, assignar i utilitzar Behaviour Trees per crear la IA i l'interficie per poder configurar i generar els nivells de manera procedural, aixi ajudant-nos a complir 3 dels objectius principals d'aquest projecte.

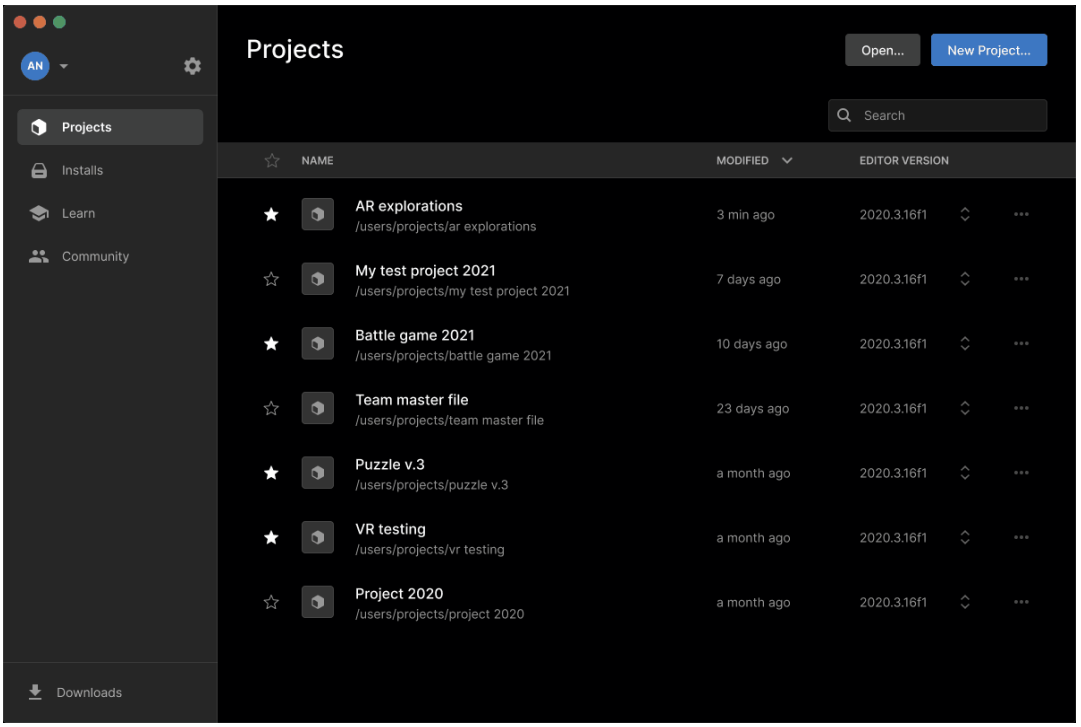


Figura 8: Launcher de Unity anomenat Unity Hub.

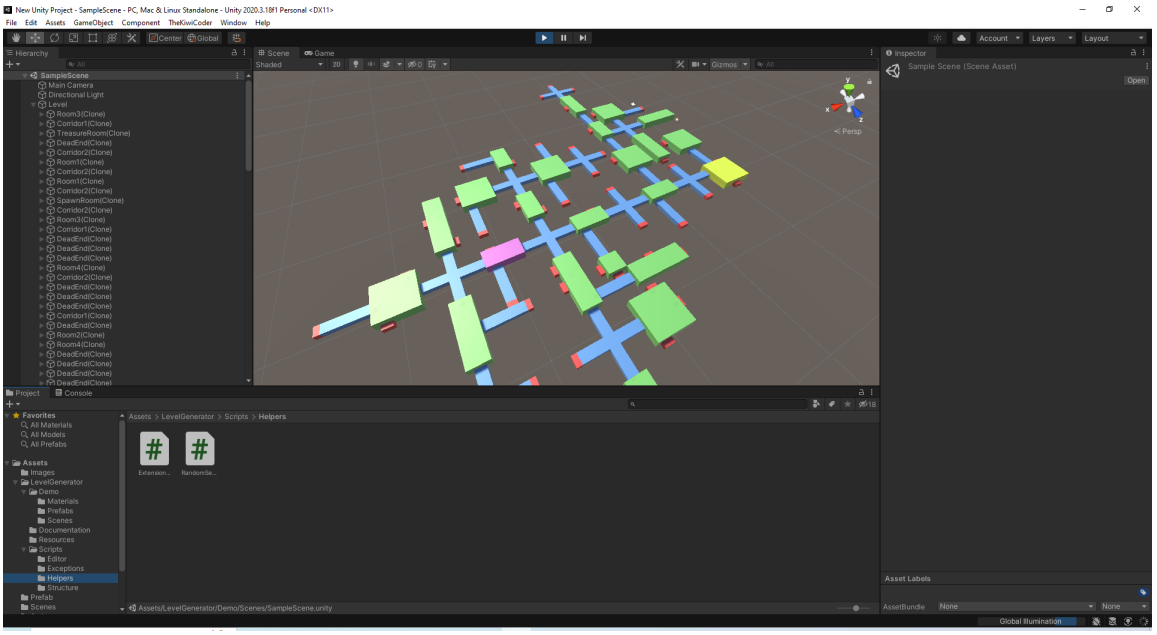


Figura 9: Projecte de el motor Unity.

Figura 10: Logotips Unity (Esquerra) i Visual Studio (Dreta).

4.2.2. Entorns de documentació

Durant tot el projecte s'utilitzarà el Docs de Google per redactar aquesta memòria. L'eina Lucidchart es farà servir per crear els esquemes i les representacions auxiliars necessàries.

Figura 11: Logotips Google Docs (Esquerra) i LucidChart (Dreta).

4.2.3. Entorns de planificació i gestió

Per realitzar el cronograma inicial i mantenir una taula de tasques farem servir el Google Docs, per ser més concrets les fulles de càlcul.

Figura 12: Logotips Google Docs SpreadSheet.

5. Disseny del videojoc

5.1 Mecàniques

5.1.1 L'espai del joc

En voler utilitzar tècniques de generació procedural per tal de crear els diferents nivells del joc, l'espai mai és el mateix (Tot i que les sales seran les mateixes, la disposició d'aquestes no ho serà). Disposem de diversos tipus de sales o elements que conformaran els nivells, però les podem separar en 3 grups:

Sales: Seran els elements més presents al joc i on apareixeran la majoria d'enemics. Les podrem trobar en diferents formes i mides, però comparteixen el mateix concepte: un espai on el jugador podrà lluitar contra els diversos enemics. Les sales tindran diversos punts o zones que permetran que s'expandeixi el nivell, és a dir segons l'algoritme apareixerà un passadís (el qual ens connectarà a una altra sala) o un camí sense sortida que delimitarà el final de l'expansió.

També disposarem de dues sales especials a més a més de les sales normals, una la qual serà la sala on apareixerà el boss i la segona serà on spawnejara el jugador a l'inici del joc.

Passadissos: Els passadissos seran els elements que uniran les diferents sales i ens permetran viatjar entre elles. És possible que apareguin enemics en aquestes zones, però seran més baixes i en un número més baix

Dead End: Aquest element serà simplement per delimitar l'expansió del nivell, és a dir aniran apareixent segons un nombre introduït que representarà la probabilitat a parar la generació.

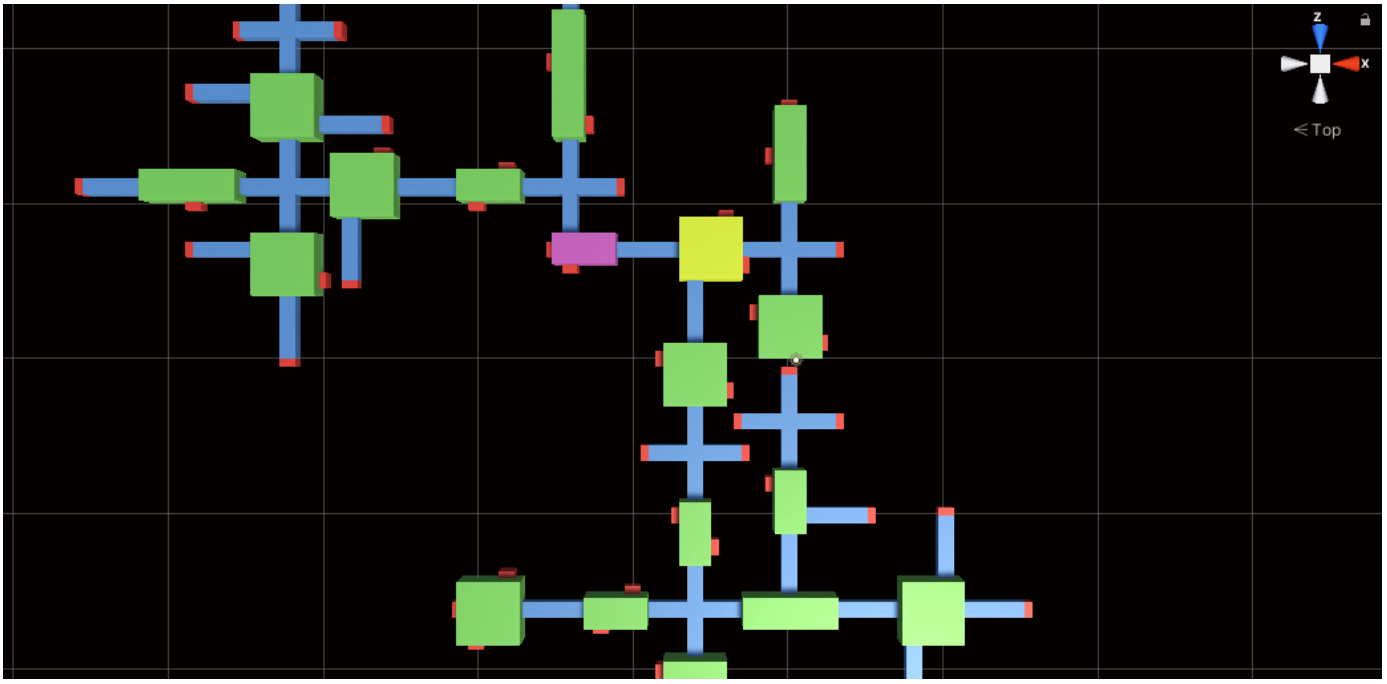


Figura 13: Distribució generada de nivell

5.1.2 Mecàniques del joc, reptes que ha d'afrontar el jugador i accions possibles

En els Rogue Like els principals elements són la mort permanent i la generació procedural dels nivells, els quals són elements ja programats que el jugador no podrà modificar, per tant, es veurà obligat a superar hi ha unes mecàniques predefinides, la qual cosa fa que la mecànica en la qual ens enfocarem més seran els dispars.

Els elements de la mort permanent i la generació procedural de nivells van juntes, ja que es complementen una a l'altra, és a dir la mort permanent no només implica perdre el progrés actual sinó que també implica que cada vegada que el jugador mori, desconeixerà el nou nivell al qual s'enfrontarà. Per tant, no podrà preveure a què s'enfrontarà així incrementant la dificultat i rejugabilitat del joc.

Pel que fa a la mecànica dels dispars serà una mecànica simple de dispars amb vàries armes i munició, així obligant el jugador a gestionar els recursos.

5.1.3 Objectes, recursos i interaccions que pot fer el jugador

En els videojocs roguelike, els objectes solen tenir un pes molt més important en referència a altres classes de jocs, ja que, com que el jugador comença cada partida en un nivell diferent, aquests solen ser els que defineixen el tipus d'estil de joc que el jugador seguirà en la partida. Els principals objectes que el jugador tindrà o anirà trobant a mesura que avança en el nivell seran els següents:

Pistola: Dispara projectils d'un en un (o també anomenat de manera semi automàtica). Aquesta arma intenta combinar una cadència de foc i dany per bala per aconseguir una arma balancejada.

Rifle: Dispara múltiples projectils. És l'arma amb una cadència de foc més alta.

Caixa de munició: Proporciona munició per una arma aleatòria al jugador.

Farmaciola: Proporciona un valor de vida fixe al jugador.

5.1.4 Economia interna del joc

No hi ha economia interna dins el joc. L'únic que el jugador obtindrà serà experiència en combatre els diferents enemics, el qual li facilitarà futures partides.

