Anima mea

Noel Sansó Barceló
Grau d'enginyeria informàtica
TFG videojocs

Nom Consultor/a: Nwdd Garcia Romero
Nom Professor/a responsable de l'assignatura: Joan Arnedo Moreno

Data Lliurement: 06/06/2021

DEDICATÒRIA

Als meus pares, Sebastià Sansó Más i Magdalena Barceló Riutort, que m'han acompanyat en aquest llarg camí i han fet tot tipus de sacrificis per oferir-me una bona educació i la possibilitat de dedicar-me i centrar-me en els estudis.

Al meu germà, Sebastià Sansó Barceló, un dels motius pels quals he elegit aquest camí, i que sempre m'ha ajudat i aconsellat amb la seva experiència i coneixements.

A la meva persona preferida, capaç de fer-me veure un raig de llum en els dies més grisos. Gràcies Maria del Mar Perelló Moyà.

FITXA DEL TREBALL FINAL

Títol del treball:	Dungeon roguelike-like top-down shooter amb unity i c#
Nom de l'autor:	Noel Sansó Barceló
Nom del consultor/a:	Nwdd Garcia Romero
Nom del PRA:	Joan Arnedo Moreno
Data de lliurament (mm/aaaa):	06/06/2021
Titulació o programa:	Enginyeria informàtica
Àrea del Treball Final:	Videojocs
Idioma del treball:	Català
Paraules clau	Rogue-like, bullet-hell, rpg, top-down, shooter

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

L'objectiu d'aquest projecte és el desenvolupament d'un videojoc com a treball de fi de grau dels estudis d'enginyeria informàtica. Per això s'han aplicat tots els coneixements obtinguts durant el grau emprant un motor de videojocs i altres eines.

El videojoc que s'ha desenvolupat és un *shooter RPG* amb perspectiva *top-down* amb elements forts de tipus *Rogue-like.* El jugador ha d'acabar amb les criatures que es trobi a cada sala de la *dungeon* per alliberar les ànimes que aquestes guarden. A més podrà millorar les seves estadístiques de combat agafant els ítems que trobarà a cada nivell per poder fer-se més fort i arribar més lluny.

La prioritat ha estat en tot moment l'ús de bones pràctiques de programació per tal de crear un joc modular al qual es poden afegir noves funcionalitats amb relativa facilitat. Pel que fa a l'art s'ha intentat que la despesa fos menor o fins i tot nul·la, però tot i això, el resultat ha estat bastant satisfactori. Finalment durant la segona etapa s'han implementat tècniques de *game feel* per millorar l'experiència de l'usuari.

Els coneixements i experiència que s'han obtingut amb aquest desenvolupament han estat moltíssims encara que en alguns camps no s'ha pogut aprofundir del tot ni tampoc s'han pogut implementar totes les idees que han anat sorgint durant l'etapa de desenvolupament a causa de les limitacions de temps. Tot i això, el resultat ha estat molt satisfactori i l'experiència de desenvolupament ha estat increïble.

Abstract (in English, 250 words or less):

The aim of this project is to develop a video game as a final degree project in computer engineering studies. To accomplish this, all the knowledge obtained during the degree has been applied using a video game engine and other tools.

The video game that has been developed is a top-down perspective RPG shooter with strong Rogue-like elements. The player must kill the creatures in each room of the dungeon to free the souls they guard. He will also be able to improve his combat stats by grabbing the items he will find at each level, so you can get stronger and get further in this adventure.

The priority at all times has been the use of good programming practices in order to create a modular game to which new functionalities can be added with relative ease.

As far as art is concerned, attempts have been made to make the expenditure less or even zero, but even so, the result has been quite satisfactory. Finally, during the second stage, game feel techniques have been implemented to improve the user experience.

The knowledge and experience obtained with this development are many, although in some fields it has not been possible to deepen completely nor have been able to implement all the ideas that have been arising during the stage of development due to of time constraints. However, the result has been very satisfying, and the development experience has been amazing.

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

El desenvolupament de videojocs és un escenari perfecte per demostrar tots els coneixements obtinguts durant l'estudi del grau en enginyeria informàtica, sobretot la part de programació d'alt nivell que és una de les parts amb què s'ha tingut més cura per tal que el producte final sigui modular i es puguin afegir funcionalitats de manera fàcil i àgil. Un altre exemple d'aplicació de coneixements estudiats seria la intel·ligència enemiga per la qual s'han emprat cerques com A* vistes a l'assignatura intel·ligència artificial. I com aquests molts altres exemples d'aplicació de coneixement obtingut durant el grau (Màquina d'estats, interfícies, herència, diagrama de classes, gestió de projectes, etc.).

A més, la indústria dels videojocs s'ha vist encara més impulsada el 2020 per la pandèmia de la COVID-19. I és que va acabar l'any passat amb un gran creixement i una facturació global de més 165.000 milions d'euros.[1]

Avui en dia, amb limitacions de mobilitat per a moltes persones en tot el món es preveu que l'entorn *gaming se* seguirà ampliant i registrant altes xifres de consum. Fa temps que aquest sector genera més ingressos que el cine i el deport junts. També s'ha de tenir en compte que avui en dia els videojocs acompanyen a les persones durant tota la seva vida i només durant uns quants anys com en el passat.

Vull destacar que aquest joc s'ha orientat a mòbil a causa de la crisi de targetes gràfiques que s'està patint, que portarà a altres plataformes a guanyar usuaris i audiència i que ha obligat a les grans firmes de videojocs a ajornar els desenvolupaments de molts de projectes.

Finalment cal destacar que desenvolupar un videojoc és un projecte en què l'enginyeria i l'art (música, disseny, efectes especials, etc.) es combinen durant tot el procés de desenvolupament i la vida del producte, fet que ha estat determinant a nivell de motivació personal, ja que en cas de no haver estudiat enginyeria informàtica, hauria volgut estudiar algun grau relacionat amb el disseny gràfic.

1.2 Objectius del Treball

L'**objectiu principal** d'aquest projecte ha estat el desenvolupament d'un conjunt de nivells jugables d'un *shooter RPG* amb perspectiva *top-down* amb influències del gènere *Rogue-like* emprant Unity i C#.

Ara bé, per assolir aquest objectiu principal s'han anat complint els objectius que s'indiquen a continuació, els quals no es troben necessàriament ordenats per data de compliment:

- Definir la història i la temàtica del joc, i acotar l'abast del projecte.
- Definir les fases de desenvolupament que seran necessàries.
- Definir la mecànica de joc i la jugabilitat principals.
- Seleccionar la plataforma de destí que més interessa.
- Seleccionar el motor de videojocs que s'emprarà així com les eines complementàries que es puguin necessitar.
- Disseny artístic dels nivells, personatge, ítems i enemics.
- Disseny del so.
- Programació del videojoc. Aquest objectiu és el que més es pot descompondre. Els diferents desenvolupaments s'explicaran al llarg de la memòria.
- Documentació de les fases de desenvolupament.

1.3 Enfocament i mètode seguit

El desenvolupament d'aquest projecte s'ha fet des de zero, és a dir, no s'ha agafat cap producte com a punt de partida per millorar-lo o afegir-hi noves característiques. D'aquesta manera s'han anat adquirint i aprenent les bases i els coneixements de Unity de manera progressiva a mesura que s'anava afegint valor el producte final, de manera que quan es necessitava una funcionalitat determinada s'investigava les diferents formes d'implementar-la, després s'afegia de la forma que millor encaixava en el projecte i finalment es provava i es reajustava.

Les proves de les funcionalitats, s'han fet de manera manual, i és una opció que gran part de desenvolupadors *Indie* segueixen, Vegem per exemple la resposta de Nijman en una entrevista sobre *game feeel:*

RPS: "When you're tweaking things like fire rates, audio, bullet travel time, etc., is it a purely intuitive and iterative process?"

Nijman: "*We firmly believe that a game is only a game when you're playing it. No pile of code/design document/list of values can really express anything as well as playing a game. That's why we spend half our development time shooting monsters. Basically this is a highly iterative, partly intuitive business. That intuition comes from having spent years making games about shooting and running around. Obviously there is some logic involved*".

A més, he intentat gaudir durant tot el desenvolupament tal com indica Nijam a la mateixa entrevista: "*One thing we've been working on hard lately is making the process of making games less painful and more fun for the entire team. We believe that fun we're having while making this can also be "felt" (that word again, argh) when you play Nuclear Throne*".

A més, la metodologia seguida ha estat la regla del Loop :

El supòsit bàsic darrere de la regla del bucle és el següent:

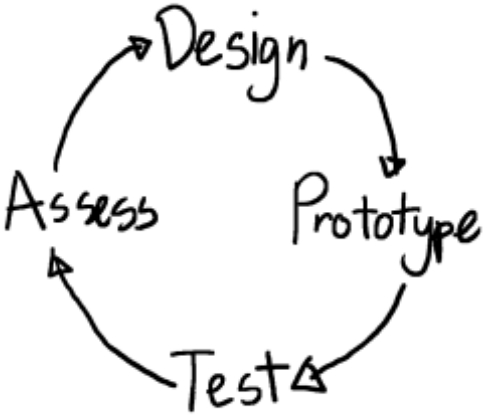
"No es pot saber si un joc és divertit fins que no s'hagi provat."

Molts dissenyadors obliden aquest supòsit. Jo mateix n'he estat culpable sobretot durant la primera part del desenvolupament on només m'encarregava d'implementar funcionalitats noves sense fer cas a si aquesta augmentaven la diversió del producte. De fet, el resultat de la primera PAC, tot i que molt funcional, va ser molt pobre a nivell de jugabilitat i diversió, gairebé no pareixia un joc. És fàcil caure en el parany de pensar que el disseny sobre el paper serà divertit quan s'implementi.

El motiu de la suposició bàsica és que no es pot predir l'element humà aleatori que els jugadors introdueixen al joc. A més, la diversió és un concepte subjectiu molt complicat. És una d'aquestes coses que "es saben quan es veuen" que només es pot validar mitjançant proves de joc.

Un cop es coneix aquest supòsit, la regla del *loop* diu:

- Els jocs es fan mitjançant canvis incrementals.
- Els canvis incrementals s'afegeixen al joc mitjançant les iteracions.
- S'han d'abordar problemes per iteració ben definits.
- Com més ràpid es completen les iteracions, més iteracions es poden fer. Com més iteracions es fan, millor serà el joc.



Il·lustració 1 - La regla del loop

"*Com més vegades proveu i milloreu el vostre disseny, millor serà el vostre joc."* (Jesse Schell, Art of Game Design).

Per tant, el gran repte del disseny de jocs és treure tants bucles efectius i substancials com sigui possible durant la línia de temps de desenvolupament del joc.

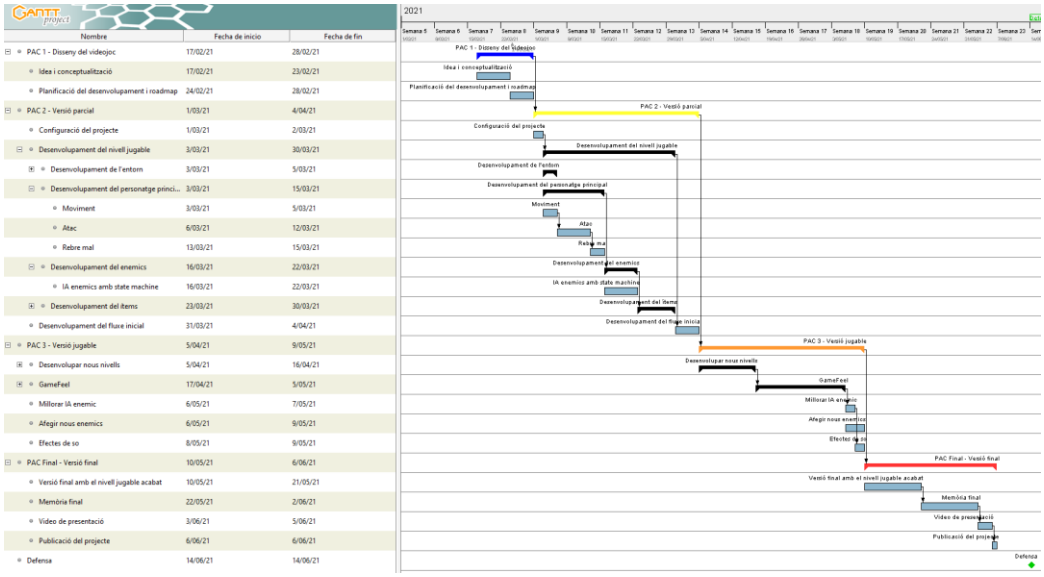
Per acabar aquesta secció, vull destacar que en aquest desenvolupament tots els rols típics d'una empresa de desenvolupament de videojocs han recaigut sobre la meva persona (programador, *game i level designer*, animador, especialista d'àudio, *tester*, etc.).

Així he intentat emprar bones pràctiques de desenvolupament per a poder afegir noves funcionalitats però al mateix temps poder tenir un producte lliurable en tot moment. Així, tenia el repositori a GitHub on anava fent el desenvolupament en diferents branques, i quan les funcionalitats eren estables les afegia a la branca principal.

A més, per tenir el codi dividit per l'entrega de les diferents PAC, he creat un segon repositori on només s'ha pujat el codi corresponent al lliurament de cada PAC.

1.4 Planificació del Treball

La planificació d'objectius d'aquest projecte s'ha realitzat en un diagrama de Gantt on s'han especificat les diferents fites i dates juntament amb la planificació per a cada un d'aquests. Cal destacar que aquesta planificació ha patit algunes desviacions respecte a la planificació inicial de la PAC, però és cert que s'han anat complint els objectius amb molta rugositat. Aquest és el diagrama de Gantt corresponent:



Il·lustració 2 - Diagrama de Gantt de la planificació del projecte

1.5 Breu sumari de productes obtinguts

Els productes obtinguts són aquells que s'havien proposat aconseguir des del principi i alguns altres que han arribat de manera indirecta per arribar a aconseguir els primers.

- Executable d'una versió jugable per a Windows i Android.
- Memòria final
- PAC amb els respectius vídeos
- Informe d'autoavaluació
- Vídeo explicatiu
- Repositori amb el codi font del projecte
- *SpriteSheet* amb tot el píxel art emprat
- Conjunt de sons i àudio recopilats

1.6 Breu descripció dels altres capítols de la memòria

Capítol 2: S'estableix el tipus de gènere, subgènere i referència a altres videojocs existents i els tipus de plataforma de destí. També es valoren les tecnologies de desenvolupament més adequades i emprades pel tipus de joc i la plataforma de distribució elegides. Finalment es fa una valoració econòmica.

Capítol 3: Descripció del joc, història i ambientació. Definició dels personatges i interacció entre aquests. Objectius del jugador i *concept art* dels nivells, personatges, etc.

Capítol 4: Justificació de l'entorn i eines elegides per al desenvolupament així com els requisits tècnics de l'entorn de desenvolupament. Recull i descripció de les eines emprades. Recursos gràfics i d'àudio emprats i el seu origen. Arquitectura i organització del joc. Explicació de la IA enemiga.

Capítol 5: Guia de creació dels nivells jugables explicant els criteris de disseny a tall de guia de joc.