5.1.5 Estudiar els diferents nivells del joc.

Per aquest joc s'ha decidit fer una construcció de nivells procedural, com ja s'ha explicat a l'apartat 5.1.1. Pel que fa als components d'aquests els tornarem a dividir en 3 grups i els explorarem.

Sales: Les sales constaran de diferents connexions que permetran enllaçar amb altres sales o passadissos. A la vegada vindran decorades per diferents objectes. Aquestes seran de diferents mides, tipus i formes.

Passadissos: Els passadissos seran els encarregats d'unir les diferents sales i seccions, n'hi haurà dos tipus: els passadissos rectes i les interseccions.

Dead End: Al ser el camí sense sortida un final, simplement sera una paret amb una textura per tant no hi ha molt a descriure.

5.1.6 Estudiar les interfícies

Les interfícies del projecte seran limitades. Al ser un joc d'acció i trets el jugador ha de poder veure clarament la pantalla sense tenir distraccions o perdre visió. Els elements que hem decidit incorporar a la interfície seran els més essencials per al jugador, és a dir la salut, l'estamina i la munició.

La salut i l'estamina la representarem amb barres que aniran incrementant o decreixent segons la quantitat, per donar un feedback visual senzill al jugador.

Per al que fa a la munició, es mostrarà la munició restant en el carregador i la total juntament amb un sprite de l'arma actual.

A l'inici del projecte teníem la idea d'implementar la vida restant de l'enemic sobre d'aquests per ser visibles per el jugador, però al final es va decidir que seria més interessant que el jugador no ho sapiguès per proporcionar una sensació d'inseguretat davant dels enemics.

5.2 Estudi i disseny de personatges

5.2.1 Pes narratiu dels personatges

El nostre projecte no disposa d'una trama narrativa. Per tant els diferents personatges que apareixen en el projecte, els quals són principalment enemics, no tenen cap mena d'impacte.

5.2.2 Els personatges com a base de la jugabilitat

El projecte no s'enfoca en aspectes artístics, la majoria de personatges tindran el mateix o un disseny semblant.

Els enemics els podem assignar les següents categories:

Enemics cos a cos: Van vestits amb roba blanca tacada de sang i amb un ganivet. Persegueixen a l'enemic, ja que per fer-li mal necessiten tocar-lo.

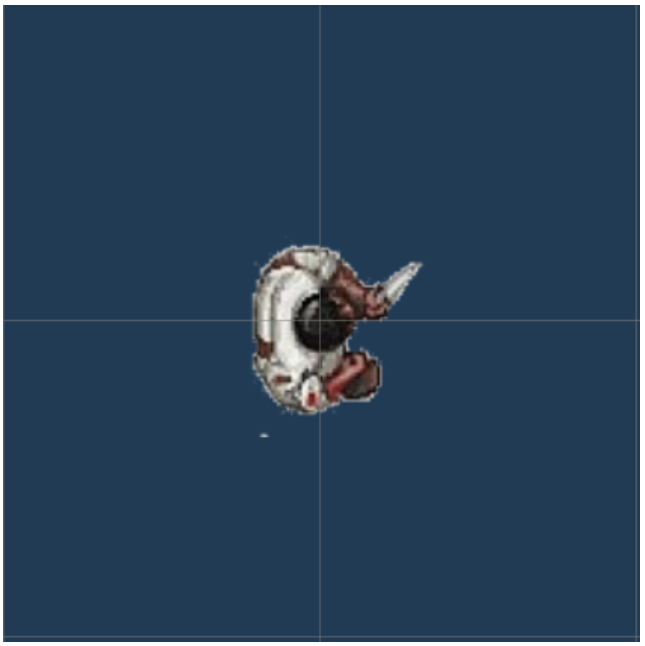


Figura 14: Enemic que ataca cos a cos

Enemics que ataquen a rang: Porten roba de color verd i van armats amb una escopeta. Atacarà a distància a causa de l'arma que porta.

Figura 15: Enemic que ataca a distància

Boss: Tenen elements característics que fan que siguin fàcils de diferenciar de la resta d'enemics. Combinarà els diferents atacs de l'enemic i els farà més perillosos.

Figura 16: Enemic de tipus boss

Personatge principal: El personatge principal és relativament fàcil de reconèixer en ser l'únic personatge que porta el color blau. El protagonista portarà una sèrie d'armes que es podran anar canviant al llarg de la partida.

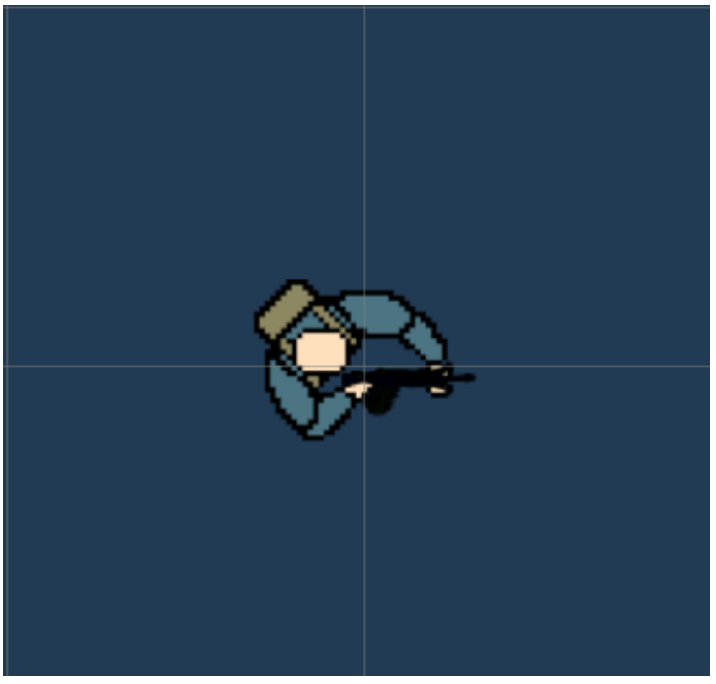


Figura 17: Personatge principal

5.2.3 Estil artístic

L'estil artístic utilitzat es caracteritza per diversos aspectes: el primer és la decisió sobre la vista que ha de tenir del jugador sobre el qual passa dins el joc. Des del principi del desenvolupament volia fer servir un estil de vista "Top Down" (de dalt cap a baix), com el de Hotline Miami, perquè el jugador pugui observar el que passa al seu voltant, però al ser el món del joc en un edifici també busquem reduir el camp de visió del jugador per donar més emoció a la partida així donant més immersió.

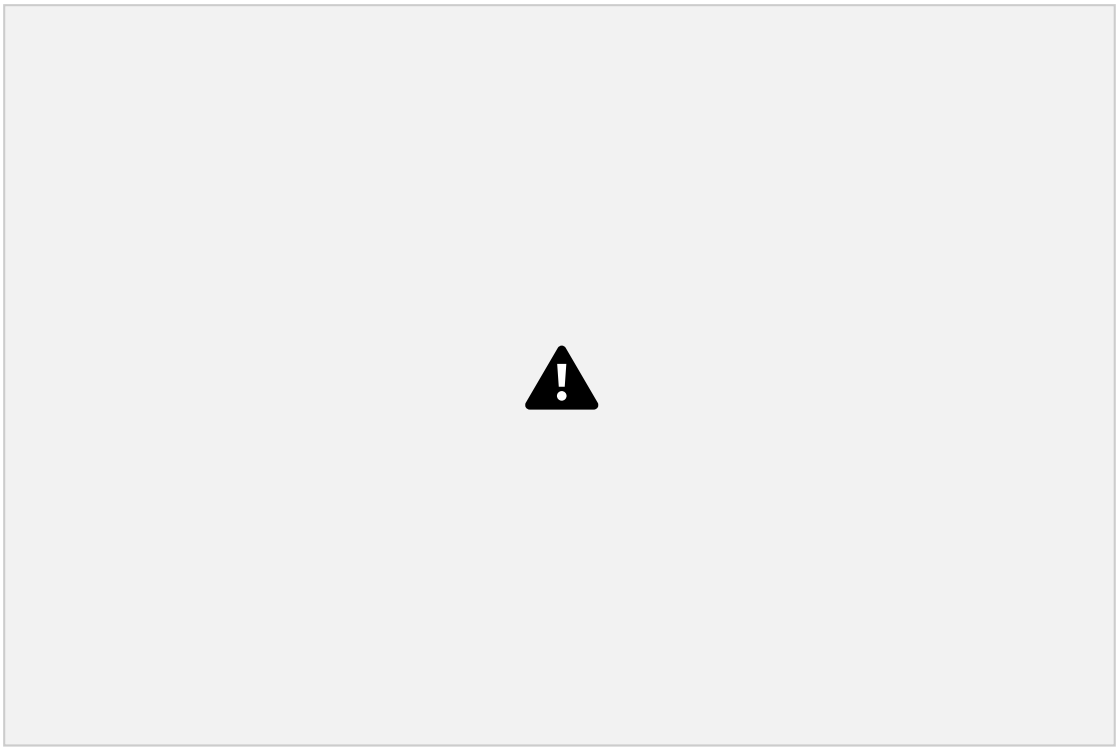


Figura 18: Hotline Miami

Respecte a els gràfics, s'ha optat per gràfics 2D, i un estil combinat entre "Pixel Art" i ús d'sprites per generar un estil "Retrowave". Aquest tipus d'estil és simple, a través d'aquest el jugador veu el que se li acosta.

5.2.4 Objectes que caracteritzen els personatges

Com hem vist a les figures 17, 16, 15 i 14 els nostres personatges es caracteritzaran per les seves armes. L'enemic cos a cos anirà amb un gabinet, l'enemic a distància anirà amb una arma, el boss combinarà els dos tipus d'armes i finalment el jugador tindrà 3 armes de foc.

5.3 Narrativa

5.3.1 Notes importants

Cal remarcar que en el projecte la narrativa es casi nula, per tant no s'hi ha dedicat molt temps.

5.3.2 Sinopsi

El personatge principal és un detectiu d'incògnit, el qual s'ha infiltrat en una instal·lació de la màfia per trobar al misteriós Sr. Hunfferd, el nou cap de la màfia, i descobrir la seva identitat. Però abans d'entrar a la instal·lació li arriba un missatge d'un altre infiltrat informant-lo que ha estat descobert. El protagonista decideix continuar i descobrir la identitat del Sr.Hunffred per la força.

5.4 Mons de joc

5.4.1 Dimensió física

L'espai físic variarà segons la sala on ens trobem, ja que cadascuna d'aquestes té un propòsit. En les sales normals el jugador disposarà d'espai suficient per afrontar els enemics, podent així maniobrar i no haver de preocupar-se massa per quedar-se sense espai de maniobra. Pel que fa a la sala del boss serà una mica més gran que les sales normals. També fa servir zones més estretes que connecten les diferents sales, és a dir, els passadissos. Aquests seran més estrets, però al no aparèixer o aparèixer un número molt reduït d'enemics, el jugador no es tindrà que preocupar massa. També cal mencionar que els enemics no seguiran al jugador de sala en sala sinó que es quedaran en les seves respectives sales, així donant al jugador més opcions per afrontar els reptes que li presentem.

El nivell no és gaire grans així obtenim una exploració del nivell més o menys ràpida. Tot i que es pot modificar l'algorisme per a generar un nivell més gran, al final hem decidit que no seria del tot idoni.

5.4.2 Dimensió temporal

La dimensió temporal tindrà rellevància en l'ambientació. Per tant, el temps on es localitza el joc és molt important, en definir part de l'estetica, com ja hem descrit a

l'apartat 5.2.3. Per al que fa al temps que passa dins del joc, no li donem importància, per tant, no el tenim en compte.

5.4.3 Dimensió ambiental

El joc tindrà una temàtica unificada. Al desenvolupar-se en un lloc físic unificat, compartirà la mateixa dimensió ambiental relacionada com hem dit a l'apartat 5.2.3:

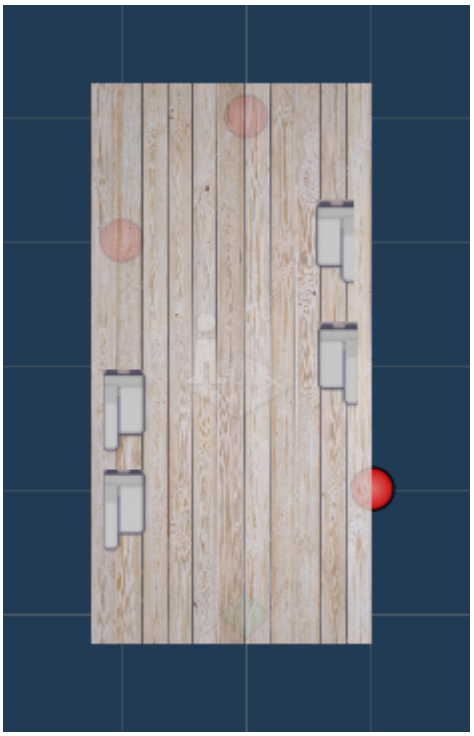


Figura 19: Sala 1

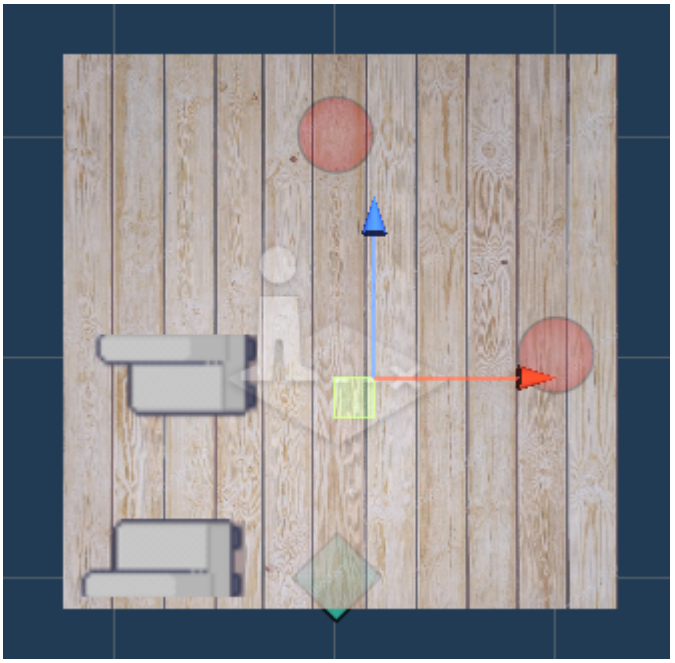


Figura 20: Sala 2

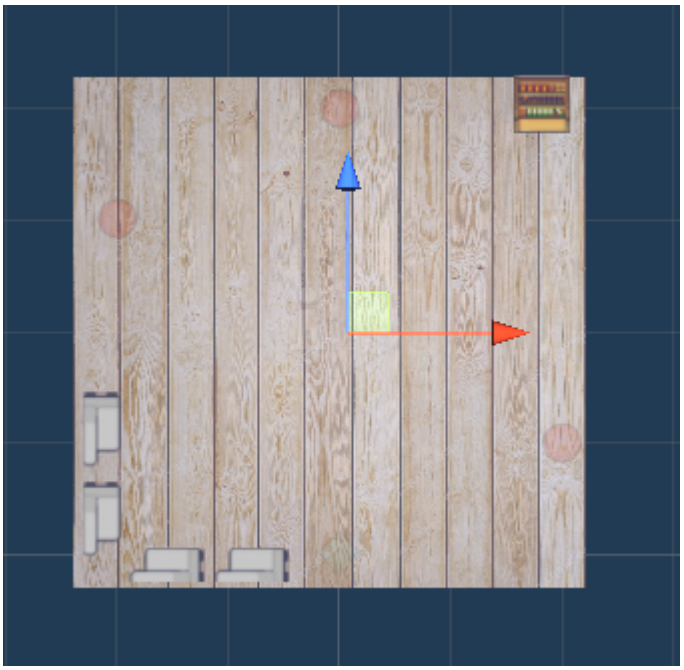


Figura 21: Sala 3

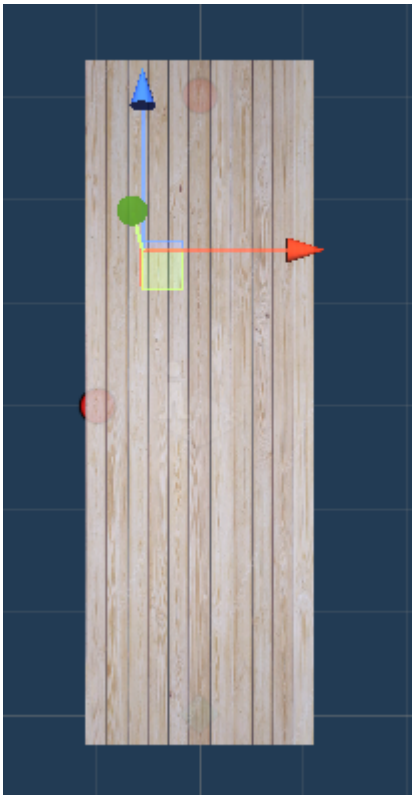


Figura 22: Sala 4

Figura 23: Sala Spawn

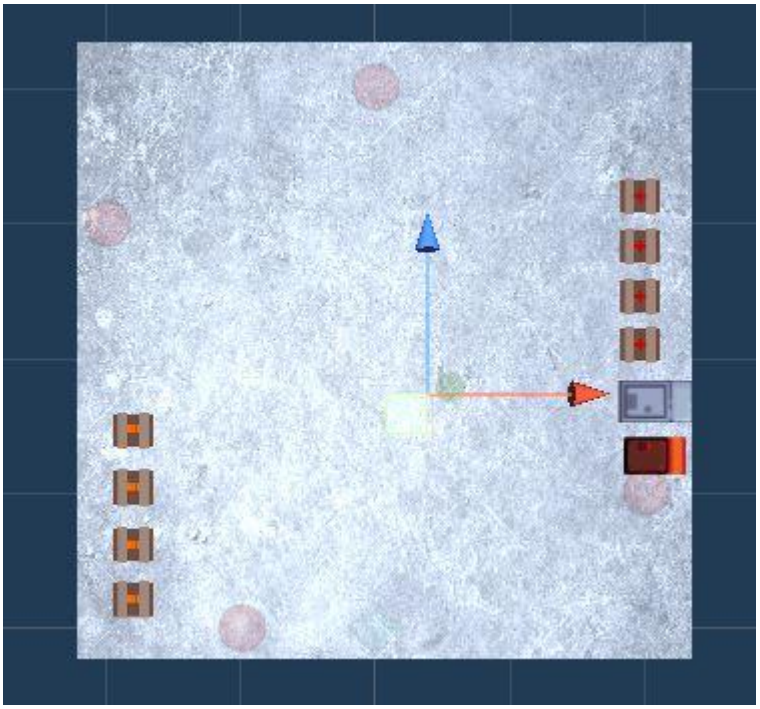


Figura 24: Sala Boss

5.4.4 Referents estètics

En aquest projecte hem seguit dos principals referents estètics, Nuclear Thorne i Hotline Miami, sobretot en com es veu el mapa, algunes interfícies, els enemics i les animacions d'aquests, veure figures 25 i 26.

Figura 25: Nuclear Thorne.

Figura 26: Hotline Miami.

5.4.5 Dimensió emocional: Influir en les emocions del jugador.

El joc no disposa de gaires components emocionals. El motiu principal pel qual el personatge principal es juga la vida és a causa de la seva feina i la voluntat de derrotar a la màfia. Aquesta missió ha estat assignada, però no sabem què en pensa el protagonista, simplement decideix acceptar-la i continuar endavant. Quan derrotar al cap de la màfia, el personatge se sentirà feliç i realitzat, ja que ha derrotat a la màfia.

5.4.6 Aspectes ètics

No hi ha aspectes ètics al projecte com a tal. L'únic problema que podríem plantejar és si és correcte per al protagonista entrar a l'edifici de la màfia, ja que al saber que serà atacat quan entri, no es podria justificar defensa pròpia i a l'entrar provocarà una confrontació que deixarà diversos morts. Però en aquesta classe de jocs no sol ser un aspecte important, ja que aquesta acció és la que desenvolupa el joc.

5.5 Interfícies

Les interfícies que el jugador podrà veure seran els menús i, per altra banda, les interfícies de joc, les quals inclouen vida, stamina, munició i altres indicadors visuals, les quals es distribuiran per la pantalla per tal d'evitar sobrecarregar la vista del jugador.

Pels textos del menú hem escollit la següent tipografia anomenada DM-80 la qual està formada per una sèrie de punts per formar lletres i números. Veure figures 27 i 28.

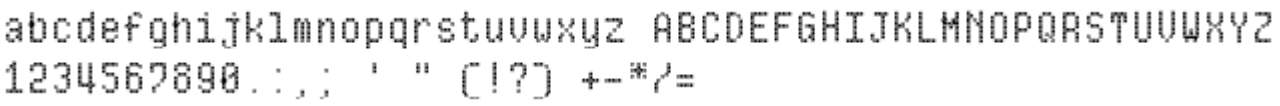


Figura 27: Tipografia DM-80.

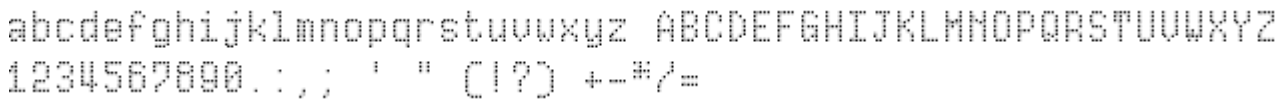


Figura 28: Tipografia DM-80 light.

5.6 Game Layout Charts

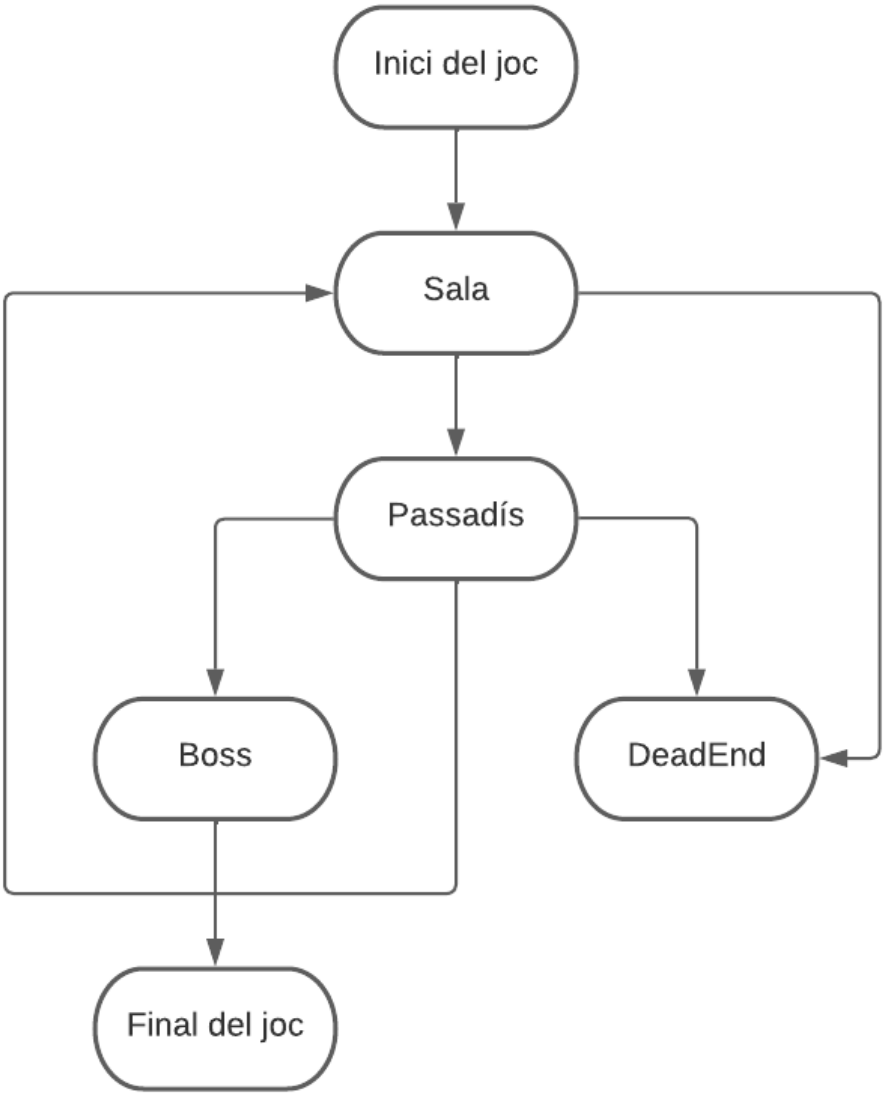


Figura 29: Game Layout Chart.