Capítol 6: Manual d'usuari amb instruccions així com els requisits de maquinari necessaris per poder jugar.

Capítol 7: Conclusions tant generals del mateix projecte com personals.

Capítol 8: Glossari amb termes utilitzats.

Capítol 9: Bibliografia, Webgrafia i altres referències.

Capítol 10: Annexos amb els requeriments, instruccions i repositori del codi font del videojoc.

2. Estat de l’art

2.1 Gènere del joc

Aquest videojoc serà un *rpg*[1] d’acció 2D *top-down*[2], amb forts elements de *rogue-like*[3]. A més, pel que fa a determinats atacs dels enemics es pot considerar que té una gran influència dels *bullet hell*[4].

[1]: Gènere de videojocs on el jugador controla les accions d'un personatge (o de diversos membres d'un grup) immers en algun món detallat. <<Wikipedia>>

[2]: És un videojoc que ofereix una perspectiva elevada per sobre de l'acció, per tant, es pot considerar que té una perspectiva de dalt. Pot ser una mica similar a una perspectiva de tercera persona amb la diferència que una vista *top-down* sempre està per sobre i en una posició i / o rotació fixes. <<Wikipedia>>

[3]: És un subgènere dels videojocs de rol que es caracteritzen per viure una aventura a través de masmorres. <<Wikipedia>>.

[4]: Subgènere dels *shoot'em up* o jocs de trets tradicionals que es caracteritzen perquè el jugador ha de esquivar quantitats ingents de bales que apareixen en pantalla en forma de patrons, que s'han de memoritzar per poder esquivar-les. <<Gamerdic>>.

2.2 Referències a videojocs existents

Un videojoc de rol, o comunament també designat mitjançant les sigles RPG (*role-playing game*), és un gènere de videojocs on el jugador controla les accions d’un personatge immers en algun món detallat.

Concretament ens trobem davant un ARPG (*action role playing games*), que són un gènere de videojocs que comparteixen moltes característiques amb els RPG, però que, a diferència d'aquests, ofereixen combats en temps real. A més, també tindrà influències *rogue-like* que és un subgènere dels videojocs de rol que es caracteritzen per la vivència d’una aventura a través de masmorres.

S’han agafat referències de jocs com les que es mostren en el següent vídeo i com que s’expliquen a continuació:

2.2.1 The Binding of Issac

És un joc d'acció *RPG* amb forts elements de tipus *Roguelike*, en el qual els nivells són generats aleatòriament. En el transcurs del viatge d'un nen anomenat Isaac, els jugadors trobaran estranys tresors que canviaran la forma d'Isaac, li donaran habilitats i poders sobrehumans

que li permeten lluitar contra onades de criatures aterridores i diabòliques, descobrir secrets i obrir-se camí a la seva supervivència.

El títol i la història del joc estan inspirades en la història del sacrifici d'Isaac recollit en la Bíblia, segons la qual Déu va manar a Abraham sacrificar a Isaac, el seu únic fill.

Il·lustració 3 - The binding of Isaac scene

2.2.2 Rogue Heroes: Ruins of Tasos

És un RPG top-down per combatre els titans a les masmorres procedurals. Mostra molta influència d'elements dels *rogue-lite* moderns.

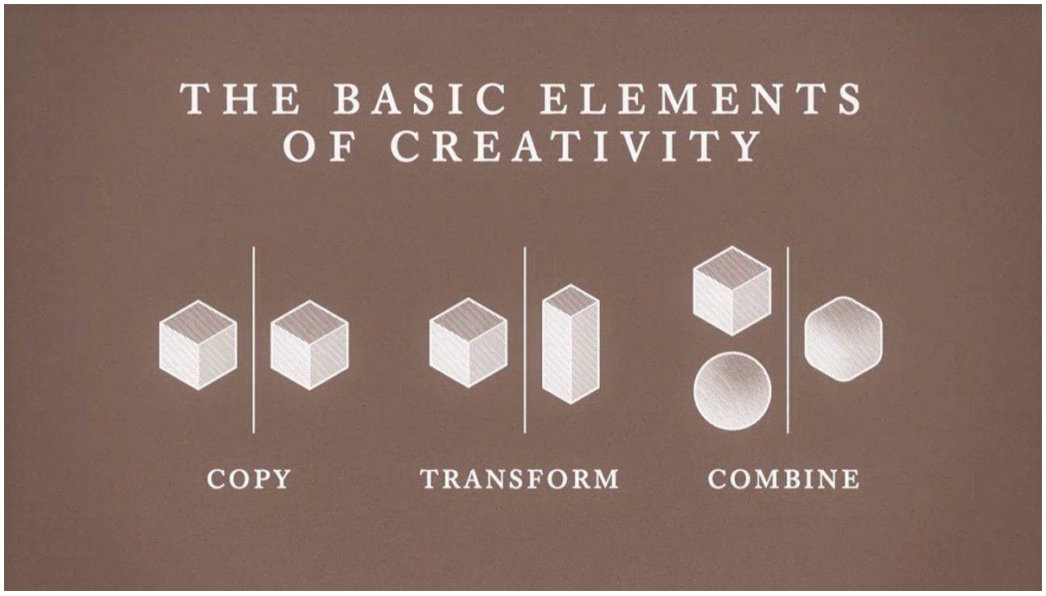
Il·lustració 4 - Ruins of Tasos scene

2.2.3 Crashlands

És un videojoc de rol d'acció-aventura desenvolupat i publicat per Butterscotch Shenanigans. El joc es descriu com un "joc de *crafting* basat en la història" i encarrega als jugadors que recopilin objectes per elaborar armes i armadures, les quals s'empraran per enfrontar-se a diferents enemics.

Il·lustració 5 - Crashlands scene

Vull destacar que en aquest projecte no s'han inventat noves mecàniques de joc, de fet, s'han pres com a referència alguns videojocs clàssics i actuals que han tingut gran repercussió dins el gènere, això sí, s'ha intentat donant-li un toc d'originalitat perquè es distingeixi de la resta.



Il·lustració 6 - The basic elements of creativity

Si ens basem pels principis bàsics de la creativitat de la figura anterior, podríem dir que el producte obtingut és una transformació.

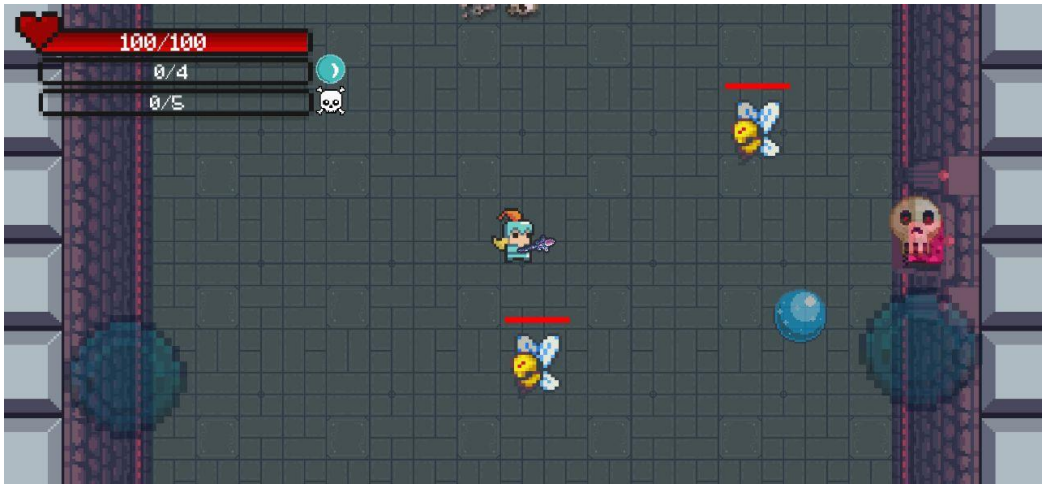
2.3 Tipus d'interacció i plataforma de destí

El jugador controla el personatge principal, el qual tindrà moviment cap a 8 direccions i treta boles màgiques (cap a totes les direccions), i pot equipar ítems que milloraran les seves habilitats, ja sigui reduint l'*attack cooldown*, millorant la velocitat, la vida màxima, la vida actual, etc.

En resum, el personatge té:

- Moviment 8 direccions
- Atac amb projectils
- Recol·lecció d'ítems que milloren les estadístiques del personatge
- Ús d'una bombolla protectora
- Ús de l'habilitat definitiva que permet tretar tres projectils:

Aquests són els controls *on-screen*:



Il·lustració 7 - Controls On-Screen dispositiu Android

La plataforma de destí és *Android*, encara que es pretén usar el nou sistema d'inputs que ofereix *Unity*, així que encara que sigui per *Android*, està preparat i és fàcilment configurable per moltes altres plataformes. A més, m'agradaria seguir desenvolupant aquest joc més enllà del TFG i crec que Android és una de les opcions més viables per mostrar al món un joc com aquest, encara que si té èxit, m'agradaria adaptar-lo a *Nintendo Switch* i *Xbox* o *PlayStation*.

2.4 Plataformes emprades per al desenvolupament

2.4.1 Avaluació d'*engines* i *kits* de desenvolupament

Unity és una eina de desenvolupament de videojocs creada per l'empresa *Unity Technologies* orientat a desenvolupar videojocs, tant 2D com 3D. Utilitza un llenguatge de programació d'alt nivell com és *C#*. Disposa d'una excel·lent documentació, de les millors documentacions de programari que existeixen i d'una comunitat disposada a ajudar davant qualsevol dubte. Una de les característiques més importants i més còmodes de *Unity* és que suporta l'exportació a una quantitat enorme de plataformes. En general, és un dels motors més complets del mercat, molt fàcil d'utilitzar gràcies al seu editor i la seva corba d'aprenentatge no és massa empinada.

Godot Engine és un motor de joc multiplataforma amb funcions per crear jocs 2D i 3D a partir d'una interfície unificada. Ofereix un conjunt complet d'eines comunes perquè els usuaris puguin centrar-se en la realització de jocs sense haver de reinventar la roda. Els jocs es poden exportar amb un sol clic a diverses plataformes, incloses les principals plataformes d'escriptori (*Linux*, *macOS*, *Windows*), plataformes mòbils (*Android*, *iOS*), així com plataformes basades en web (*HTML5*) i consoles. *Godot* és completament gratuït i de codi obert sota la molt permissiva llicència MIT. La documentació oficial està allotjada a *ReadTheDocs*. La comunitat *Godot* la manté al seu propi repositori de GitHub.

És un motor comercial enfocat especialment en el desenvolupament de videojocs en 2D. Té el seu propi llenguatge de script anomenat *GML* (*GameMaker Language*), aquest té una sintaxi força senzilla. El seu

editor és simple, utilitza un sistema d'agafar i arrossegar (*drag and drop*). En general, per a tasques senzilles es pot fer servir els esdeveniments prefabricats que ofereix el motor, però si es cerquen coses més complexes i a mida, es poden programar scripts.

2.4.2 Selecció de l'engine

Entre totes aquestes opcions, s'ha decidit usar *Unity*, en primer lloc perquè s'usa *C#*, un llenguatge molt potent que he usat en el període de pràctiques. A més, la documentació és de les millors que he vist i es poden trobar molts de manuals, tutorials i tota classe de documentació a internet, tant d'usuaris actius com dels que es poden trobar a la mateixa pàgina oficial de *Unity*, així que d'aquesta forma existeix una garantia que no em quedaré encallat quan tingui qualsevol dubte.

Encara que es poden desenvolupar jocs per a diferents plataformes com *Windows*, *HTML5*, *Android*, *iOS*, *PlayStation*, *Xbox*, etc. emprant *Unity*, gran part de la comunitat està d'acord en el fet que és molt més fàcil publicar jocs 2D per a mòbil amb aquesta plataforma.

També cal esmentar, que m'agradaria que aquest joc fos publicat per a consola en versions futures en cas de tenir èxit, i la facilitat que ofereix *Unity* per exportar a aquestes plataformes ha estat un motiu clau.

Un altre motiu de menor importància que ha acabat d'inclinar la balança, és el nou sistema d'inputs que s'ha desenvolupat que fer que l'exportació a múltiples plataformes es simplifiqui enormement.

2.5 Valoració econòmica del projecte

Cal destacar que s'ha intentat sempre l'ús de programari gratuït o amb versions gratuïtes. Pel que fa als *assets*, per reduir el temps de desenvolupament s'han adquirit un conjunt d'sprites a un preu molt reduït, però és cert que com a norma general s'han emprat recursos sense cost.

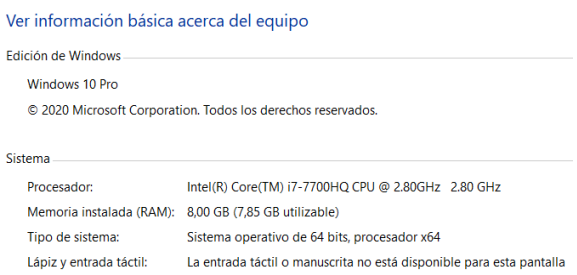
2.5.1 Recursos de programari

Nom	Descripció	Preu
Unity	*Framework C#* gratuït per al desenvolupament de videojocs.	0,00 euros
Visual Studio	*IDE* de programació, la versió comunitària és gratuïta.	0,00 euros
Audacity	Programari d'edició d'àudio i captació de so digital. Es tracta d'un programa completament gratuït	0,00 euros

TexturePacker	És una eina per a desenvolupadors de jocs i dissenyadors web que permet entre altres coses empaquetar *spritesheets*.	0,00 euros
Gimp	És un programa d'edició d'imatges digitals en forma de mapa de bits, tant per dibuixos com per fotografies. És un programa lliure i gratuït.	0,00 euros
	TOTAL	0,00 euros

2.5.2 Recursos de Maquinari

S'ha considerat l'ordinador portàtil com una despesa, però és cert que s'ha emprat durant tota la carrera i probablement es podria obviar el seu cost.

Nom	Descripció	Preu
Ordinador portàtil		1.200 euros

2.5.3 Recursos humans

Disseny d'*sprites* i sons

Els recursos de programari emprats es troben a la següent taula, la major part es troben a la col·lecció d'itch.io que he creat i que podem accedir a través de la font recursos diversos:

Recurs	Font	Despesa total
Recursos diversos	Col·lecció emprada	8,23 euros
Ítems i pocions	http://finalbossblues.com/timefantasy/freebies/halloween-icons/	0 euros
HealthBar	https://opengameart.org/content/pixel-health-	0 euros

	bar-asset-pack-2	
Banda sonora	Col·lecció emprada	0 euros
	TOTAL	8,23 euros

Alguns d'aquests *sprites* han estat editats amb *Gimp* per adaptar-los a les necessitats i mecàniques particulars del joc.

Programació

Tota la codificació l'he feta jo mateix així que el cost d'aquesta és 0, això si, si tenim en compte el cost econòmic que haguessin suposat les hores de desenvolupament. El cost aproximat de desenvolupament hauria estat 270 x 15 euros/hora = **4.050 euros**.