L'algorisme de disseny procedural del nivell està pensat per a una estructura de joc adaptable, és a dir, permet modificar la possibilitat d'aparició dels diferents elements que conformen el nivell i la quantitat que es generarà. Amb aquests tipus d'algorismes es busca simplificar la generació de nivells i crear diversitat, per això en aquest projecte hem buscat utilitzar un algorisme d'aquest tipus per facilitar la generació d'aquests i simplificar la creació de nivells.

5.7 Producció externa

En el projecte hem utilitzat diferents elements externs que no hem produït. Els elements són els seguents:

PixelMe: És una eina que permet transformar qualsevol imatge a un estil pixela art. És una eina desenvolupada per l'usuari de Twitter @sato_neet.

Procedural Level Generator: És una eina que permet fer servir un algorisme de generació procedural personalitzable, aixi donant la llibertat a l'usuari que l'utilitzi de personalitzar el resultat. És una eina desenvolupada per l'usuari Juan Rodriguez a la plana Unity Asset Store.

Behaviour Tree by TheKiwiCoder: És una eina que ens permet crear Behaviour Trees i ens proporciona components necessaris per operar aquests. És una eina desenvolupada per l'úsuari TheKiwiCoder a Youtube.

TOPDOWNPACK: Paquet d'sprites de diversos tipus utilitzats per generar mobiliari. Creat per l'usuari HIS a la web itch.io.

Modern Interiors: Paquet d'sprites de diversos tipus utilitzats per generar mobiliari. Creat per l'usuari LimeZu a la web itch.io.

Top Down Gun Pack: Paquet d'sprites de diversos tipus utilitzats per generar armes. Creat per l'usuari that games guy a la Unity Asset Store.

5.8 Elements de feedback

Per als diferents textos utilitzats en les interfícies hem utilitzat Unity Text juntament amb una font de text externa com ja hem explicat a l'Apartat 5.5. A l'Apartat 6.3 es detalla la implementació de totes les interfícies d'aquest apartat.

5.8.1 Barra de vida

L'indicador de salut del personatge consisteix en una barra de vida que anirà incrementant o disminuint segons la vida restant, la barra està situada a la part esquerra superior de la pantalla.

5.8.2 Barra de stamina

L'indicador de stamina del personatge consisteix en una barra de stamina que anirà incrementant o disminuint segons la stamina restant, la barra està situada a la part esquerra superior de la pantalla.

5.8.3 Comptador de munició

El comptador de munició constarà d'un text que s'anirà actualitzant a mesura que el jugador utilitzi munició. El comptador mostrarà les bales restants al carregador i les totals. El comptador està situat a la part dreta superior de la pantalla.

5.8.4 Icona de l'arma

La icona de l'arma constarà d'un sprite que anirà canviant segons l'arma seleccionada, la icona està situada a la part dreta superior de la pantalla.

5.8.5 Menú de pausa

Quan el jugador prem el botó de pausa (la tecla ESC), el joc s'atura i s'obre un menú d'opcions on es podrà triar entre reprendre la partida o tornar al menú principal.

6. Implementació i proves

6.1 Estructura de les escenes

El joc està format per 4 escenes, veure la figura 30.

menu: És la primera escena, i consisteix en el menú del joc.

win: És l'escena que es mostra al guanyar en el joc**.**

gameOver: És l'escena que es mostra al perdre en el joc**.**

SampleScene: És l'escena principal del joc, on concentra tot el contingut. Dins aquesta escena es gestiona la creació del nivell

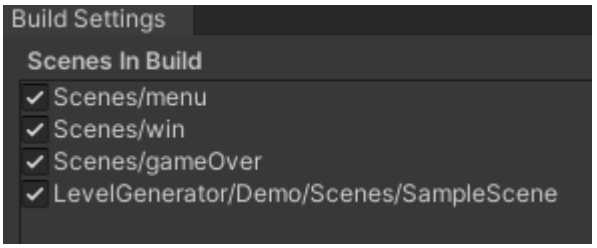


Figura 30: Estructura d'escenes.

6.2 Generació procedural del nivell

Com s'ha mencionat diverses vegades al llarg del document el
nivell del joc serà generat proceduralment. Això comporta que cap nivell serà igual a l'anterior. El codi està dividit en sis scripts diferents:

Level Generator: Serà l'script encarregat de generar les dades necessàries per muntar el nivell a través dels diversos scripts i prefabs que formaran part del món.

Section: Serà l'script encarregat de generar les diferents sales i generar les seves connexions i expandir el nivell.

Exits: Indicarà per on l'algorisme es pot expandir.

DeadEnd: Crea una zona sense sortida que evitarà que l'algorisme expandeixi el nivell.

Bounds: Serveix per delimitar la zona de cada sala i evitar col·lisions.

AdvancedExit: Serveix per forçar a l'algorisme a generar sales adjacents a altres. Per exemple obliguem que al costat de la sala spawn es generi un DeadEnd.

6.2.1 "Prefabs" de la generació procedural.

Com hem vist a l'apartat 5.4.3 els "prefabs" són simples paquets amb un element ja generat que en permetran instanciar-los segons requereixi l'algorisme de manera més ràpida. En aquest cas l'ús té els seus avantatges i desavantatges. Ens permet una

construcció més modular i ens permet modificar com quedarà el mapa de manera més fàcil i senzilla, però també limita l'aleatorietat a un predefinit d'opcions.

6.3 Implementació d'enemics

Com hem mencionat a altres apartats es va decidir utilitzar els Behaviour Trees. Aquests ofereixen una sèrie d'accions bàsiques com serien repetir accions, fer una sèrie d'accions en seqüència o en paral·lel, ... En el nostre projecte el més important són les accions, ja que seran el que guiarà a la IA enemiga.

6.3.1 Behaviour Tree enemic cos a cos

El behaviour Tree anomenat EnemyCQC és el que controla els enemics cos a cos, veure figura 31.

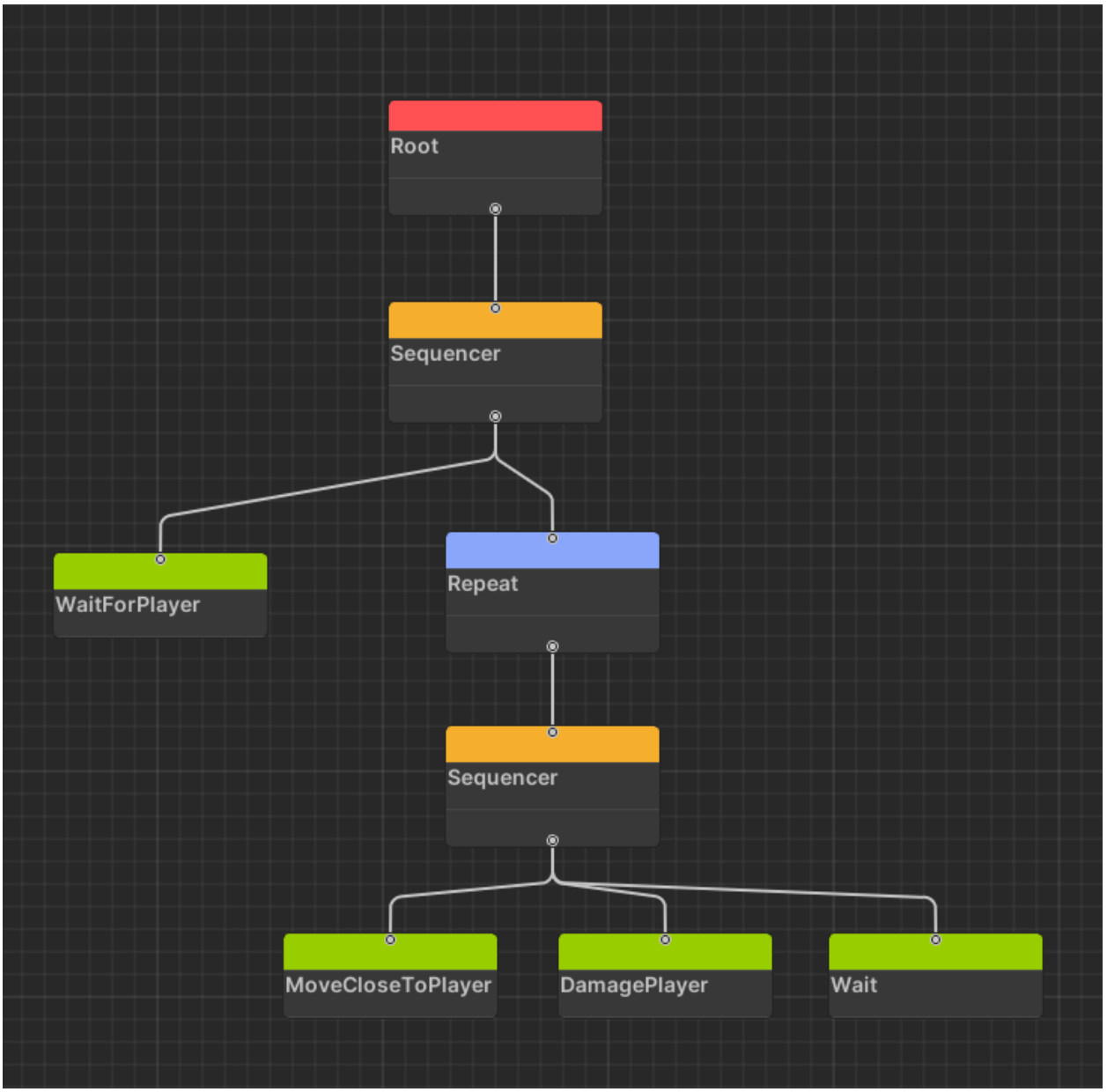


Figura 31: Behaviour Tree Enemy CQC.

En aquest abre podem veure 4 accions o scripts:

WaitForPlayer: Utilitzant l'eina NavMeshAgent, que ens permet automatitzar el moviment de la IA simplement amb un objectiu, esperarem que el jugador entri en el rang de l'enemic.

```
public class WaitForPlayer : ActionNode
{
    public float tolerance = 10;

    protected override void OnStart()
    {
        //context.agent.boxCollider
    }

    protected override void OnStop()
    {
    }

    protected override State OnUpdate()
    {
        if (context.agent.pathPending)
        {
            return State.Running;
        }

        if (Vector2.Distance(blackboard.Player.transform.position, context.agent.transform.position) < tolerance)
        {
            return State.Success;
        }

        if (context.agent.pathStatus == UnityEngine.AI.NavMeshPathStatus.PathInvalid)
        {
            return State.Failure;
        }

        return State.Running;
    }
}
```

Figura 32: WaitForPlayer.cs.