Despesa total

Així, amb totes les despeses calculades als anteriors apartats, el cost de desenvolupament d'aquest projecte és de:

Despeses de programari	**0,00 euros**
Despeses de maquinari	**1.200 euros**
Despeses de recursos humans	**4.050 + 8,23 euros**
Total	**5.258,23 euros**

3. Definició del joc

3.1 Història, ambientació o trama

Des de fa molts d'anys, els habitants de Solaria, els quals són coneguts per les seves habilitats màgiques, han sofrit els atacs de les criatures misterioses, que s'alimenten de les ànimes de les persones, ja que amb cada ànima que consumeixen es fan més forts. Va ser durant un d'aquests atacs quan Tristan, un cap de família, va veure com un monstre robava l'ànima de la seva filla.

Així, Tristan haurà de buscar i matar la criatura que posseeix l'ànima de la seva filla abans que no sigui massa tard, ja que un cos sense una ànima només es manté viu durant uns dies abans de convertir-se en un monstre.

En aquest viatge el nostre protagonista s'haurà d'endinsar en el món misterios d'aquestes criatures, on es podrà ajudar d'ítems recollibles, els quals milloraran les seves habilitats màgiques i el faran més fort, per així poder seguir avançant en aquest viatge.

3.2 Objectius plantejats al jugador

L'objectiu del jugador serà anar superant les diferents sales (*dungeons*) que representen el món dels monstres, només un cop hagi mort tots els enemics de cada sala, podrà passar a la següent. La mort dels enemics donarà diferents ítems que milloraran les habilitats del personatge, i alliberarà les ànimes que les criatures han absorbit. Un cop eliminats tots els enemics de cada sala podrà accedir a les següents sales que l'acostaran a trobar el *boss* final, el qual guarda l'ànima de la seva filla.

3.3 Definició dels personatges i elements principals

3.3.1 Jugador principal

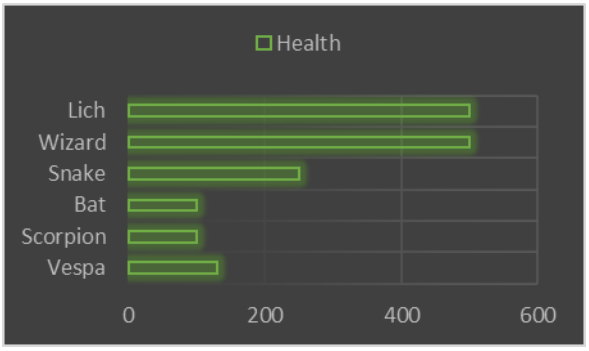
El jugador disposa d'un bastó amb el qual treta boles màgiques que li permetran eliminar les diferents criatures que es trobi. La força, cadència i rang d'aquests projectils així com la velocitat del personatge pot variar amb la recol·lecció d'ítems.

3.3.2 Enemics

- *Scorpion*
- *Vespa*
- *Snake*
- *Bat*
- *Kraken wizard*
- *Lich*

Estadístiques dels enemics

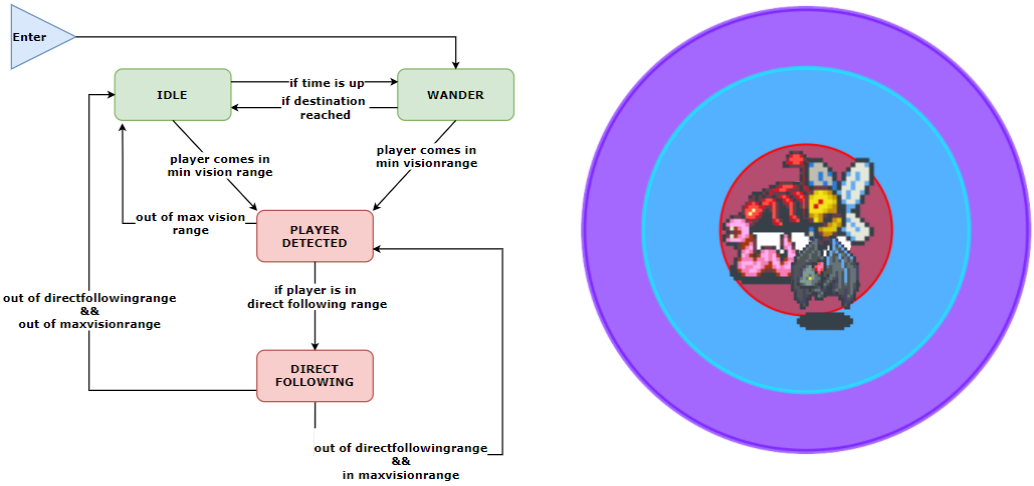
Il·lustració 8 - Estadístiques dels enemics



Il·lustració 9 - Estadístiques de vida dels enemics

Comportament dels enemics

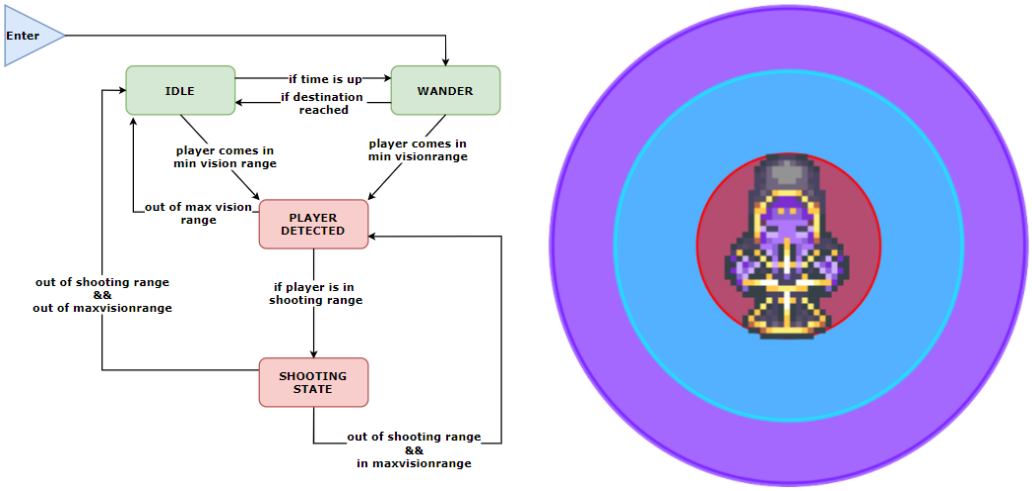
Aquest conjunt d'enemics es mouen a posicions ràndom dins la sala fins que el personatge entra dins el seu rang mínim de visió, un cop passa això el comencen a perseguir emprant la cerca A* fins que s'apropen tant(*direct following range*) que ja no empren cap cerca per seguir el personatge sinó que van directament a la seva posició. També pot passar que l'enemic surti del seu rang de visió màxim, només així els enemics tornen a desplaçar-se de manera ràndom per la sala.



Il·lustració 10 - Comportament Enemic 1

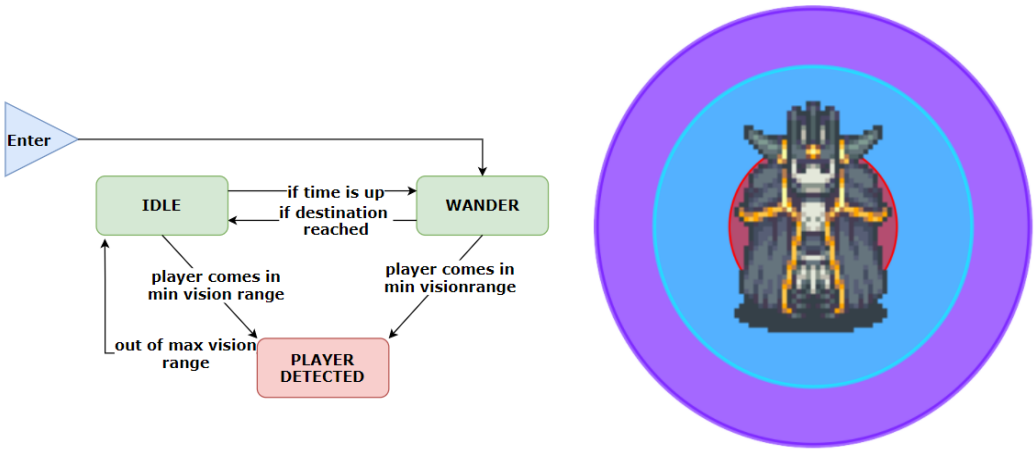
***El rang vermell representa el *shooting range* o el *direct following range* depenent de si el personatge pot tretar o només persegueix.**

El *Kraken Wizard*, quan detecta el jugador el persegueix emprant l'algorisme A* fins que entra al seu shooting range i llavors fa un tret cap a 8 direccions i en espiral respectivament. També pot passar que l'enemic surti del seu rang de visió màxim, només així els enemics tornen a desplaçar-se de manera aleatòria per la sala.



Il·lustració 11 - Comportament Wizard

El Lich, quan detecta el jugador el persegueix emprant l'algorisme A* fins que entra al seu shooting range i llavors fa un tret en espiral.

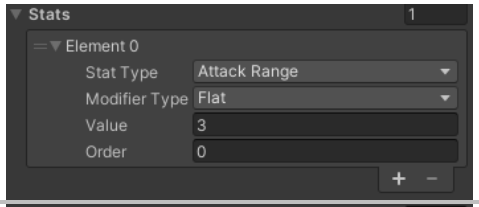
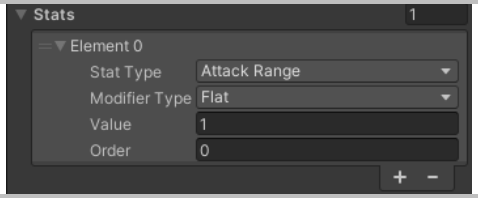


Il·lustració 12 - Comportament Lich

Aquest enemic fa referència un ésser fantàstic, un *lich* (o *liche*) que és un tipus de criatura morta vivent. Sol ser el resultat de la transformació que un bruixot poderós obra sobre si mateix mitjançant encanteris o rituals per assolir la immortalitat. Els *lichs* acostumen a ser representats com a éssers parcialment o totalment cadavèrics o esquelètics, ja que el procés de conversió en *lich* sol lligar el seu intel·lecte amb el seu cadàver inanimat, el qual generalment segueix decaient tot i conservar el seu poder. En la majoria de les ficcions, els *lichs* tenen més poder que quan eren vius, i posseeixen un gran maneig de la *necromància* i altres arts semblants, controlant hordes de morts vivents menors com ara soldats o servents. << viquipèdia >>

3.3.3 Objectes recollibles

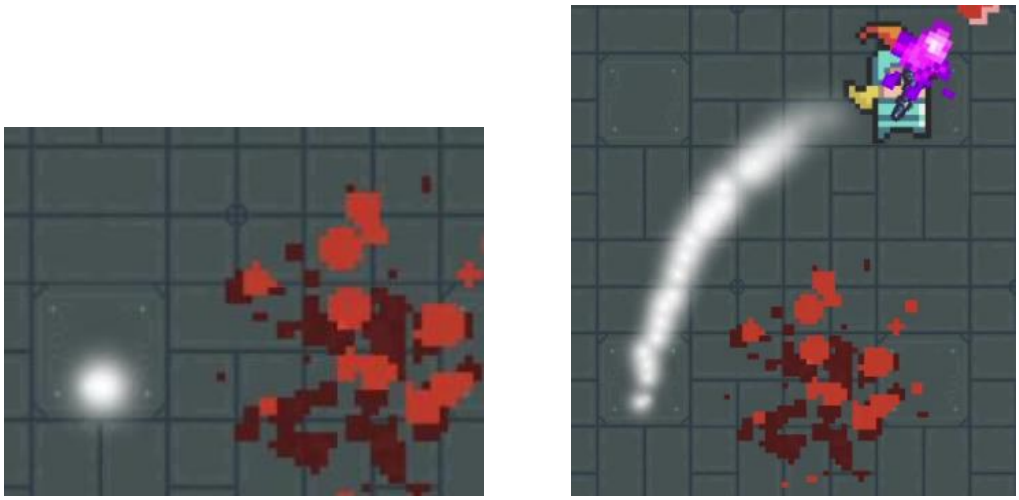
Ítems

	Ull de bruixa	
	Ala de rat-penat	
	Mà d'ogre	
	Poció vermella	+20 *Current Health*

Poció blava

Ànimes

A més dels que es mostren, hi ha un altre ítem recollible que són les ànimes, encara que aquestes no milloren els atributs del personatge principal.



Il·lustració 13 - Comportament de l'ànima

3.3.4 Objectes no recollibles

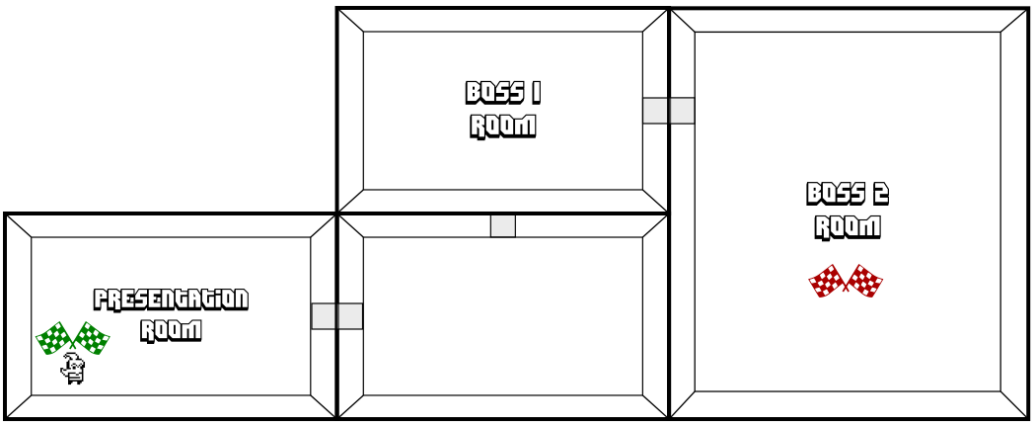
A més, tenim altres objectes que afecten al *gameplay* però que no són recollibles, els podem veure a la següent imatge:

Il·lustració 14 - Objectes no recollibles

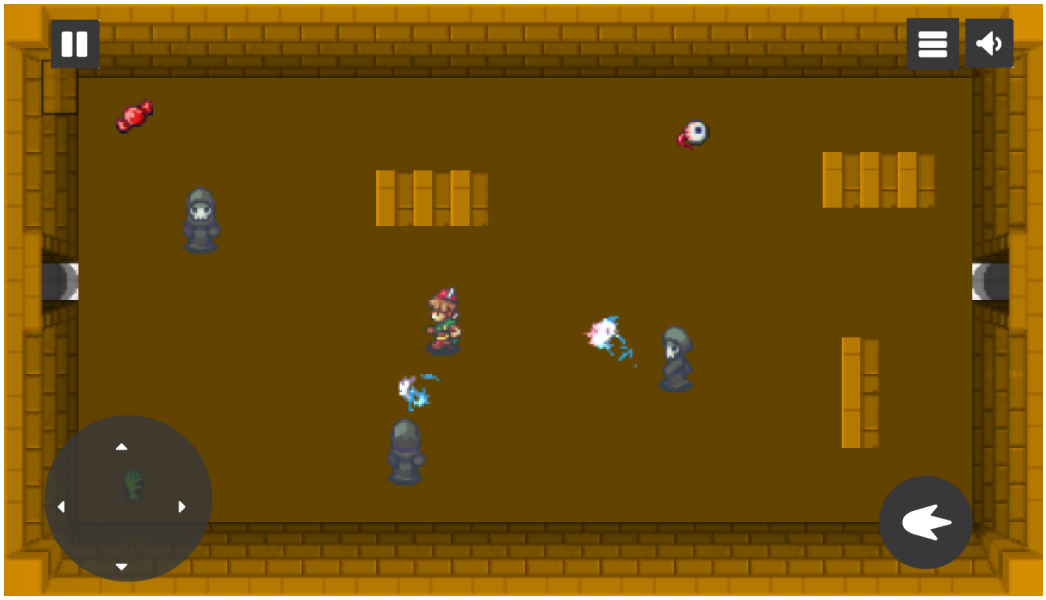
Aquests no són els únics *Tiles* emprats, com a Annex s'afegirà un *spritesheet* amb tots els *Tiles* emprats.