MoveCloseToPlayer: Aquest script farà que la IA intenti acostar-se al jugador i una vegada prou a prop passarà a l'acció d'atacar.

```
public class MoveCloseToPlayer : ActionNode
{
    public float speed = 5;
    public float stoppingDistance = 1f;
    public bool updateRotation = true;
    public float acceleration = 40.0f;
    public float tolerance = 1f;

    private GameObject playerinfo;
    private float damp = 6.0f;

    protected override void OnStart()
    {
        //context.agent.stoppingDistance = stoppingDistance;
        context.agent.stoppingDistance = stoppingDistance;
        context.agent.speed = speed;
        //context.agent.destination = blackboard.Player.transform.position;
        //context.agent.updateRotation = updateRotation;
        context.agent.acceleration = acceleration;
        //tolerance = blackboard.chargeDistanceInicialize;

        playerinfo = GameObject.FindGameObjectWithTag("Player").transform.GetChild(0).gameObject;

    }

    protected override void OnStop()
    {
    }

    protected override State OnUpdate()
    {

        //Look at enemy
        Vector3 direction = playerinfo.transform.position - context.agent.transform.position;
        float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;
        Quaternion rotate = Quaternion.Euler(0, 0, angle);
        context.agent.gameObject.GetComponent<Rigidbody>().rotation = Quaternion.Lerp(context.agent.gameObject.GetComponent<Rigidbody>().rotation, rotate, 5 * Time.fixedDeltaTime);

        context.agent.destination = playerinfo.transform.position;

        if (context.agent.pathPending)
        {
            return State.Running;
        }

        if (Vector2.Distance(playerinfo.transform.position, context.agent.transform.position) < tolerance)
        {
            //Stop movement
            context.agent.destination = context.agent.transform.position;
            return State.Success;
        }

        if (context.agent.pathStatus == UnityEngine.AI.NavMeshPathStatus.PathInvalid)
        {

            return State.Failure;
        }

        return State.Running;
    }
}
```

Figura 33: MoveCloseToPlayer.cs.

DamagePlayer: Aquest script farà l'acció de atacar al jugador.

```
public class DamagePlayer : ActionNode
{
    public int damage = 10;
    // Start is called before the first frame update
    protected override void OnStart()
    {
        GameObject.FindGameObjectWithTag("Player").transform.GetChild(0).gameObject.GetComponent<PlayerMovement>().DamagePlayer(damage);
    }

    protected override void OnStop()
    {
    }

    // Update is called once per frame
    protected override State OnUpdate()
    {
        return State.Success;
    }
}
```

Figura 34: DamagePlayer.cs.

Wait: L'script Wait actua com una pausa en l'execucció.

6.3.2 Behaviour Tree enemic a distancia

El behaviour Tree anomenat EnemyRange és el que controla els enemics a distancia, veure figura 35.

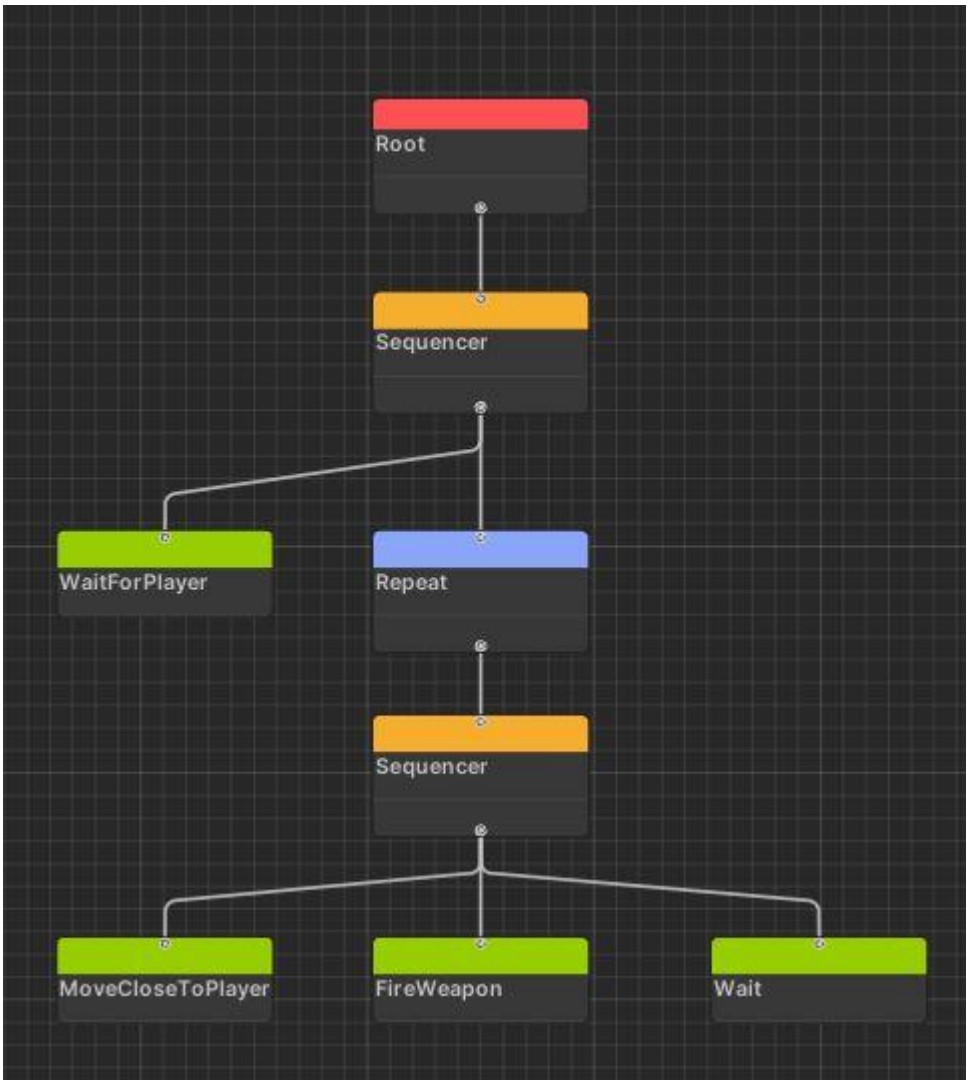
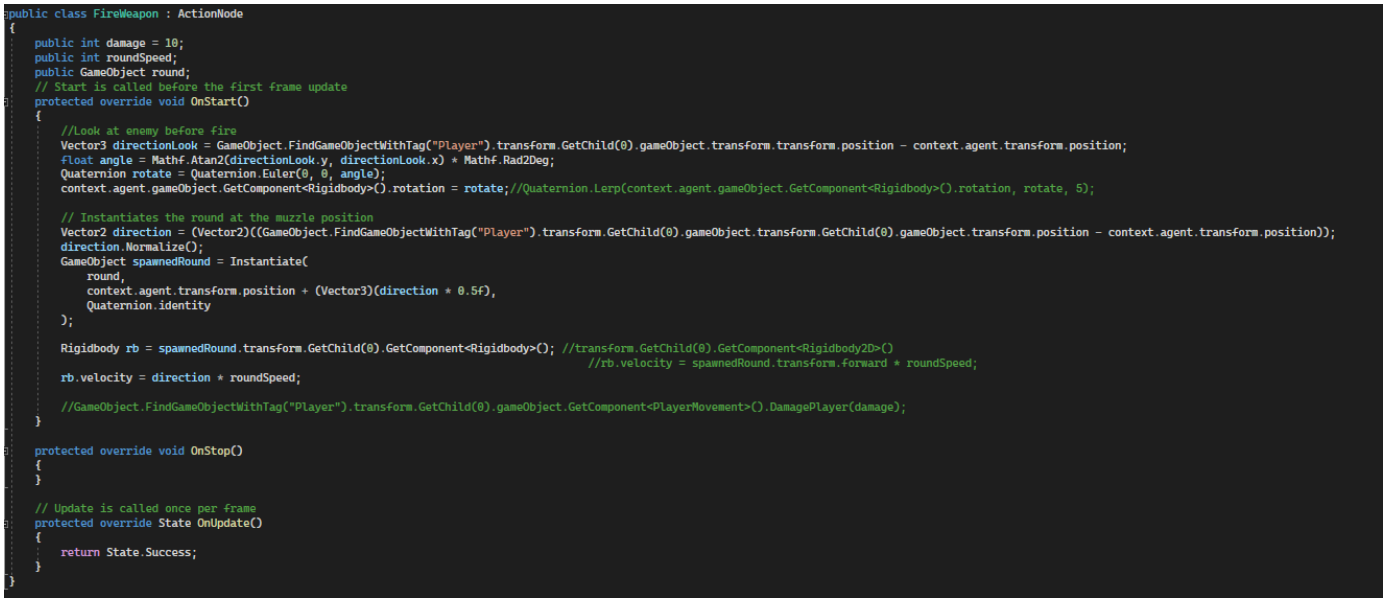


Figura 35: Behaviour Tree Enemy Range.

FireWeapon: Aquest script ens permet disparar al jugador



```
public class FireWeapon : ActionNode
{
    public int damage = 10;
    public int roundSpeed;
    public GameObject round;
    // Start is called before the first frame update
    protected override void OnStart()
    {
        //Look at enemy before fire
        Vector3 directionLook = GameObject.FindGameObjectWithTag("Player").transform.GetChild(0).gameObject.transform.transform.position - context.agent.transform.position;
        float angle = Mathf.Atan2(directionLook.y, directionLook.x) * Mathf.Rad2Deg;
        Quaternion rotate = Quaternion.Euler(0, 0, angle);
        context.agent.gameObject.GetComponent<Rigidbody>().rotation = rotate;//Quaternion.Lerp(context.agent.gameObject.GetComponent<Rigidbody>().rotation, rotate, 5);

        // Instantiates the round at the muzzle position
        Vector2 direction = (Vector2)((GameObject.FindGameObjectWithTag("Player").transform.GetChild(0).gameObject.transform.GetChild(0).gameObject.transform.position - context.agent.transform.position));
        direction.Normalize();
        GameObject spawnedRound = Instantiate(
            round,
            context.agent.transform.position + (Vector3)(direction * 0.5f),
            Quaternion.identity
        );

        Rigidbody rb = spawnedRound.transform.GetChild(0).GetComponent<Rigidbody>(); //transform.GetChild(0).GetComponent<Rigidbody2D>()
                                                                                      //rb.velocity = spawnedRound.transform.forward * roundSpeed;
        rb.velocity = direction * roundSpeed;

        //GameObject.FindGameObjectWithTag("Player").transform.GetChild(0).gameObject.GetComponent<PlayerMovement>().DamagePlayer(damage);
    }

    protected override void OnStop()
    {
    }

    // Update is called once per frame
    protected override State OnUpdate()
    {
        return State.Success;
    }
}
```

Figura 35: FireWeapon.cs

6.3.3 Behaviour Tree Boss

El behaviour Tree anomenat EnemyBoss és el que controla al boss, veure figura 36.