3.4 Concept Art

3.4.1 Nivells i objectes



Il·lustració 15 - Esborrany del conjunt de nivells



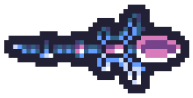
Il·lustració 16 - Esborrany amb Tiled de l'estètica que es pretenia aconseguir

Il·lustració 17 - Captura del resultat final de l'escena del projecte de Unity

3.4.2 Jugador

Il·lustració 18 - Sprite del jugador principal

Il·lustració 19 - Sprite del muzzle flash i la bala



Il·lustració 20 - Sprite de l'arma

3.4.3 Enemics

Il·lustració 21 - Dos sprites de l'scorpion

Il·lustració 22 - Dos sprites de la vespa

Il·lustració 23 - Dos sprites de la Snake

Il·lustració 24 - Dos sprites del kraken wizard

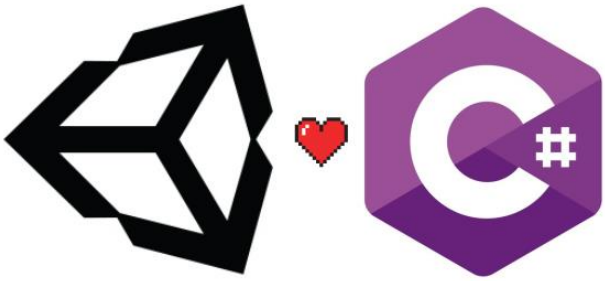
Il·lustració 25 - Dos sprites del Lich

3.4.4 Ítems

Il·lustració 26 - Sprites dels ítems

4. Disseny Tècnic

4.1 Entorn escollit i requeriments



Per a aquest projecte s'emprarà *Unity*, que és un *engine* i *framework* 2D i 3D que proporciona un sistema per dissenyar escenes de jocs i aplicacions per a 2D i 3D. Es diu que és per crear jocs i aplicacions perquè no només existeixen jocs, sinó també simuladors de formació, aplicacions *first-responder* i altres aplicacions centrades en el negoci desenvolupades amb *Unity* que necessiten interactuar amb l'espai 2D / 3D.

Unity permet la interacció no només mitjançant el codi, sinó també mitjançant components visuals, i exportar-los a totes les plataformes principals de forma gratuïta. A més, permet importar i muntar recursos, escriure codi per interactuar amb els objectes, crear o importar animacions per utilitzar-les amb un sistema d'animació avançat i molt més.

Alguns dels avantatges que han fet que Unity sigui l'*engine* i el *framweork* emprat per dur a terme aquest projecte:

- Aquest *framework* ha treballat per garantir el suport multiplataforma i permetre canviar de plataforma literalment amb un sol clic.
- Posseeix una àmplia documentació i multitud de vídeos explicatius. Pel fet que porta més temps accessible econòmicament a molts usuaris, la quantitat d'informació sobre *scripting* i recursos desenvolupats per tercers és enorme.
- A l'hora d'implementar els *scripts*, es pot triar entre dos llenguatges de programació: *C#* i *JavaScript*.
- Està basat en components, cosa que permet la reutilització i estructuració de les entitats usades.
- Els requisits tècnics per treballar amb aquest motor són mitjans-baixos.
- L'*Asset Store* ofereix infinitat de solucions ja creades, models, animacions, etc. i disposa de molts components gratuïts.
- Permet utilizar un *IDE* com *Visual Studio* amb molta facilitat.

4.2 Requeriments de desenvolupament

Requisits Mínims

- *SO*: *Windows 7/8/10*
- Processador: *Core 2 Duo* o superior
- Memòria: 1 GB de RAM
- Gràfics: *GPU* Compatible amb *DirectX11 amb* 512 MB Vídeo RAM
- Emmagatzematge: 100 MB d'espai disponible
- Targeta de so: compatible amb *DirectX*.

Requisits Recomanats

- *SO*: *Windows 7/8/10*
- Processador: *Core 4 Duo* o superior
- Memòria: 2 GB de *RAM*
- Gràfics: *GPU* Compatible amb *DirectX11 amb* 1 GB Vídeo RAM
- Emmagatzematge: 100 MB d'espai disponible
- Targeta de so: compatible amb *DirectX*.

4.3 Eines emprades

Visual Studio 2019

Microsoft Visual Studio és un entorn de desenvolupament integrat (*IDE*, per les sigles en anglès) per a Windows i macOS. ... Visual Studio permet als desenvolupadors crear llocs i aplicacions web, així com serveis web en qualsevol entorn compatible amb la plataforma. NET. Ens permetrà a codificació del joc emprant C#.

Gimp

És un programa d'edició d'imatges digitals en forma de mapa de bits, tant dibuixos com fotografies. És un programa lliure i gratuït. S'ha emprat per editar els diferents *sprites* i crear diferents pantalles.

Audacity

És una aplicació informàtica multiplataforma lliure que es pot usar per a enregistrament i edició d'àudio. S'ha emprat principalment per l'edició de sons.

TexturePacker

Programa que ens facilita la creació de full de *sprites* (*Sprite sheets*). Aquests fitxers *sprites* són col·locats en un full formatat per una graella i serà exportat a un sol fitxer imatge. Aquesta imatge serà carregada en el videojoc i retallada en execució. aquest mètode redueix considerablement el consum de memòria. El programa té una versió gratuïta no comercial

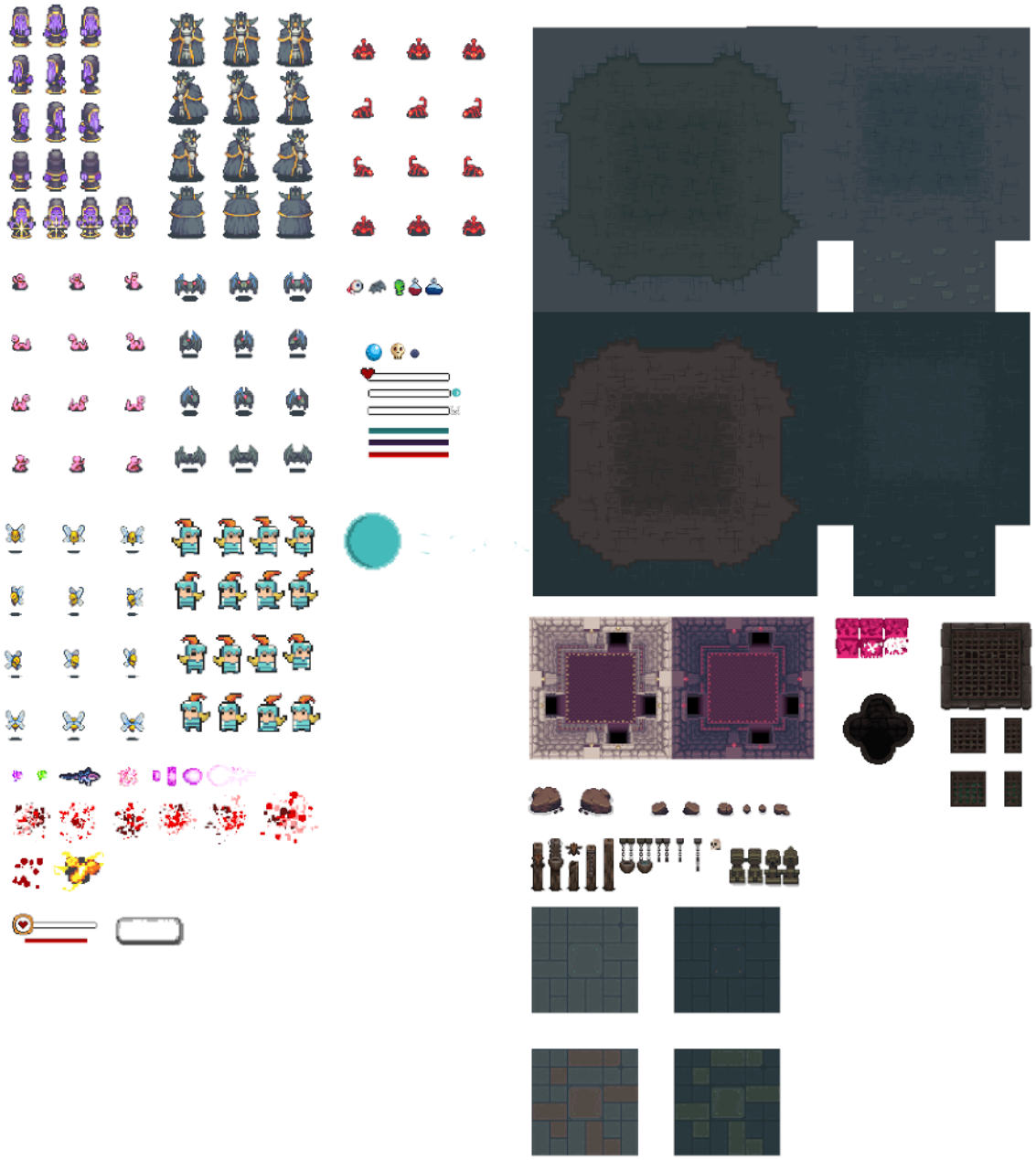
4.4 Recursos emprats

Tots els recursos tant gràfics com de sons usats en el desenvolupament han estat obtinguts de pàgines de distribució de recursos següents:

- **Itch.io:** En aquesta pàgina es permet crear llistes de recursos de manera que s'ha creat una llista amb tots els recursos emprats.
- **OpenGameArt**

4.4.1 Sprites

Els *sprites* emprats es troben a la següent imatge, encara que no tots els sprites animats contennen tots els frames de l'animació perquè no s'haguéssin pogut agrupar en un sol *Spritesheet*, aquesta imatge está a mode de resum perquè es tingui una referència de tots els *sprites* emprats, però per veure'ls tots amb totes les animacions s'ha d'accedir a la llista proporcionada.



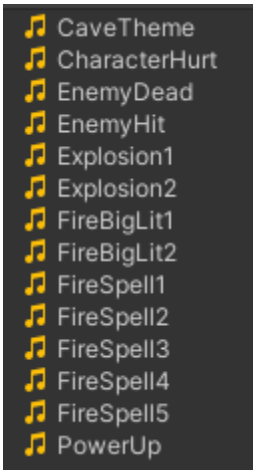
Il·lustració 27 – Versió reduïda de l'sprite-sheet emprat

4.4.2 Sons i música

Els sons s'han extret del recurs d'Itch.io que es pot trobar en aquest enllaç.

Pel que fa a la banda sonora també s'ha extret de la mateixa web encara que és d'un recurs differents. El podem trobar en aquest enllaç.

Com es pot observar només s'ha emprat un subgrup d'aquests recursos.



Il·lustració 27 - Recursos de so i música emprats

4.4.3 Fonts

La font emprada tant pels Menús com pels diferents canvas és la font Minecraft que s'ha descarregat de la web *DaFont*.

Il·lustració 28 - Font Minecraft

4.5 Arquitectura del joc i disseny

4.5.1 Estructura dels components del jugador

Lectura dels Inputs de l'usuari

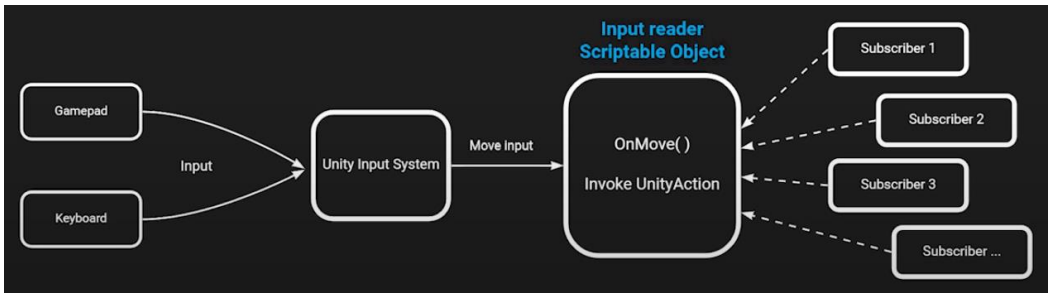
Per aquesta tasca s'han combinat dues funcionalitats noves de Unity: el nou *Input System* i l'*Scriptable Object*.
Aquesta aplicació està basada en el tutorial oficial de *Unity* on apareix una aplicació dels *Scriptable Objects* molt interessant, ja que s'explica

que desar dades no és l'única aplicació d'aquests. Per exemple, són una bona eina per crear capes sobre una funcionalitat de *Unity* ja existent per reutilitzar funcionalitats comunes.

Així es pot convertir un ScriptableObject en un objecte capaç d'escoltar als inputs rebuts pel jugador i invocar les *Unity Actions* associades.

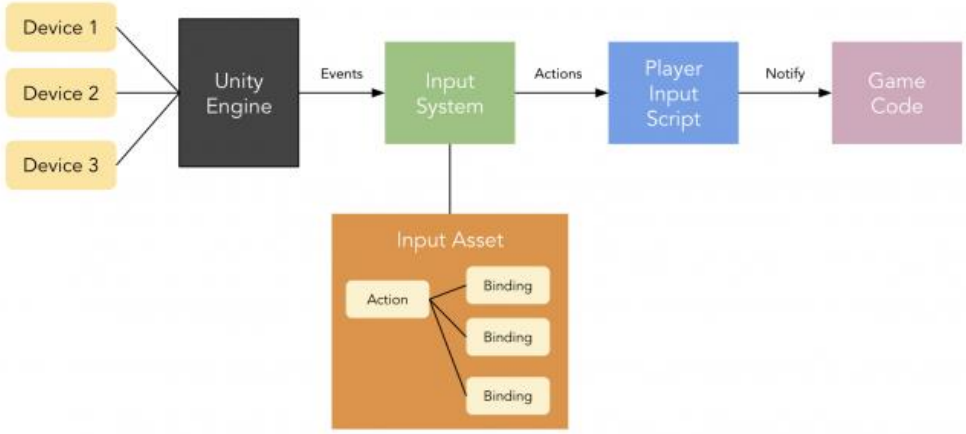
Per exemple, es pot fer que l'*Scriptable Object* implementi una funcionalitat que s'anomena *OnMove*() que escolta al *MoveInput* oferit pel sistema d'inputs de *Unity*. Així quan es produeix una entrada de tipus *MoveInput* de l'usuari, es crida la funció *onMove*() que al seu torn invoca una *UnityAction* de manera que tots els objectes subscrits en aquesta acció executaran els seus *callbacks*.

D'aquesta manera totes les classes que vulguin escoltar algun esdeveniment dels que propaga l'*ScriptableObject* , només necessita tenir una referència a l'*ScriptableObject* i programar la resposta pels inputs als quals se subscriu.



Il·lustració 29 - Combinació de l'Input system i l'Scriptable Object

Per entendre el nou sistema d'inputs va ser de gran ajuda el següent [post]:



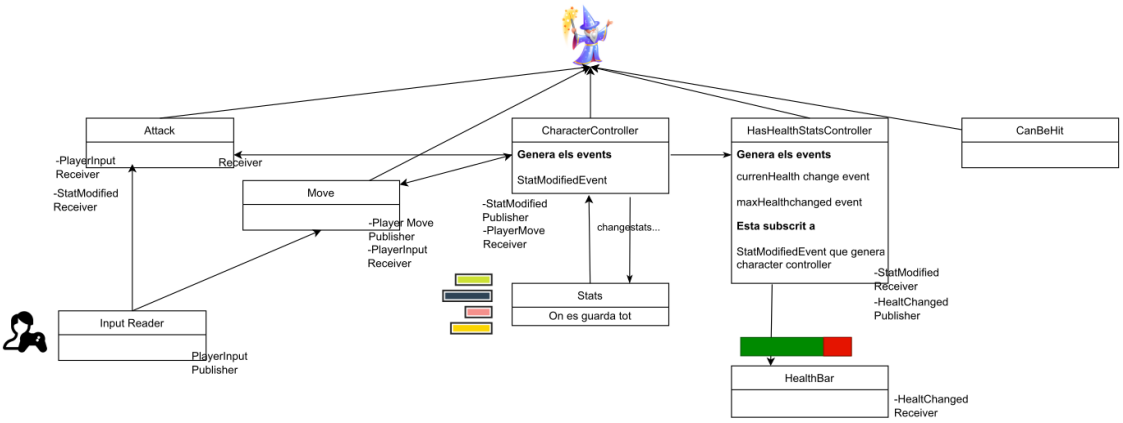
Il·lustració 30 - Flux del nou sistema d'inputs

Player Movement

En primer lloc es va haver de fer la lectura dels inputs de l'usuari i, per això, s'ha emprat el nou sistema d'inputs de *Unity* combinat amb esdeveniments, de manera que cada classe se subscriu a determinats esdeveniments i rep notificacions quan aquests esdeveniments succeeixen. Per exemple, la classe *Move* està subscrita al *MoveEvent*, que al seu torn crida a *l'script CharacterController* perquè es produeixi el moviment indicat (es mou el *RigidBody2D* corresponent). El mateix passa amb *l'script Attack*.

Aquesta pràctica o estructura basada en esdeveniments, no només es veu en el *Move*, també en altres scripts com ara l'*Attack*, el *CharacterController*, etc.

Es pot veure molt bé com s'empren aquests esdeveniments per comunicar les diferents classes que requereix el nostre jugador en el següent diagrama:



Il·lustració 31 - Components del Player

A la imatge no es mostren tots els components del nostre jugador per no fer-lo excessivament complex, és per veure el flux dels esdeveniments i com es gestiona el comportament del personatge.

Player shooting

A més del llançament de projectils, s'ha hagut d'implementar la **recepció de mal** amb la creació de la interfície ***IHasHealth***. També s'ha creat una interfície que han d'implementar els objectes que poden aturar la bala, l'***ICanBeHit***.

Pel fet que s'ha desenvolupat la recepció de mal també s'ha creat la ***HealthBar***, per poder reflectir aquests canvis en la vida, i s'ha fet de manera genèrica perquè aquesta barra de vida se situï sobre el cap d'un *Following Target.*

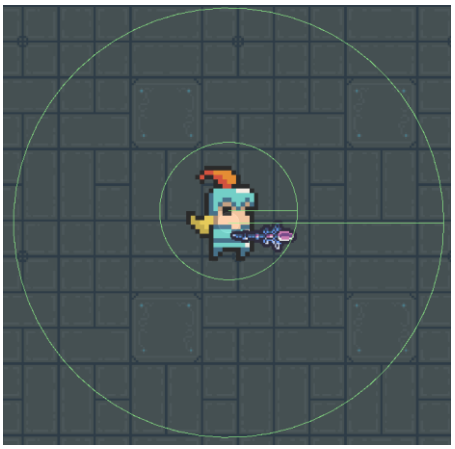
Pel que fa als projectils llançats, aquests s'instancien en funció de les *Stats* del *Player* i amb la rotació de l'arma, que conté el mètode *shoot* i que controla la rotació de l'arma. Així, el projectil té la mateixa rotació que l'arma. Més endavant veurem com per generar un factor aleatori la bala s'instancia amb aquesta direcció i una petita rotació a l'atzar.

Shield

S'ha implementat la possibilitat d'instanciar un escut un cop s'han recollit un nombre suficient d'ànimes, que es recullen amb el soul magnet (explicat al següent apartat). Un cop s'activa aquest escut aquest protegeix del mal al contacte i als projectils enemics i es va debilitant amb l'impacte d'aquests fins que es romp. Mentre està actiu no es poden recollir més ànimes, només un cop aquest estigui desactivat. S'activa amb el botó d'habilitat definitiva o amb la lletra q en cas d'estar jugant amb PC.
Quan un enemic impacte amb aquest es produeix un knockback de l'enemic en la direcció oposada de la direcció d'atac de l'enemic.

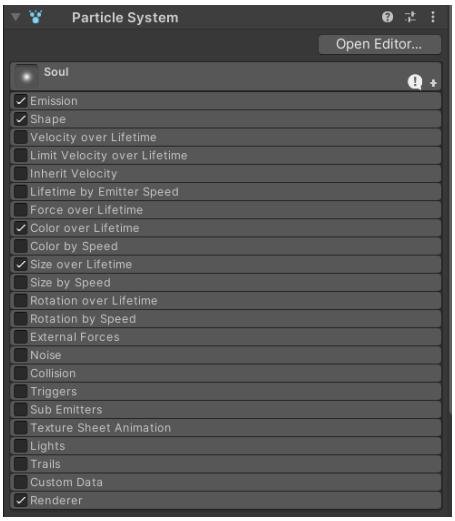
Souls i SoulMagnet

S'ha creat un *Script Soul* que quan col·lideix amb el *soul magnet*, que es representa a la imatge amb el *CircleCollider 2D* de major radi, comença a moure's cap al Player, i quan col·lideix amb el *CircleCollider 2D* amb el radi més petit, anomenat *LootArea*, s'afegeix al comptador de *souls* del jugador.



Il·lustració 32 - Soul Magnet colliders

Per donar una major estètica i representar el moviment de les ànimes s'ha emprat el *Particle System* de *Unity*.



Il·lustració 33 - Particle system associat a una Soul

Il·lustració 34 - Soul en moviment per apreciar l'efecte del particle system

Habiltat definitiva

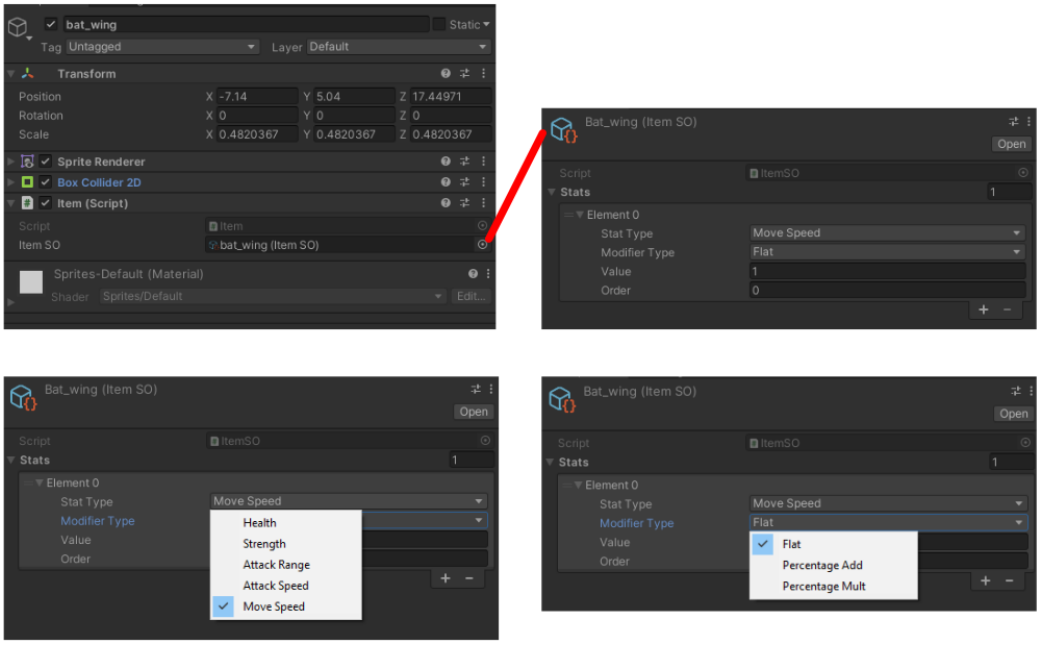
L'habilitat definitiva es pot activar un cop s'hagin obtingut les *kills* necessàries. Consisteix en un tret triple, durant un temps determinat. Simplement s'indica al *CharacterController* que està emprant aquesta habilitat i l'hora de tretar es comprova si s'està emprant l'habilitat definitiva. Si és així s'instancien tres bales, una que és l'original i dues amb una rotació diferent a l'original per generar l'efecte de tret triple.

4.5.2 *Player Stats System*

El desenvolupament d'aquest sistema de les *stats* del jugador s'ha desenvolupat de manera que és la classe *stats* que guarda totes les *stats* del jugador, i és el *characterController* qui accedeix directament a aquestes per canviar-les (es pot veure al primer diagrama). El desenvolupament s'ha fet amb el temps estimat i de manera satisfactòria.

Items stats modifier

Cada ítem disposa d'un llistat de modificadors d'una classe determinada, depenent de l'atribut a què van dirigits i un tipus, depenent de si és un percentatge (+30% de força), un multiplicador (velocitat x 30%) o una suma (*flat modifier*) (vida màxima +3). D'aquesta manera, *l'Stat*, que disposa d'un valor base i un llistat de modificadors, que s'apliquen a aquest valor. Així quan es recull un objecte es crida al mètode equip i s'afegeixen els modificadors associats a aquell objecte. Està molt ben estructurat, ja que cada ítem té associat un *scriptable object* fàcilment configurable on es poden afegir modificadors de diferents tipus de forma fàcil i ràpida:



Il·lustració 35 - Creació d'ítems amb els modificadors

4.5.3 *Level basic SetUp* i *TileMap*

La creació del *TileMap* ha suposat més esforç del que es pensava. Ja que a més de configurar un *TilePalette,* amb el qual es dibuixa el *TileMap*, s'ha creat una forma eficient de gestionar els *colliders*. En un primer moment emprava *TileMap collider*, però les complexes formes que generava pels *colliders* de les diferents *Tiles*, en determinats moments generava *bugs i moviments poc fluids*.

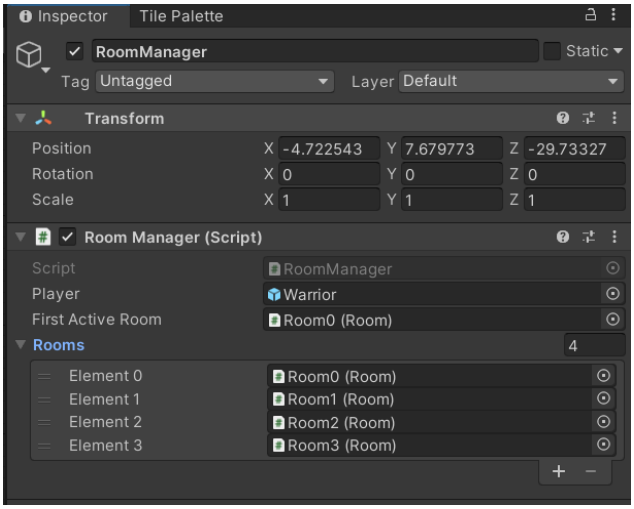
A més, es volia poder oferir la funcionalitat de què determinats objectes fossin per una part no caminables, però per altre havien de poder ser travessats per projectils.

Així, la millor forma de gestionar-ho que s'ha trobat, ha estat fer que el mapa tingui uns límits amb *colliders* amb **l'etiqueta "*wall*"** i afegint i que els que puguin ser col·lidits per projectils implementen la interfície ***ICanBeHit***.

A més, els objectes que actuïn com a parets dins el mapa hauran de tenir també aquesta etiqueta. Aquest sistema també serveix, com veurem més endavant(a l'apartat d'IA enemiga), per determinar els *nodes* caminables pels enemics. Trobem información més detallada sobre aquest desenvolupament en l'apartat Configuració del mapa i IA Enemiga i MapClass.

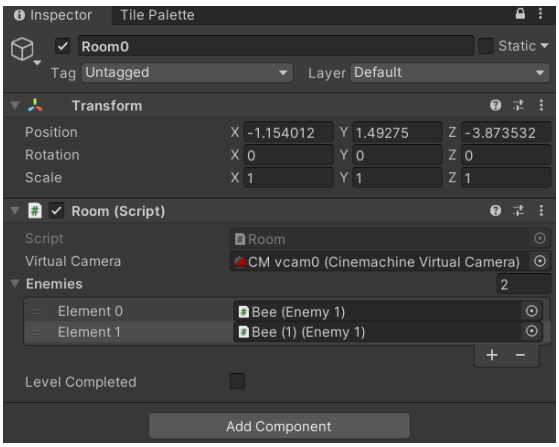
4.5.4 Gestió de les sales i les càmeres

Existeix un *RooManager* que s'encarrega de gestionar els canvis de sala.
En canviar de sala, s'ha de canviar la prioritat de la *virtual camera* associada a les respectives sales, s'han d'activar tots els *gameobjects* associats a la sala i s'han d'activar tots els associats a la següent.