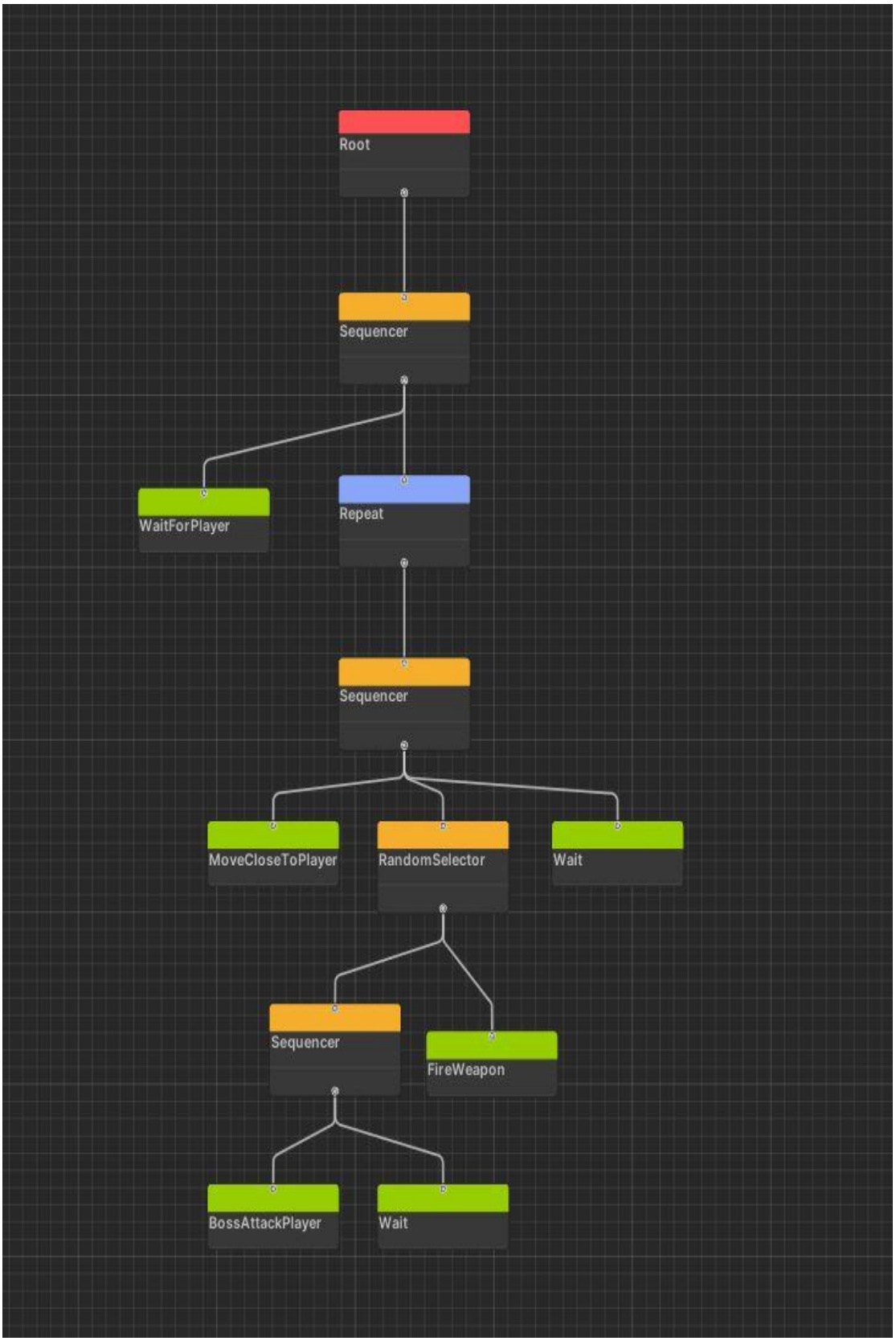


Figura 36: Behaviour Tree EnemyBoss

Com es pot veure a les imatges els Behaviour Trees reutilitzen bastants tasques, cosa que simplifica la feina. En el Behaviour Tree EnemyBoss totes les funcions són utilitzades als altres arbres, l'única que és nova és BossAttackPlayer, però és una variació de MoveCLoseToPlayer. El boss primer s'apropa a certa distància del kugador amb MoveCloseToPlayer, quan està prou a prop l'acció BossAttackPlayer s'activa i simplement carregarà contra l'última posició del jugador coneguda. Així es dona l'oportunitat al jugador de què escapi, finalment també cal saber que a mesura que el boss perd vida la seva velocitat augmenta. Veure figura 37.

```
protected override void OnStart()
{
    health = context.agent.gameObject.GetComponent<EnemyHealth>().health;
    startTime = Time.time;

    //Calculate the speed and aceleration //CANVAIR LA VIDA
    if (health < 100 && health > 75)
    {
        speed = 50;
        acceleration = 40.0f;

    }
    else if (health < 75 && health > 50)
    {
        speed = 75;
        acceleration = 50.0f;
    }
    else if (health < 50 && health > 25)
    {
        speed = 85;
        acceleration = 60.0f;
    }
    else //Less than 25% health
    {
        speed = 100;
        acceleration = 70.0f;
    }

    //context.agent.stoppingDistance = stoppingDistance;
    context.agent.stoppingDistance = stoppingDistance;
    context.agent.speed = speed;
    //context.agent.destination = blackboard.Player.transform.position;
    //context.agent.updateRotation = updateRotation;
    context.agent.acceleration = acceleration;
    //tolerance = blackboard.chargeDistanceInicialize;

    playerinfo = GameObject.FindGameObjectWithTag("Player").transform.GetChild(0).gameObject.transform;
}

protected override void OnStop()
{

}

protected override State OnUpdate()
{
    //Look at enemy

    Vector3 direction = playerinfo.transform.position - context.agent.transform.position;
    float angle = Mathf.Atan2(direction.y, direction.x) * Mathf.Rad2Deg;
    Quaternion rotate = Quaternion.Euler(0, 0, angle);
    context.agent.gameObject.GetComponent<Rigidbody>().rotation = Quaternion.Lerp(context.agent.gameObject.GetComponent<Rigidbody>().rotation, rotate, 5 * Time.fixedDeltaTime);

    context.agent.destination = playerinfo.transform.position;

    Debug.Log(Vector2.Distance(playerinfo.transform.position, context.agent.transform.position));
    Debug.Log(Vector2.Distance(playerinfo.transform.position, context.agent.transform.position) < tolerance);

    if (context.agent.pathPending)
    {
        return State.Running;
    }

    if (Vector2.Distance(playerinfo.transform.position, context.agent.transform.position) < tolerance)
    {
        return State.Success;
    }

    if ((context.agent.pathStatus == UnityEngine.AI.NavMeshPathStatus.PathInvalid) || (Time.time - startTime > duration))
    {
        return State.Failure;
    }

    return State.Running;
}
```

Figura 37: BossAttackPlayer.cs

6.4 Jugador

6.4.1 Moviment del jugador

El sistema de moviment del nostre jugador és relativament senzill, simplement agafarem l'input de les tecles i aplicarem força al Rigidbody del jugador perquè el sistema computi les físiques, veure figura 38.

```
void FixedUpdate()
{
    Vector3 movement = new Vector3(Input.GetAxis("Horizontal"), Input.GetAxis("Vertical"), 0f);
    GetComponent<Rigidbody>().MovePosition(GetComponent<Rigidbody>().position + (movement * Time.deltaTime * moveSpeed));
    //transform.position += movement * Time.deltaTime * moveSpeed;
}
```

Figura 37: PlayerMovement.cs Fixed Update

6.4.2 Sistema de armes

En sistema d'armes es divideix en dos scripts:

ScriptShoot: Aquest script s'encarrega de comunicar el jugador amb l'script PistolaScript, a més a més és l'encarregat de gestionar les diferents armes del jugador i actualitzar el HUD amb la informació pertinent, veure figura 38 i 39.

```
public class ScriptShoot : MonoBehaviour
{
    public PistolaScript gun;
    public PistolaScript rifle;
    public PistolaScript shootgun;

    public Text ammo;

    public GameObject imageOfGun;

    public Sprite gunImage;
    public Sprite rifleImage;
    public Sprite shootgunImage;

    private PistolaScript actualGun;

    public int selectedWeapon = 1; // 1-Gun, 2-Rifle , 3-Shotgun

    void Start()
    {
        deactivateGun();
        activateGun(gun);
        selectedWeapon = 1;
        updateIcon(gunImage);
        updateAmmoText();
    }
```

Figura 38: ScriptShoot.cs

```
void Update()
{
    if (Input.GetKey(KeyCode.Mouse0))
    {
        actualGun.Shoot();
        updateAmmoText();
    }

    if (Input.GetKey(KeyCode.R))
    {
        actualGun.Reload();
    }

    if (actualGun.reloading = true)
    {
        updateAmmoText();
        actualGun.reloading = false;
    }

    //Weapon selection
    if (Input.GetKey(KeyCode.Alpha1))
    {
        changeWeapon(1);
    }
    else if (Input.GetKey(KeyCode.Alpha2))
    {
        changeWeapon(2);
    }
    else if (Input.GetKey(KeyCode.Alpha3))
    {
        changeWeapon(3);
    }
    else if (Input.GetKey(KeyCode.Alpha4))
    {
        changeWeapon(4);
    }
}

void changeWeapon(int weapon)
{
    selectedWeapon = weapon;
    //Cridariam aqui una altra funcio per canviar l'sprite de arma i les altres crides.
    switch (selectedWeapon)
    {
        case 1:
            activateGun(gun);
            updateIcon(gunImage);
            break;
        case 2:
            activateGun(rifle);
            updateIcon(rifleImage);
            break;
        case 3:
            activateGun(shootgun);
            updateIcon(shootgunImage);
            break;
        case 4:

            break;
    }
    updateAmmoText();
}
```

Figura 39: ScriptShoot.cs

PistolaScript: Aquest script funcionarà a través d'una màquina d'estats, veure figura 40. És a dir el script només podrà estar un dels estats que tingui programat quan s'executi. Si el script està en estat de tret, es generarà una instància bala a la direcció on apunti el ratolí. Veure figura 41.

```
void Update()
{
    switch (shootState)
    {
        case ShootState.Shooting:
            // If the gun is ready to shoot again...
            if (Time.time > nextShootTime)
            {
                shootState = ShootState.Ready;
            }
            break;
        case ShootState.Reloading:
            // If the gun has finished reloading...
            if (Time.time > nextShootTime)
            {
                playAudio.clip = reload;
                playAudio.Play();
                if (totalAmmo > ammunition)
                {
                    totalAmmo = totalAmmo - (ammunition - remainingAmmunition);
                    remainingAmmunition = ammunition;
                    //totalAmmo -= ammunition;
                    shootState = ShootState.Ready;
                    reloading = true;
                }
                else if (totalAmmo < ammunition)
                {
                    remainingAmmunition = totalAmmo;
                    totalAmmo = 0;
                    shootState = ShootState.Ready;
                    reloading = true;
                }
            }
            break;
    }
}
```

Figura 40: PistolaScript.cs màquina d'estats

```
/// Attempts to fire the gun
public void Shoot()
{
    // Checks that the gun is ready to shoot
    if (shootState == ShootState.Ready && (totalAmmo > 0 || remainingAmmunition > 0))
    {
        for (int i = 0; i < roundsPerShot; i++)
        {
            playAudio.clip = shoot;
            playAudio.Play();
            // Instantiates the round at the muzzle position
            Vector3 worldMousePos = cam.ScreenToWorldPoint(Input.mousePosition);
            Vector2 direction = (Vector2)((worldMousePos - transform.position));
            direction.Normalize();
            GameObject spawnedRound = Instantiate(
                round,
                transform.position + (Vector3)(direction * 0.5f),
                Quaternion.identity
            );

            /*
             GameObject bullet = (GameObject)Instantiate(
                                bulletPrefab,
                                transform.position + (Vector3)(direction * 0.5f),
                                Quaternion.identity);
             */

            // Add a random variation to the round's direction
            spawnedRound.transform.Rotate(new Vector3(
                Random.Range(-1f, 1f) * maxRoundVariation,
                Random.Range(-1f, 1f) * maxRoundVariation,
                0
            ));

            Rigidbody rb = spawnedRound.transform.GetChild(0).GetComponent<Rigidbody>(); //transform.GetChild(0).GetComponent<Rigidbody2D>()
            //rb.velocity = spawnedRound.transform.forward * roundSpeed;
            rb.velocity = direction * roundSpeed;
        }

        remainingAmmunition--;
        if (remainingAmmunition > 0)
        {
            nextShootTime = Time.time + (1 / fireRate);
            shootState = ShootState.Shooting;
        }
        else
        {
            playAudio.clip = noAmmo;
            playAudio.Play();
            Reload();
        }
    }
    else
    {
        //playAudio.clip = noAmmo;
        //playAudio.Play();
    }
}
```

Figura 41: PistolaScript.cs

En cas que l'estat sigui recarregant es faran les operacions necessàries per actualitzar la munició restant, veure figura 42.

```
/// Attempts to reload the gun
public void Reload()
{
    // Checks that the gun is ready to be reloaded
    if (shootState == ShootState.Ready)
    {
        nextShootTime = Time.time + reloadTime;
        shootState = ShootState.Reloading;
    }
}
```

Figura 41: PistolaScript.cs

6.4.3 HUD

Hem decidit que en aquest projecte no ens podem permetre un HUD sobrecarregat d'informació, ja que ocuparia molta part de la pantalla del jugador i això repercutiria sobre la jugabilitat. Per tant, hem decidit mantenir un HUD minimalista on només es mostrin les barres de vida, stamina, la munició i l'arma seleccionada, veure figura 42.

Figura 42: HUD

Aquesta informació es mantindrà actualitzada a través de PlayerMovement.cs i ScriptShoot.cs que cridaran diferents funcions que accedeixen als elements del HUD.

Finalment, existeixen 2 objectes que proporcionen vida i munició, veure figura 24 a l'apartat 5.4.3.

6.5 Estètica i so

L'ús de l'estètica i el so ha quedat relegat davant altres prioritats, però també ha tingut importància en aquest projecte. Hem introduït efectes a les armes i altres elements del projecte utilitzant AudioPlayers, també hem afegit música al menú i la pantalla de joc per proporcionar una experiència més satisfactòria al jugador. Com hem comentat en apartats anteriors l'estètica recau sobretot en un estil Retrowave i PixelArt, per conseqüència no hi han hagut massa complicacions a l'hora d'implementar-ho.