Il·lustració 36 - Room Manager

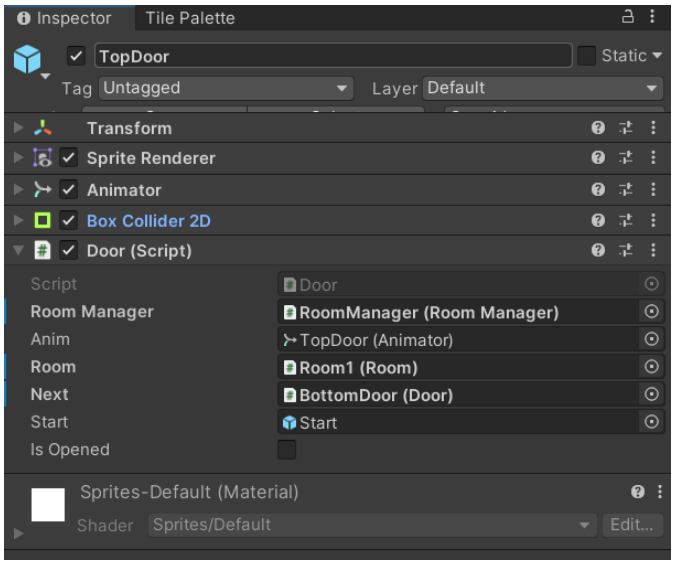
I cada *Room* presenta la següent configuració:



Il·lustració 37 - Configuració Room

I un *MapSetup* que conté les *walls* que delimiten la zona caminable del nivell, els *gameobjects* que permeten als enemics generar el mapa que els permetrà moure's evitant col·lisions, un *polygon collider* que representa els extrems del mapa i que és emprat pel *camera confinner*

de la *virtual camera* associada i les portes que permeten viatjar d'una habitació a la següent:



Il·lustració 38 - Configuració d'una Room

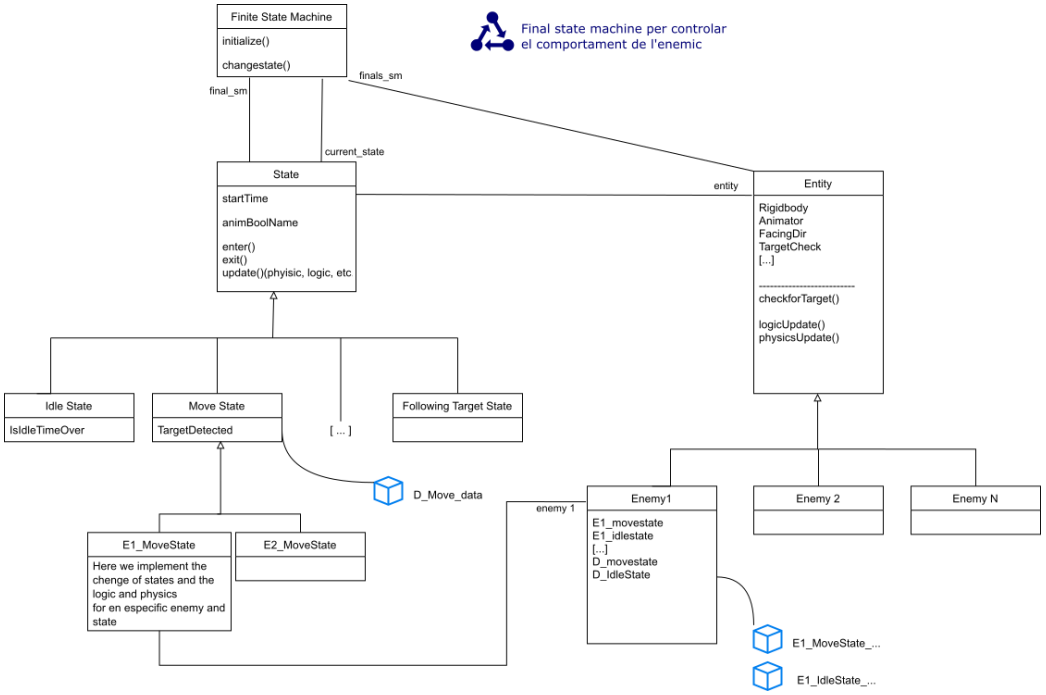
4.5.5 Camera *script*

Pel que fa a l'*script* de la càmera, aquesta segueix al personatge, però canvia la posició quan es canvia la direcció d'apuntat per tal que el jugador tingui un major camp de visió de la zona cap a on apunta. A més s'ha afegit *ScreenShake* per millorar el *geme feel*. Ara bé, per millorar la jugabilitat amb el mòbil el que s'ha fet ha estat que la càmera segueixi el personatge per evitar un excés de moviment.

De manera que l'*script* de la càmera seguirà al *Player* en el cas de jugar amb mòbil, però en el cas de consola i PC s'emprarà un sistema diferent perquè el jugador tingui més visió cap a la zona on està apuntant.

4.5.6 Enemy State Machine

Després d'intentar desenvolupar el comportament de l'enemic emprant una sola classe, la quantitat de condicions a comprovar per determinar el seu moviment eren massa grans. Així, es va decidir implementar una *Final State Machine* perquè cada enemic fos el responsable d'implementar els canvis d'estat segons determinades condicions, i que cada estat hagués d'implementar només la lògica i la física per un estat. L'estructura resultant és la que es mostra, i permet desenvolupar enemics de forma organitzada i molt més àgil.



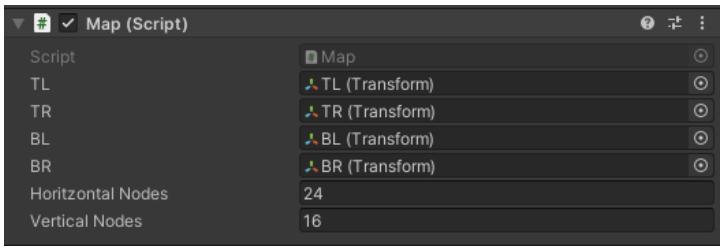
Il·lustració 39 - Diagrama de classes de la màquina d'estats enemiga

Els estats de cada enemic i les transicions es poden veure a nivell conceptual a l'apartat comportament dels enemics vist anteriorment.

4.5.7 Enemy AI(Pathfinding) and Map Class

El desenvolupament de la IA enemiga, ha estat sense cap dubte la part que major temps i esforç ha suposat. És cert que existeixen solucions ja implementades pel *pathfinding* dels enemics. Però s'ha considerat que seria un punt molt positiu per aquest projecte el desenvolupament des de zero d'un sistema que permeti al nostre enemic moure's de forma aleatòria per dins el mapa o perseguir l'enemic esquivant les parets de forma eficient.

Així el que s'ha fet, ha estat crear una classe *Map* que conté les posicions dels extrems del mapa, i el nombre de *nodes* horitzontals i verticals que es volen per generar el mapa:



Il·lustració 40 - Configuració de la classe Map

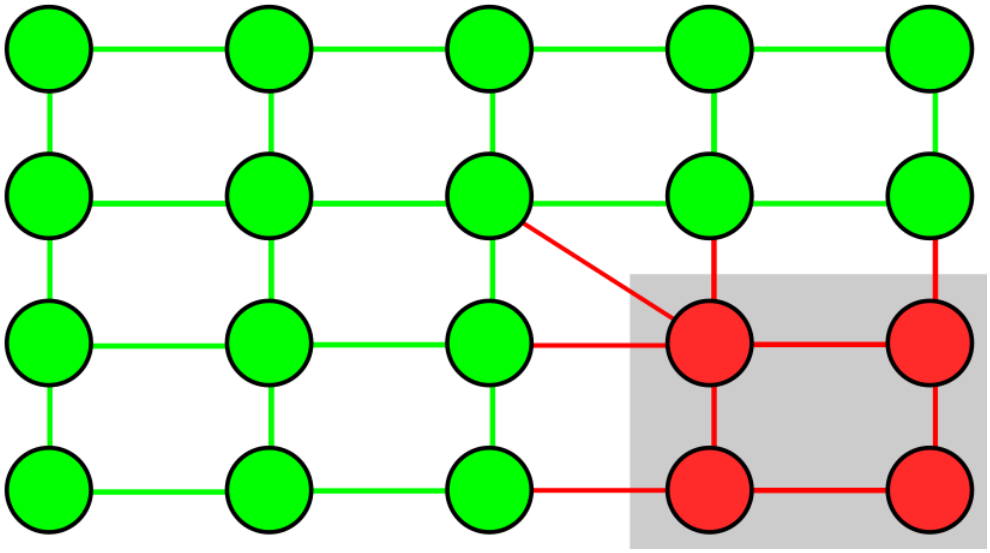
Aquest, té un mètode que permet generar un *grid* amb els *nodes* i connexions vàlides i invàlides, i cada enemic tindrà el seu propi *grid*, ja que un node o una connexió poden ser vàlids per un enemic però no per

un altre depenent del radi del seu *collider*. Un *grid* és simplement un *array* bidimensional de *nodes*. Les estructures per generar el mapa són:

```
public class Node
{
    public Vector2 Position;
    public bool IsValid;
    public float GCost;
    public float HCost;
    4 referencias
    public float FCost => GCost + HCost;

    1 referencia
    public IEnumerable<Edge> Connections => new Edge[] { T, L, B, R, TL, TR, BL, BR };

    public Edge T;
    public Edge L;
    public Edge B;
    public Edge R;
    public Edge TL;
    public Edge TR;
    public Edge BL;
    public Edge BR;

    1 referencia
    public Node(Vector2 position, bool isValid)
    {
        Position = position;
        IsValid = isValid;
    }
}
```

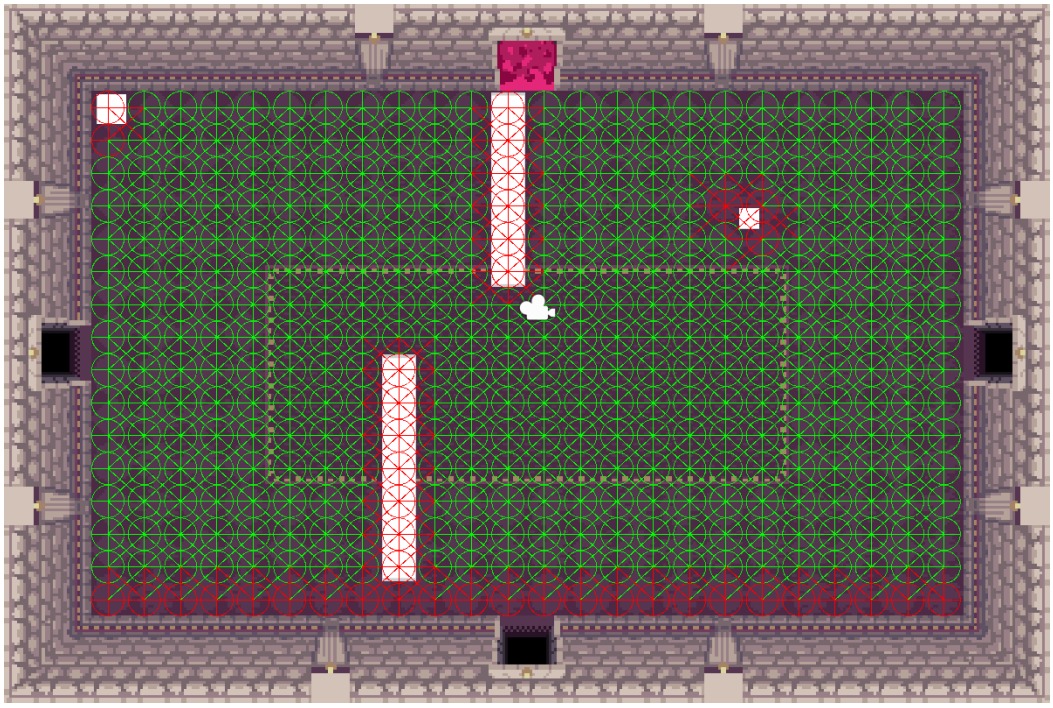
```
public class Edge
{
    public bool IsValid;
    public Node To;
    public Node From;
    public float Cost;

    8 referencias
    public Edge(bool isValid, Node from, Node to, float cost = 1)
    {
        IsValid = isValid;
        To = to;
        From = from;
        Cost = cost;
    }
}
```

Il·lustració 41 - Variables i estructures per generar el mapa



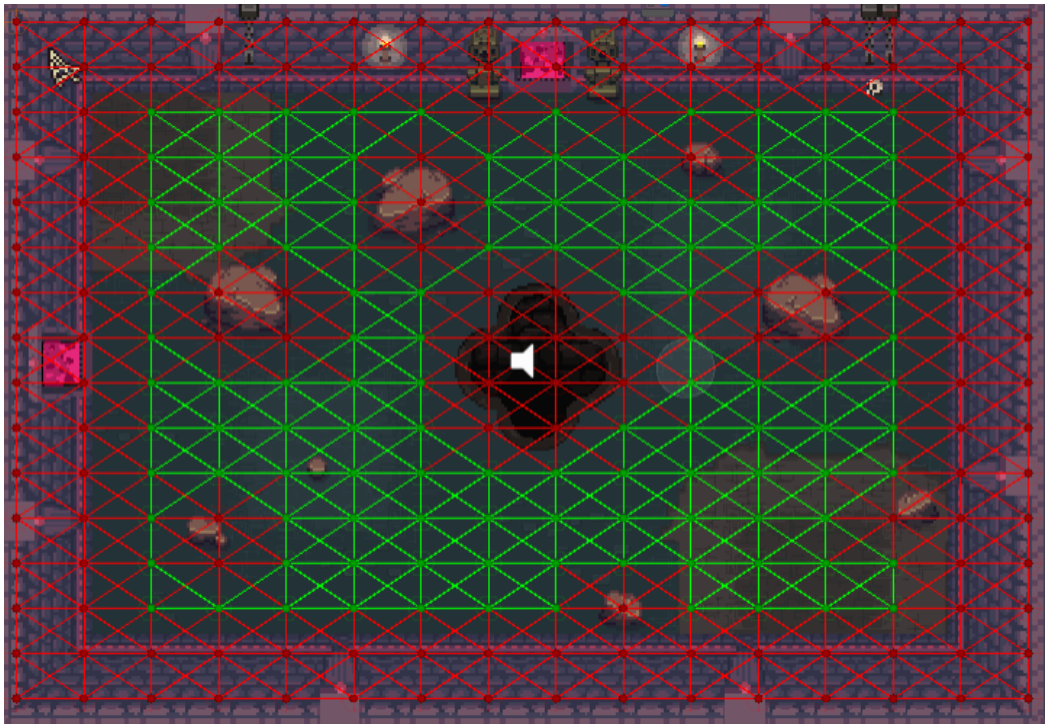
Il·lustració 42 - Concepte de grid

Amb els *Gizmos* s'ha representat gràficament un possible *grid* generat:



Il·lustració 43 - Representació amb Gizmos d'un possible grid generat

Com es pot veure a la il·lustració aquest *grid* pinta nodes i connexions vàlides per un radi concret, per aquest motiu cada enemic té el seu propi *grid*, perquè les connexions i nodes vàlids poden variar d'un enemic a un altre. **Tots els enemics d'una sala tenen el mateix *Map,* però no tots tenen el mateix *grid***.



Il·lustració 44 - Gizmos d'un possible grid generat amb els objectes definitius

Un altre punt destacable és que tant l'estructura *Edge* com *Node*, mostren un cost (dos en el cas de *Node*) que s'empraran per a la implementació de l'algorisme A* que s'ha emprat per tal de construir el camí vàlid més ràpid fins al nostre objectiu, en aquest cas el jugador. No s'adjunta el codi de l'algorisme A* perquè no és un algorisme nou. Però es fa un breu resum del seu funcionament.

Una cerca A* és un algorisme de *pathfinding*. És a dir, en termes més senzills, donat un mapa, des d'un node inicial, quina és la manera més eficient d'arribar a un segon lloc, evitant murs, obstacles i camins sense sortida.

Com hem vist anteriorment, ja tenim els nodes i connexions vàlides. Així, en aquest punt ,'algorisme A* funciona de la següent manera:

- S'afegeixen tots els nodes disponibles per la cerca a la llista de nodes oberts
- Després s'afegeix el node més proper (actual) a la llista de nodes tancats.
- Després ha de decidir a quin node d'entre els més propers s'ha de moure i això ho fa amb el *Path Scoring* que funciona de la següent manera:

  Donarem a cada node una puntuació G + H on:

  G és el cost del moviment* des del punt de partida A fins al quadrat actual. Per tant, per a un quadrat adjacent al punt de partida A, aquest seria 1, però **augmentarà a mesura que ens allunyem del punt de partida**.

  H és el cost estimat del moviment des de la casella actual fins al punt de destinació. Sovint s'anomena heurística perquè encara no en sabem el cost, només és una estimació. En aquest cas com a funció heurística s'ha emprat la distància euclidiana entre els nodes, d'aquesta manera s'assegura que no se sobreestima el cost (si es sobreestima el cost, és possible que el camí generat no sigui el més curt).

  *** El cost del moviment** en aquest joc, pel fet que disposem de 8 direccions, és de 1, o en el cas que sigui un moviment en diagonal d'1,1.

  Una opció encara més vàlida, hauria estat calcular la mida de les diagonals depenent del nombre de nodes horitzontals i verticals, ja que són paràmetres que es poden modificar. Però amb els nodes emprats, i després de fer moltes proves, els costs de moviment 1 i 1,1 per les diagonals mostra resultats molt satisfactoris.

Ara que es coneix el càlcul de la puntuació de cada node (s'anomena F, i és igual a G + H), Es pot veure com funciona l'algorisme A *.

Es trobarà el camí més curt repetint els passos següents:

1. Obtenir el node de la llista oberta que tingui la puntuació més baixa. Anomenem aquesta posició S.
2. Es treu S de la llista de nodes oberts i s'afegeix S a la llista de nodes tancats.
3. Per a cada quadrat T dels nodes adjacents de S:
    a. Si T apareix a la llista tancada: ignoreu-lo.
    b. Si T no és a la llista oberta, s'afegeix i es calcula la seva puntuació.
    c. Si T ja és a la llista oberta, es comprova si la puntuació F és més baixa quan es fa servir el camí actual generat. Si és així, s'actualitza la puntuació i s'actualitza també el pare.

4.5.8 Animacions bàsiques

Per millorar el *gameplay* i evitar excessives complicacions amb les animacions es va decidir que el personatge principal no tingués animació d'atac, ja que l'atac amb vareta màgica presenta moltes complicacions a l'hora d'aconseguir un tret que generi un bon game *feel*. Així es va decidir només implementar les animacions de *idle* i *walk* i fer que l'arma anés rotant en funció de la direcció d'apuntat i tret del Player.

Pel que fa als enemics, la màquina d'estats està preparada per canviar d'animació en funció de l'estat al qual s'entra, però la falta d'*sprites* gratuïts ha fet que aquesta funcionalitat estigui preparada però no s'hagi pogut implementar, de manera que els enemics presenten les animacions de *idle* i *walk*, i en el cas del Wizard també presenta les animacions d'atac.

La resta d'animacions s'expliquen al següent apartat, ja que s'ha implementat amb la finalitat d'oferir un major game *feel*.

4.5.9 GameFeel

S'han aplicat les següents tècniques de *game feel* per millorar la jugabilitat.

Sons

He creat un gestor d'àudio al qual es pot accedir des de qualsevol dels altres *scripts* i permet reproduir música (en bucle) i sons.

Reducció de la vida enemiga

Major cadència de tir

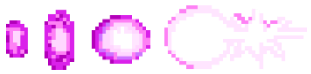
Més enemics

Bales més grans i canvis en la mida dels *colliders*

Els colliders de les bales enemigues s'ha reduït per tal que s'aconsegueixi la sensació d'esquivar bales, mentre que les del jugador tenen el collider fins i tot una mica més gran per tal que faci la sensació de millor punteria.

Bales més ràpides

MuzzleFlash



Il·lustració 45 - Muzzle Flash

Menys precisió en el tret

Per tal de crear una major sensació de retrocés(*recoil*) és una bona tècnica llevar precisió a l'arma i jugar una mica amb el factor aleatori.

Retrocés de l'arma

S'ha programat un *script* amb el qual es pot regular el *recoil*.

Efectes de l'impacte

Il·lustració 46 - Impacte del projectil

Animacions per l'enemic ferit (sprite en blanc) i sang

Il·lustració 47 - Efectes de l'impacte

Retrocés enemic

Per programar el retrocés de l'enemic s'ha creat un estat anomenat *knockback state* al qual s'entra quan rep l'impacte de la bala durant un instant i es canvia la seva direcció (oposada a la bala) i el material perquè es creï l'efecte de *knockback*.

Il·lustració 48 - Knockback de l'enemic

Explosions a l'atzar

Il·lustració 49 - Explosió ràndom

Permanència a l'escena (amb taques de sang)

Il·lustració 50 - Taca de sang

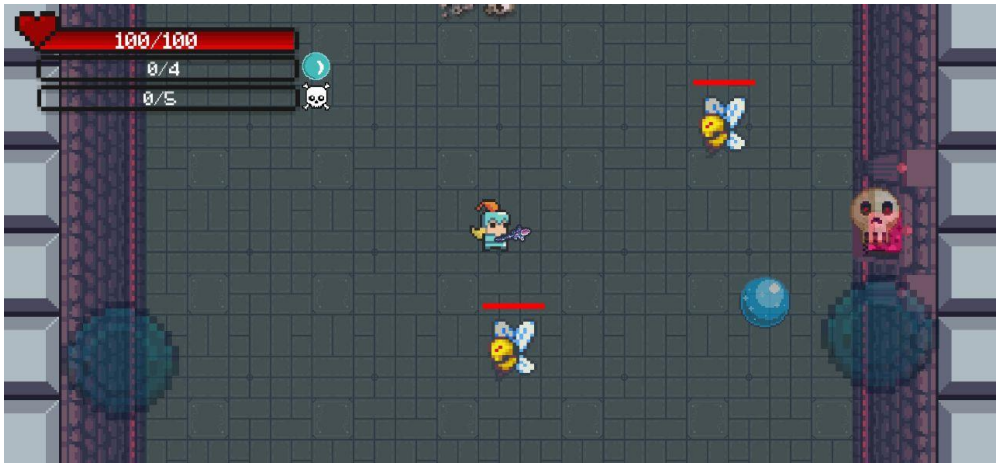
Script de càmera: *camera lerp*, *posició de la càmera* i *screen shake*

També s'ha creat un script de *screen shake* compatible amb les diferents virtual cameras, de manera que escolta els esdeveniments de tretament del *Player* i quan es produeix genera el *shake* a la virtual camera corresponent, és a dir la que està activa en aquella sala.

4.5.10 GamePad

EL *gamepad* obtingut funciona prou bé i empra els estàndards del nou sistema d'inputs de *Unity*, de fet, s'ha emprat una classe pròpia de *Unity* que es diu *On-Screen Stick* al qual s'hi pot associar un *ControlPath*. En el nostre cas hi hem associat els *joystick* dret i esquerre respectivament per al moviment i el tret. A més, s'han afegit dos botons per emprar l'habilitat definitiva i la bombolla. Per prémer pausa simplement s'ha de navegar simulant que es volgués dur a terme l'acció de tornar enrere (Back) amb qualsevol dispositiu *Android*.

Un objectiu que no s'ha pogut assolir per les limitacions de temps, ha estat crear un *joystick* dinàmic o *floating joystick*, ja que millora molt la jugabilitat en dispositius mòbils.



Il·lustració 51 - Controls On-Screen

4.5.11 Utilitats

Physics helper

És una classe estàtica accessible des de totes bandes que permet fer càlculs sobre angles i vectors principalment.

FollowGameObject

Com el seu nom indica permet que l'objecte amb aquest *script* associat segueixi l'objecte que li indiquem amb l'editor o per codi.

Audio Manager

És una classe estàtica accessible des de totes bandes que permet reproduir sons i música.

DestroyOnExit

Script que permet destruir les instàncies de les animacions un cop finalitzades.

InstanceCounter

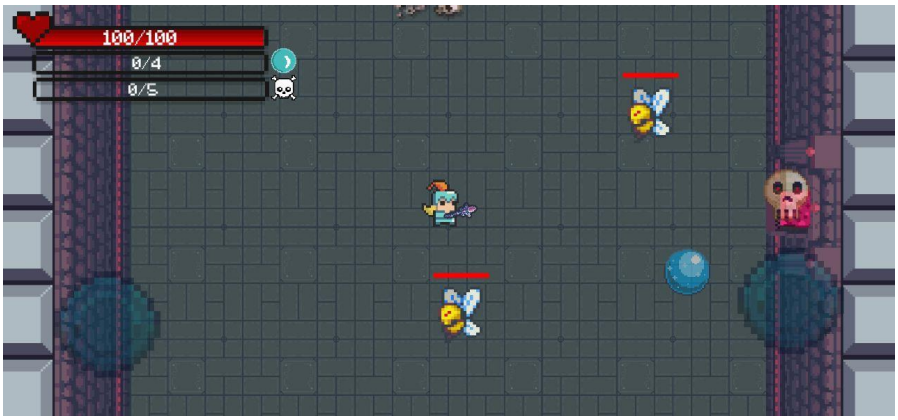
Permet que les taques de sang rebin un nombre que es va incrementant que serà l'ordre de renderitzat dins la capa, d'aquesta manera he solucionat un problema de superposició de *sprites* que generava un efecte òptic no desitjable.

4.5.12 HUD

Una visualització cap-dalt, visualització frontal, visualització *head-up* o simplement *HUD* és una pantalla transparent que presenta informació a l'usuari de manera que aquest no ha de canviar el seu punt de vista per veure aquesta informació.

El *HUD* pot presentar diferents estats. A l'estat inicial es mostra la barra de vida i la barra de *kills* i ànimes. Les dues darreres en un principi estan buides i a mesura que es van aconseguint *kills* i ànimes es van omplint. Quan les barres són plenes apareix un missatge que indica que es pot emprar una bombolla protectora o o l'habilitat definitiva respectivament. Un cop s'activa, el HUD mostra la vida de la bombolla o el temps restant de l'habilitat definitiva.

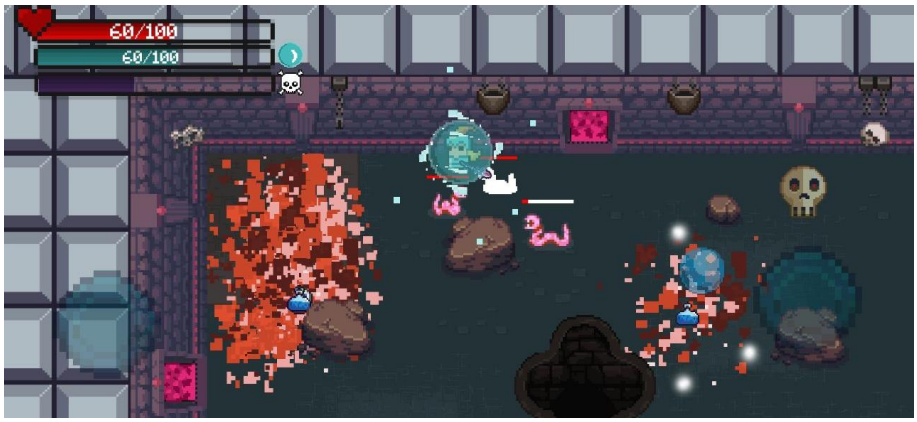
Aquest comportament es pot veure en les següents imatges:



Il·lustració 52 - Hud inicial

Il·lustració 53 - Hud amb on s'han aconseguit kills i souls

Il·lustració 54 - HUD per la bombolla i l'habilitat definitiva carregades



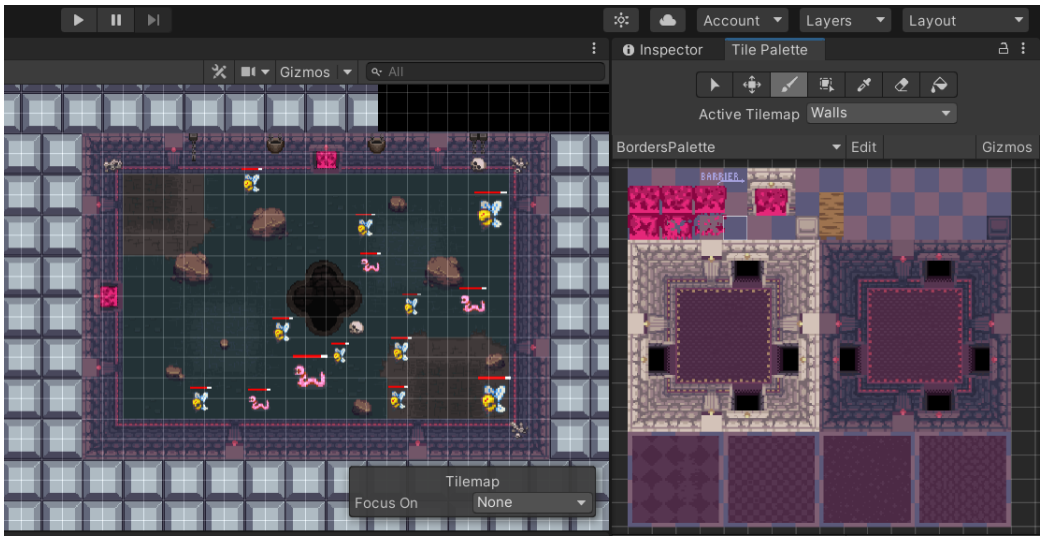
Il·lustració 55 - Hud quan s'estan emprant la bombolla i l'habilitat definitiva

5. Disseny de nivells

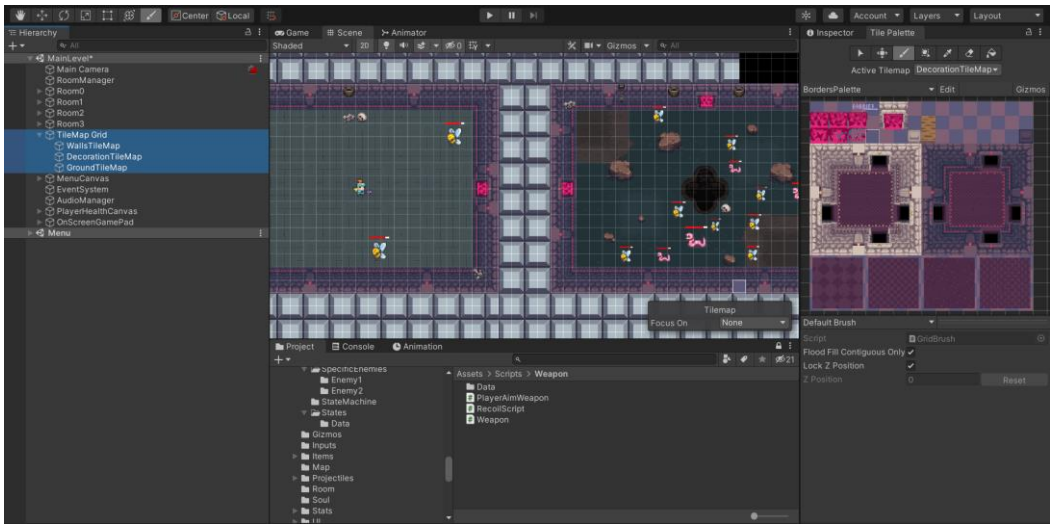
Es pot separar la creació de nivells en dues fases: creació gràfica dels nivells i configuració del Mapa.