6.6 Menús

En l'apartat dels menús seré breu, disposem d'un menú inicial, un de pausa el qual para el joc a través de l'ús de timeScale i un per victòria o derrota. Tots estan interconnectats permeten així continuar la partida una vegada mort o havent guanyat.

6.7 Proves realitzades

Durant el desenvolupament del projecte s'han anat realitzant diverses proves a mesura que es desenvolupava i afegia nou contingut per comprovar el correcte funcionament. Durant aquest procés de desenvolupament s'ha anat provant el joc i canviat o millorat diferents elements segons a com responia el joc. Aquestes millores inclouen, però no es limiten a les mecàniques de trets, HUD, la creació del nivell.

6.7.1 Problemes actuals

Finalment comentar que hi han problemes amb els assets ja que a vegades no es comporten de forma correcta.

7. Resultats

7.1 Legislació i normativa vigent

Pel que fa als problemes de Copyright, quasi tots els recursos utilitzats en el projecte han estat creats per nosaltres, o són de lliure ús. En el cas dels recursos que no són de lliure ús només caldria substituir-los en cas de treure el joc al mercat o contactar amb els propietaris per arribar a un acord econòmic pel seu ús. A més a més, si la venda del joc fos un èxit i superéssim la xifra de 100.000$ en ingressos bruts, definit pels propietaris de Unity, ens veuríem obligats a adquirir un altre tipus de llicència que ens permet elevar aquest límit.

El joc desenvolupat no presenta cap problema en aspectes legislatius. En cap moment es desa informació de caràcter personal del jugador, per tant, no s'aplica en cap situació la LOPD (Llei Orgànica de Protecció de Dades). Tampoc apliquem la LSSICE (Llei de Serveis de la Societat de la Informació i Comerç Electrònic), ja que el projecte no constitueix cap activitat econòmica.

7.2 PEGI

El Sistema PEGI (Pan European Game Information) és el mecanisme d'autoregulació dissenyat per la indústria per dotar els productes d'informació orientativa sobre l'edat adequada per al consum. Aquest sistema està integrat per dos tipus d'icones descriptores, una relativa a l'edat recomanada i una altra al contingut específic susceptible d'anàlisi. El disseny dels logotips informatius es basa en els llums de seguretat viària, fent-ne més fàcil i visual la interpretació. Les següents figures representen els diferents elements de classificació PEGI:

Figura 43: Etiquetes PEGI edats recomanades.

Figura 44 : Etiquetes PEGI classificació d'elements del joc.

En el nostre cas, al ser un joc basat en l'eliminació d'enemics, hauria de dur l'etiqueta de violència. A més a més, al ser els enemics humans, tindria que tenir la qualificació +18, com Hotline Miami.

7.3 Resultat final

En aquest apartat mostrarem captures de diferents moments del joc per a veure el resultat final del projecte.

Figura 45 : Menú

Figura 46 : Captura Joc

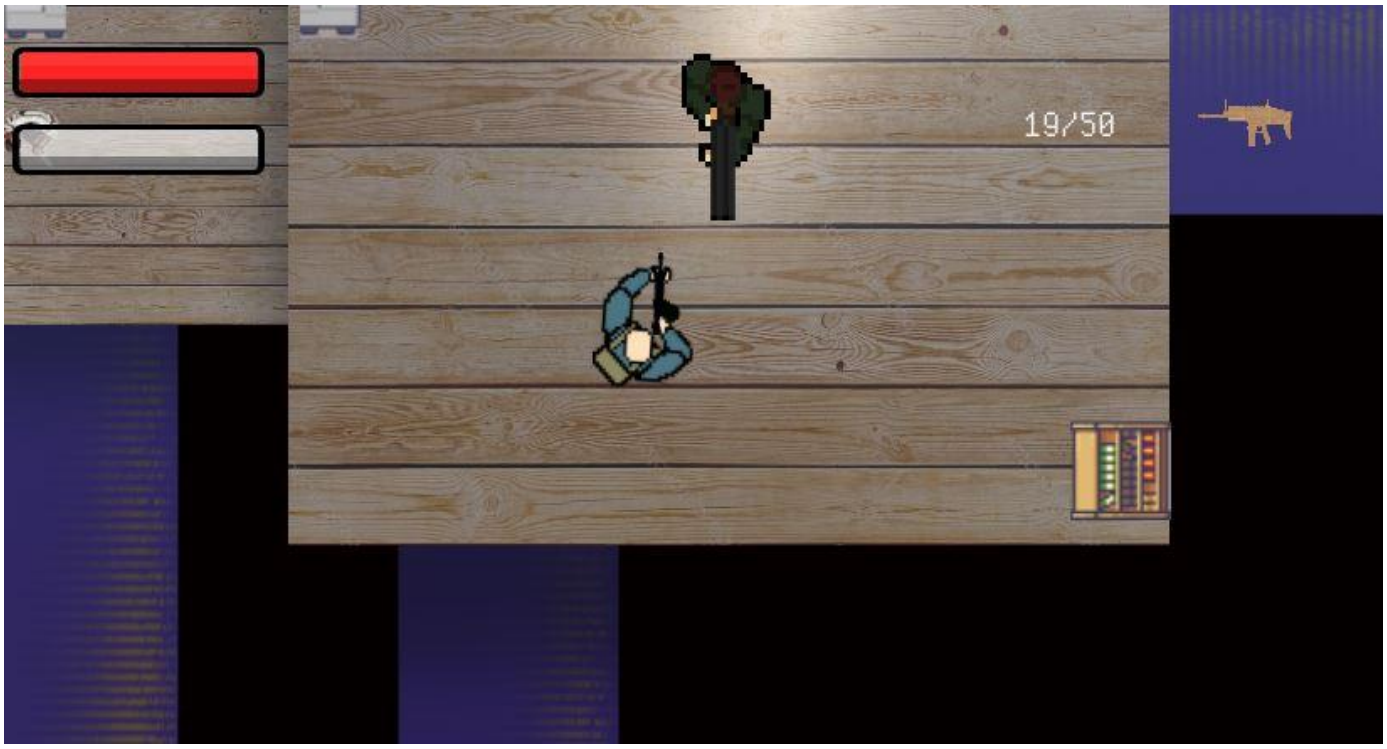


Figura 47 : Captura Joc

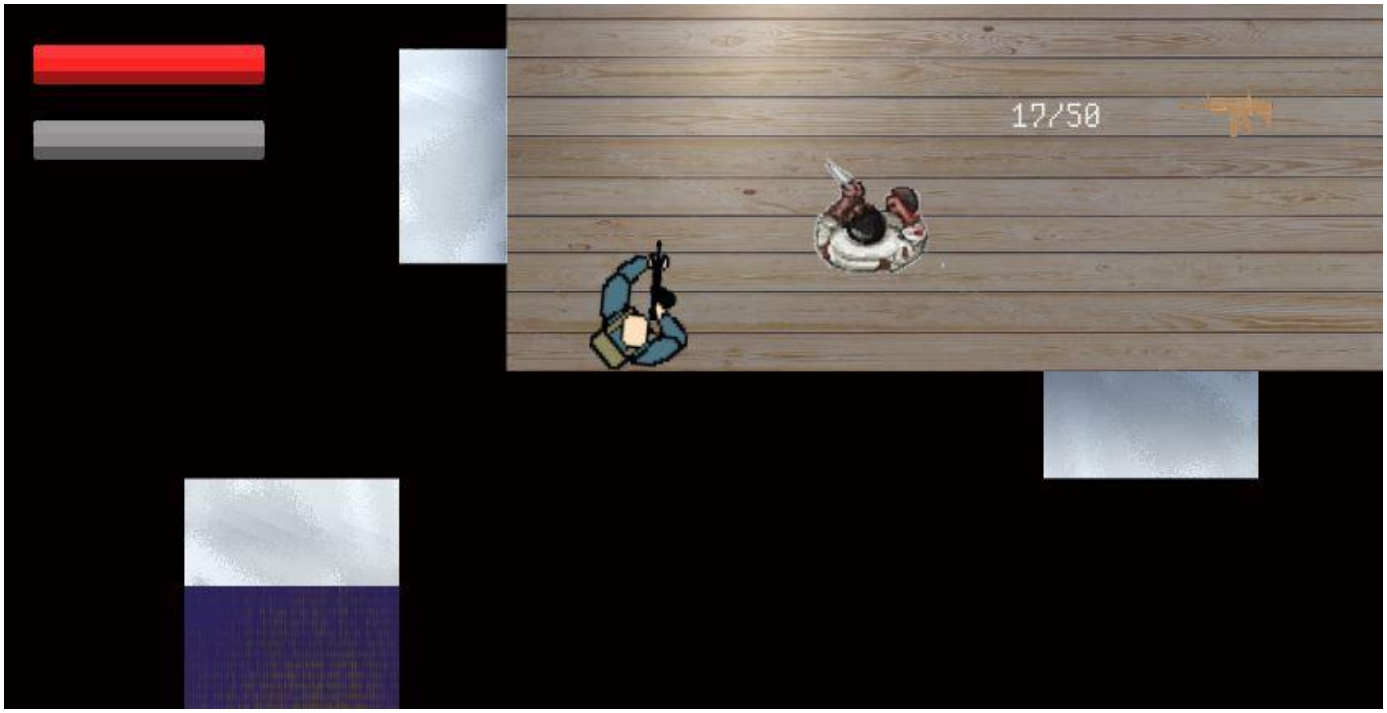


Figura 48 : Captura Joc

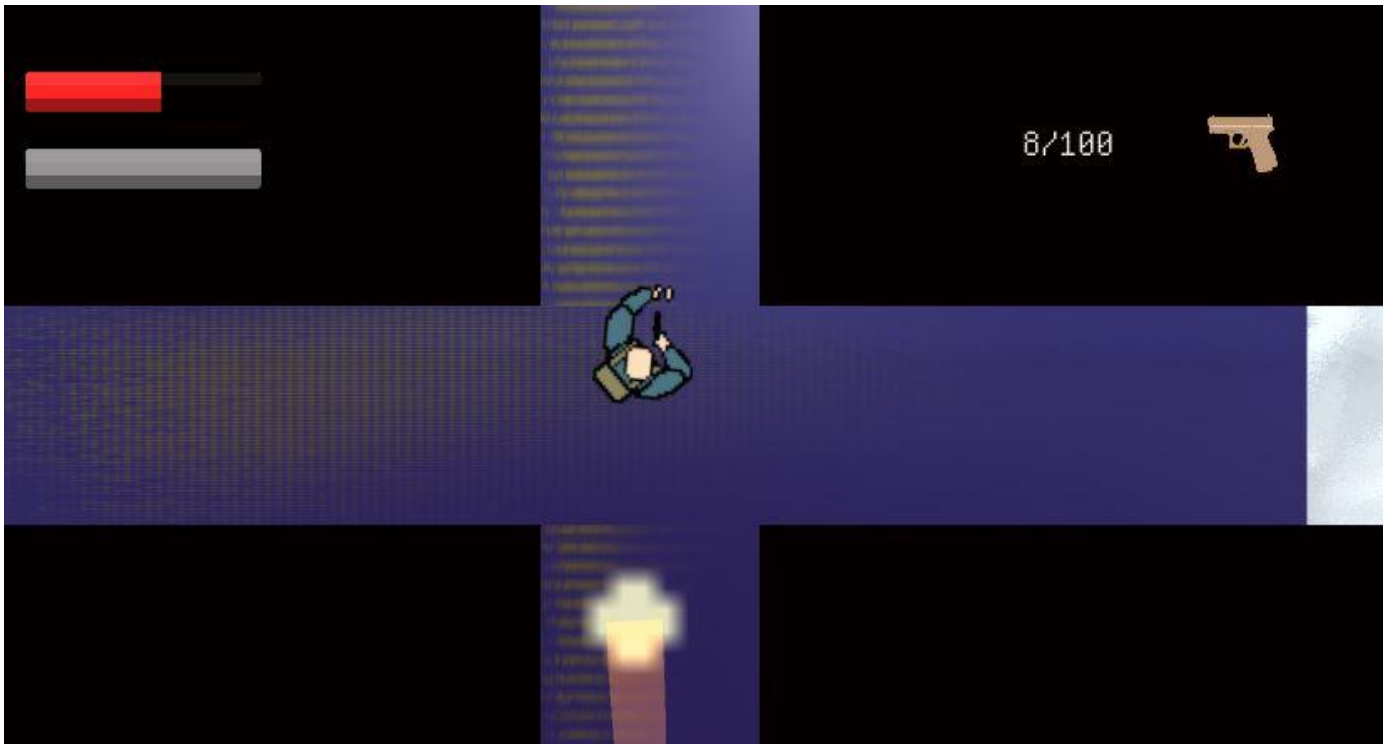


Figura 49 : Captura Joc

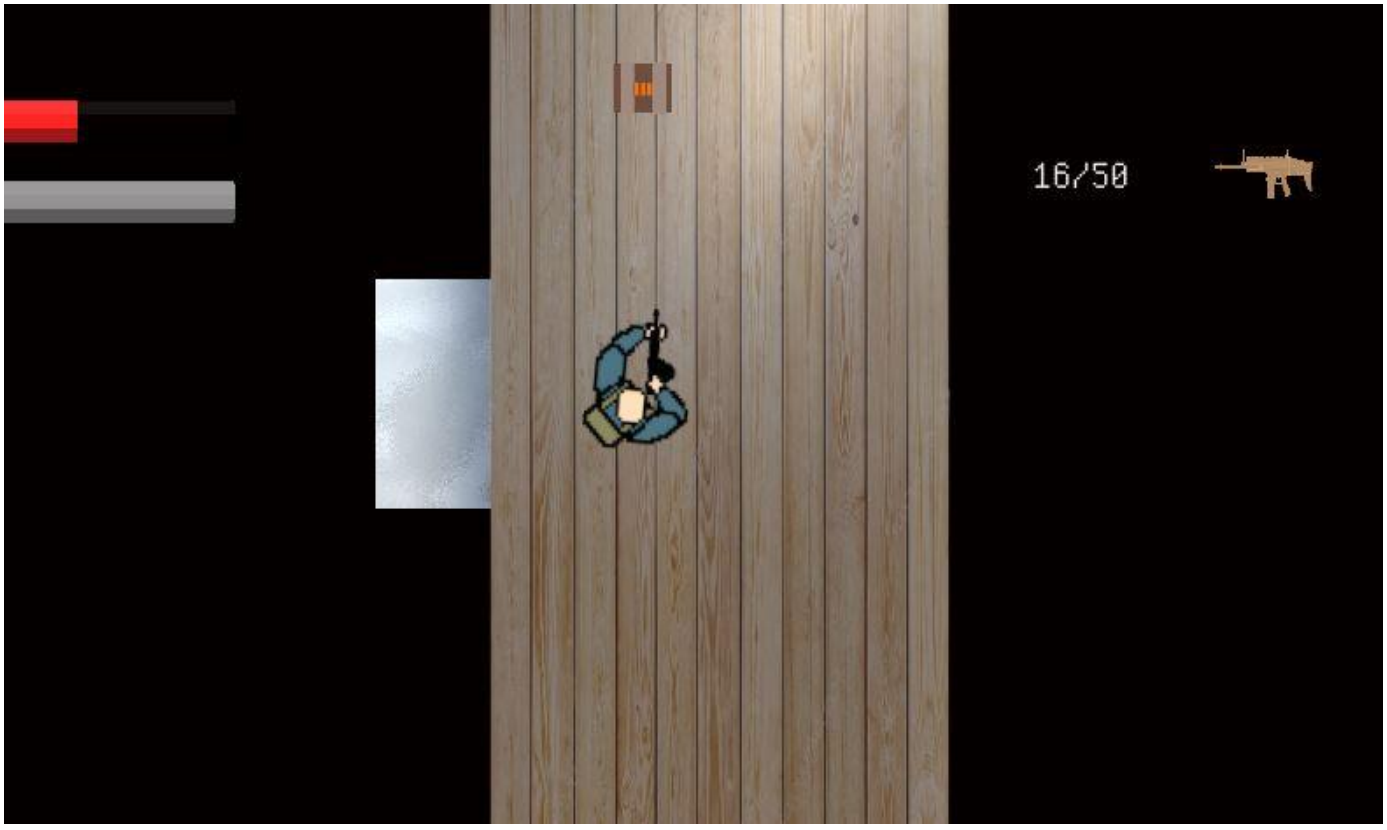


Figura 50 : Captura Joc

Figura 51: Captura Joc

Figura 52 : Captura Joc

Figura 53 : Captura Joc

8. Conclusions

8.1 Valoració del treball

Després d'aquestos mesos de treballar i desenvolupar el projecte m'he trobat en un punt on, tot i haver desenvolupat el projecte i haver dedicat bastant temps a fer-lo funcionar i optimitzar, he hagut de deixar elements que planejava afegir o inclús retallar en altres parts del projecte per arribar a la meta final de temps.

És el primer projecte on manipulo la generació procedural, tot i haver llegit sobre el sistema i haver vist tutorials i vídeos concepte mai havia tocat aquesta tecnologia. Això ha comportat que, tot i haver utilitzat un Plug-in per a fer aquesta generació, he estudiat el codi per intentar comprendre com funciona i fer canvis o modificacions adients per adaptar-lo a la idea que tenia. També, en crear les sales o escenaris que fa servir el sistema, he après a crear aquestes de manera orgànica i distribuïda, és a dir, sense sobrecarregar la vista del jugador i fer-les més naturals, inclús afegint scripts a aquests "objectes" per tal de personalitzar-los de manera més adient.

Una altra tasca que he realitzat ha estat l'ús dels behaviour trees, aquesta tasca ha estat més desafiant, ja que tot i ja haver treballat amb aquest sistema, he intentat reaprendre la manera correcta d'organitzar-los i crear-los. De la mateixa manera que un humà pensa i du a terme accions, he intentat recrear-ho per tal de formar una intel·ligència més orgànica i atacar situacions més generals, per tal de després anar a maneres més concretes de realitzar l'acció i aixi d'evitar grans longituds de codi.

L'apartat artístic ha estat una de les tasques més dures, ja que no s'hem dona especialment bé i he intentat dur a terme les altres tasques primer (cosa que ha portat que durant gran part del desenvolupament el joc es jugués amb formes geomètriques simples en comptes dels sprites actuals). Això ha comportat que aquesta tasca fos la que menys ganes de treballar m'ha portat i s'ha allargat més del que m'agradaria reconèixer. Això ha estat un problema, ja que ha repercutit en el temps, per tant, un dels propòsits que he fet durant aquest projecte ha estat millorar la meva habilitat artística i les diverses tasques relacionades per tant d'evitar problemes en un futur.

Finalment, tot i haver escrit tot l'anteriór puc dir que he après molt fent aquest projecte. Es diu que dels errors s'en aprèn i després d'aquest projecte ho puc afirmar sense dubtes, hecomes gran quantitat d'errors en diversos aspectes i tasqués m'he anat adonant d'aquests i els he intentat corregir en la major manera possible. N'hi ha hagut que han estat més fàcils, però alguns, com els relacionats amb l'organització del temps o l'estructura del projecte, han estat més difícils.
Tot i això, també volia donar importància a la motivació, ja que és una lluita constant contra aquesta, perquè quan acompanya et pot estar hores i hores treballant i ser eficient en la feina, però quan ens abandona, una hora de feina es fa eterna i és possible que s'hagues fet només mitja hora de feina útil. És molt important compaginar el descans i no cremar-te quan es treballa en un projecte important, això és una lliçó que em quedarà per sempre.

Per últim no podem oblidar el que s'ha mencionat a l'apartat 6.7.1, a causa de problemes esporàdics amb els assets.

8.2 Desviacions de la planificació original

A causa de diversos contratemps i tasques que se n'han anat acumulant, hem hagut de retallar temps de treball en l'estética del joc (incloent personatges), en el número de

diferents elements en la generació procedural (diferents sales, passadissos, etc) i en plantejat un segon Boss.

9. Treball futur

Com hem mencionat a l'apartat 8.2, hem tingut vàries desviacions de la planificació global. Per tant, el primer pas en un futur seria arreglar/completar els elements que hem hagut de retallar, tot i això també hi ha altres continguts necessàris a desenvolupar si volguéssim continuar el projecte en un futur:

- **Millora de IA general**
  En aquest apartat voldríem millorar els enemics bàsics i els bosses per oferir un repte més gran al jugador i adaptabilitat de la IA depenent de la situació.

- **Millora de generació de nivells**
  En l'apartat de generació de nivells voldríem oferir la possibilitat que el jugador pogués configurar la generació, així modificant els elements utilitzats i definint la probabilitat d'aparició d'aquests. Un exemple seria la possibilitat d'excloure la sala X de la generació de nivells i fer que la sala Y tingues un 10% més de possibilitats d'aparèixer, així podríem inclús crear més dificultat en el joc sense haver de modificar els enemics.

- **“Jugabilitat continua”**
  Voldríem modificar el sistema actual per tal que, quan es derroti l'enemic, es baixi a un altre nivell i que no acabi el joc.

- **Desenvolupar la narrativa**
  En aquest apartat en refèrrim a la creació de una narrativa per unir els entorns: una historia, conversacions entre enemics, conversacions amb aliats, …

- **Rework de l’art**
  Com hem comentat a l'apartat 7.1 hem utilitzat elements que no són de lliure ús i per comercialitzar el joc ens faria falta arribar a un acord econòmic amb el d’autor. En un possible futur voldríem que l’art fos de ús lliure, o millor inclús, creat per nosaltres.

- **Millora general de elements**
  En aquest apartat ens referim a l’increment dels diferents objectes als escenaris, més armes, més enemics, … Tot el necessari per millorar l’experiencia de joc i evitar repetir sempre els mateixos elements.

10. Bibliografia

- Unity Technologies Unity Manual.
  https://docs.unity3d.com/Manual/index.htmlhttps://docs.unity3d.com/Manual/index.html

- Unity Technologies Unity Forum.
  https://forum.unity.com/

- Unity Technologies Unity Answers.
  https://answers.unity.com/

- Stack Exchange, Inc Stack Overflow.
  https://stackoverflow.com/

- Stack Exchange, Inc Game Development Stack Exchange.
  https://gamedev.stackexchange.com/

- Wikimedia Foundation Wikipedia, the free encyclopedia.
  https://www.wikipedia.org

- Itch.io Open indie game marketplace and DIY game jam host
  https://itch.io/

- TheKiwiCoder Canal de Youtube "TheKiwiCoder"
  https://www.youtube.com/c/TheKiwiCoder

- Procedural Level Generator by Juan Rodriguez
  https://assetstore.unity.com/packages/tools/ai/procedural-level-generator-136626

11. Annexos

Com a Annex, s'ha adjuntat el projecte sencer de Unity. Dins de la carpeta del projecte, es pot explorar tots i cada un dels assets creats així com tot el codi escrit. Els elements del projecte estan organitzats per carpetes, cada una conté els recursos visuals i el codi dels objectes de joc. Per exemple a la carpeta TheKiwiCoder es troba tots els elements relacionats amb el Add-On que permet la creació de behaviour trees.

12. Manual d'instal·lació

12.1 Iniciar el joc

Per a iniciar el joc, en la carpeta "DetectiveAway", hem de buscar i executar el fitxer executable "New Unity Project.exe". Un cop començat el joc es mostrarà el logotip del Unity i després el menú principal. Per començar una partida hem de fer clic al botó Start.

12.2 Controls

- Moviment: el jugador es mou en totes direccions utilitzant les tecles WASD.
- Disparar: Botó esquerre del ratolí.
- Canviar arma: Els botons 1 (pistola), 2 (Rifle) i 3 (Escopeta).
- Espai: Es realitza un ràpid lliscament cap a la direcció que s'està mirant.
- Shift: Es comença a còrrer.
- Pausa: Tecla Esc. S'atura el joc i s'obre el menú de pausa.

12.3 Objectiu del joc

L'objectiu del joc és trobar el cap de la màfia i derrotar-lo, per aconseguir-ho s'haurà d'explorar la instal·lació de la màfia.