5.1 Creació gràfica dels nivells

Per la creació gràfica dels nivells el que s'ha fet ha estat crear un ***TilePalette*** i un ***TileMap***. Quan es crea un *TileMap*, el component Grid es crea automàticament i s'emparenta amb el *TileMap* i actua com a guia quan es col·loquen els *Tiles* al *TileMap*. Aquests *Tiles* se seleccionen del TilePalette que ofereix certes funcionalitats i eines de dibuix amb *Tiles* que faciliten molt aquesta tasca.

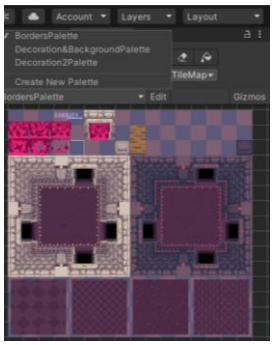


Il·lustració 56 - Grid i TilePalette



Il·lustració 57 - Estructura del Grid

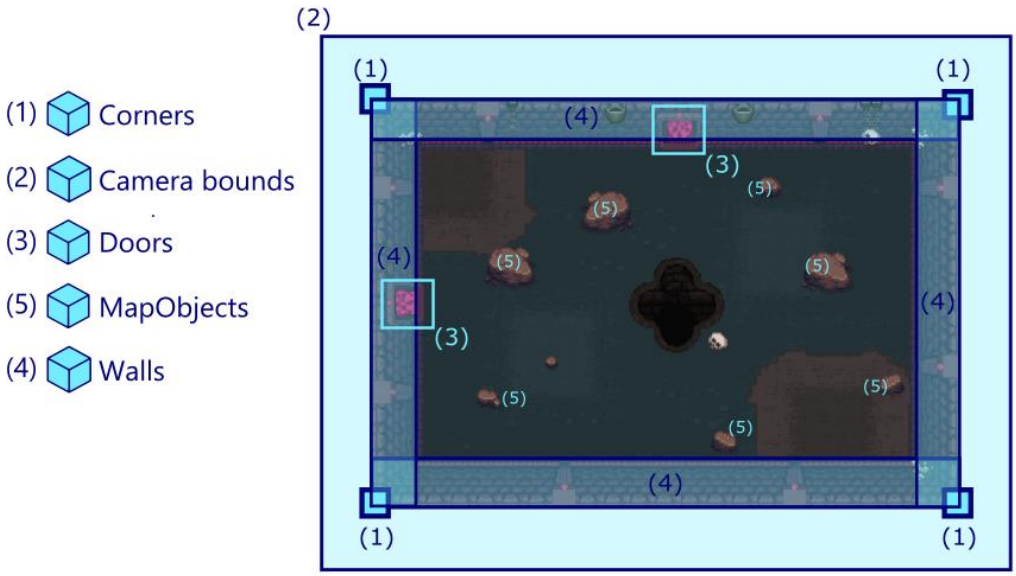
Es pot observar com per tenir una millor organització i facilitar el pintat s'han creat tres *TileMaps* diferents un pel fons, un per les parets i un per la decoració. A més per tenir les paletes més curioses s'han creat diferents *TilePalettes* segons el contingut d'aquestes.



Il·lustració 58 - Diferents TilePalettes

5.2 Configuració del Mapa

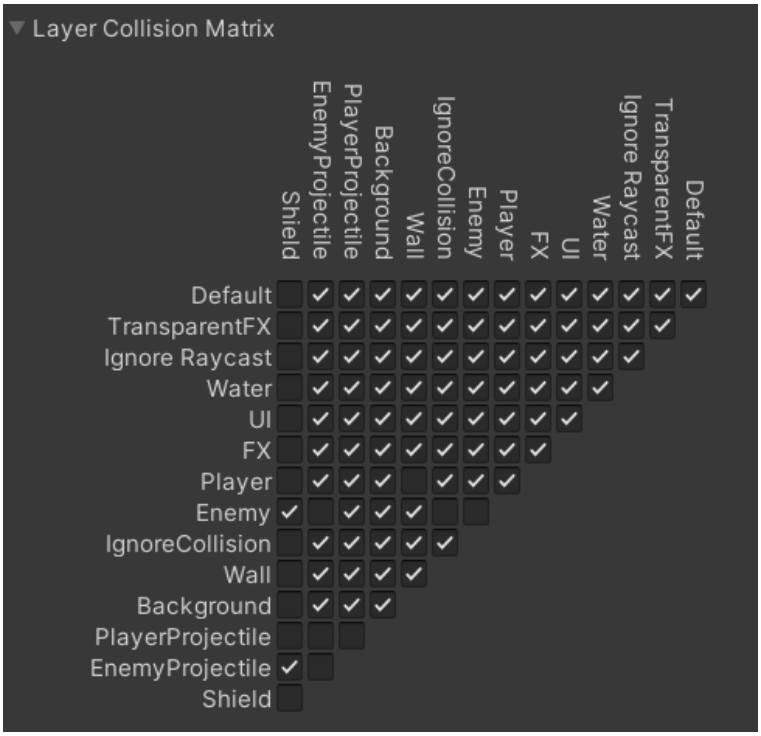
Per tal que els elements que requereixen interactuar amb el mapa (jugador, enemics, càmera, bales, etc.) ho puguin fer, s'han creat una sèrie de *gameobjects* que els permeten saber quins són els límits del mapa, o quin tipus d'interacció hi ha d'haver amb els objectes que hi trobaran. Conté les walls que delimiten la zona caminable del nivell, els *gameobjects* que permeten als enemics generar el mapa que els permetrà moure's evitant col·lisions, un *polygon collider* que representa els extrems del mapa i que és emprat pel *camera confinner* de la *virtual camera* associada i les portes que permeten viatjar d'una habitació a la següent:



Il·lustració 59 - Configuració del mapa

Per tal de millorar el treball d'interacció d'objectes s'han usat capes i *Tags* associats als diferents *gameobjects* per determinar les interaccions quan es generen col·lisions o es produeixen determinats esdeveniments, un

exemple seria la generació del *grid* pel qual es pot moure l'enemic, ja que crea un grid evitant que les connexions del grid es creuin amb *gameobjects* que tenen l'etiqueta *wall*.
A més, per evitar la detecció de determinades col·lisions s'ha emprat la *Collision Matrix*, que ha quedat configurada de la següent forma:



Il·lustració 60 - Matriu de col·lisions

5.3 *RoomManager*, canvis d'escena i *VirtualCameras*

Existeix un *room manager* que s'encarrega de gestionar els canvis de sala, que només es pot realitzar quan tots els enemics d'una sala han estat eliminats.
En canviar de sala, s'ha de canviar la prioritat de la *virtual camera* associada a les respectives sales, s'han de desactivar tots els *gameobjects* associats a la sala i s'han d'activar tots els associats a la següent. Per gestionar el canvi de sala també es canvia la prioritat de la càmera de la nova sala perquè aquesta sigui la *virtual camera* activa (la que té més prioritat). Així cada sala tindrà la seva *virtual camera* associada.

Pel que fa als enemics, aquest s'activen quan ha acabat la transició de la càmera perquè el jugador tingui temps de visualitzar la nova sala.

5.4 Configuració de nous nivells

1. Dibuixar-lo amb el *TilePalette.*
2. Afegir els objectes que funcionen com a delimitadors (*walls*, *polygon collider* per la càmra, *corners* i portes).
3. Afegir els *gameobjects* decoratius.
4. Afegir els enemics.

5. Afegir la nova sala al *RoomManager*.
6. Crear una *virtual camaera* que associarem a la nova sala i afegir-hi el jugador al camp *Follow*.

5.5 Descripció dels nivells

És cert que per les limitacions de temps només s'han pogut desenvolupar 3 sales, ara bé, existeix molta facilitat per crear nous nivells. Ja que com s'ha mostrat abans el procés és bastant simple.

La primera sala és introductòria que permet veure les dinàmiques bàsiques del joc.
La segona sala conté diferents enemics que persegueixen al jugador, però encara no es veu el tret de projectils enemics.
La tercera sala ja conté el primer *boss*, concretament en trobem dos, mostra major dificultat perquè s'han d'eliminar els enemics que ens persegueixen i s'ha d'anar amb compte amb el tret de 8 direccions del primer *boss*.
La darrera sala inclou els rats-penats que també fan mal par contacte i el *boss* final, que fa un tret en espiral bastant potent.

6. Manual d'usuari

6.1 Requisits tècnics de programari

6.1.1 Mòbil

Els requisits del dispositiu mòbil de gamma més baixa amb què s'ha executat el joc sense cap problema són els següents:

- Sistema operatiu 4.1 o superior
- CPU: *Intel Quad Core*
- Memòria *RAM* mínima d'1 GB
- Emmagatzematge de 600 MB

6.1.2 PC

El requisits del dispositiu de gama més baixa amb que s'ha executat el joc sense cap problema en el cas de mòbil són els següents:

- Processador: *AMD* (x86) o *Intel*
- Memòria *RAM* d'1 GB
- Espai d'emmagatzematge de 2 GB
- Sistema operatiu *Windows* 7 o superior

6.2 Instruccions

Il·lustració 61 - Controls

7. Conclusions

Durant el procés de desenvolupament d'aquest videojoc s'han arribat a diferents conclusions que s'agrupen en les diferents categories que apareixen a continuació.

7.1 Lliçons i aprenentatges

El desenvolupament d'aquest projecte ha suposat la possibilitat d'aplicar molts dels conceptes vists durant el grau, des de disseny d'estructures de dades, fins a enginyeria del software. A més, crec que ha suposat un impuls molt gran a l'hora de guanyar fluïdesa programant de manera que a nivell d'habilitats de programació ha estat una experiència molt enriquidora.

A nivell personal, sempre havia tingut dubtes sobre la meva orientació professional, ja que les carreres relacionades amb el disseny gràfic sempre m'han cridat l'atenció, però amb aquest desenvolupament considero que he trobat una àrea on l'art i la programació es combinen a la perfecció i que per tant s'ajusta al meu perfil.

Vull destacar que un dels motius pels quals he anat afegint més funcionalitats de les que s'havien plantejat inicialment ha estat l'enfocament que s'ha aplicat en aquest projecte, és a dir, no només s'ha considerat el TFG com una tasca que s'havia de completar per obtenir el títol, també s'ha enfocat el treball final com un projecte personal al qual s'hi han reflectit les meves habilitats artístiques i de programació de la millor forma possible. És una experiència molt enriquidora veure en acció les funcionalitats que s'han anat creant durant el cicle de vida d'aquest projecte.

Tot i haver assolit els desenvolupaments planificats d'es d'un principi, en alguns moments s'han hagut de prendre decisions per reduir l'abast de la solució perquè al cap i a la fi es tracta d'un projecte amb una durada concreta que s'ha de complir.

La fase inicial és la més complicada perquè per comprovar les noves funcionalitats se'n necessiten programar d'altres. Un exemple és el *Player shooting*, que també requereix implementar un sistema de vida, que al mateix temps requereix implementar un barra de vida, etc. A més, l'experiència viscuda, m'ha demostrat que seguint les bones pràctiques, la corba per implementar noves funcionalitats no és lineal, ja que si es pensa una solució escalable, el temps de desenvolupament inicial és una mica major, però aquest temps es recupera a mesura que va avançant el projecte. Això s'ha comprovat amb el nou sistema d'inputs que va requerir un temps d'investigació i el disseny d'un input genèric, però que al final ens ha permès exportar el joc a diferents plataformes tan sols canviant la plataforma de la *build*.

Per altra banda, la fase final pel fet que ja es té un nombre molt major d'iteracions i funcionalitats implementades, permet obtenir un producte molt

més atractiu i divertit i afegint petits canvis es poden millorar moltíssim les mecàniques del joc i el *game feel*.

El producte obtingut supera les expectatives personals, ja que he pogut iterar molt sobre la solució i s'ha anat millorant el resultat obtingut fins a arribar a un resultat satisfactori.

Vull destacar la necessitat de crear tests en el desenvolupament de videojocs, en el meu cas no s'han programat però és una tasca pendent. S'han viscut moments en què quan es provava una nova funcionalitat es comprovava que s'havien espatllat altres funcionalitats, però no es tenia certesa d'amb quin desenvolupament s'havia produït aquest canvi de comportament no desitjat. Aquestes situacions es podrien evitar programant tests.

7.2 Reflexió sobre l'assoliment dels objectius

A escala general els objectius s'han assolit de manera satisfactòria, i fins i tot s'han anat complint nous objectius que han considerat que afegirien molta jugabilitat. Per exemple, moltes de les recomanacions del consultor, han suposat la creació de nous objectius, la majoria dels quals s'han pogut assolir amb èxit.

L'únic objectiu inicial que no s'ha complit ha estat les animacions per la màquina d'estats enemiga, ja que la màquina d'estats està preparada perquè quan l'enemic canvia d'estat, aquest passa per paràmetre el nom de l'animació d'aquell estat, però la falta dels *sprites* per aquestes animacions ha fet que es deixi aquest desenvolupament pendent de complir.

Així, tot i que la majoria dels objectius inicials s'han complit, durant el procés i les iteracions n'han aparegut de nous i no tots s'han pogut complir.

Aquests objectius no assolits són els següents:

- Fer que el Lich no surti de l'estat player detected, de manera que l'atac sigui continu i que a més de tretar, en aquest estat vagi instanciant rats penats enemics i cada cert temps recuperi vida.

- Disposar d'uns controls *on-screen* amb millor jugabilitat per dispositius mòbils.

- Disposar d'una *UI* agradable. Tot i que s'ha aconseguit una *UI* bàsica es considera prioritari millorar-la abans de publicar aquest joc a la *PlayStore* o altres plataformes.

- Disposar de més nivells per allargar el *gameplay* i augmentar el nombre d'enemics que tinguin nous comportaments per sorprendre el jugador.

7.3 Seguiment de la planificació i la metodologia

La metodologia s'ha seguit de manera correcta, és cert que l'ús de tests per les diferents funcionalitats hagués permès un major nombre d'iteracions, però la falta de temps per programar aquests tests ha fet que les iteracions sobre el producte en determinats moments hagin obligat a revisar iteracions anteriors, fet que ha conduït a dedicar temps a corregir funcionalitats que ja estaven implementades.

A més, a causa de a la inexperiència en aquesta àrea, ha estat molt complicat complir la planificació, ja que moltes funcionalitats poden no oferir el resultat esperat o requerir desenvolupaments extres que no s'havien tingut en compte.

Així, les principals desviacions en la planificació han estat l'**IA enemiga**, que ha requerit una setmana més de l'estimació inicial. Errades en la **màquina d'estats** que ha suposat 5 dies de retard. Les **recomanacions del consultor** a cada PAC que han requerit millores que no es tenien en compte en la planificació inicial, que han suposat quasi una setmana de retard en total.

Aquest temps s'ha recuperat dedicant més hores de les que es pensava en un principi, desplaçant determinats desenvolupaments a l'etapa final(fet que ha suposat un menor temps per desenvolupar la memòria i el vídeo de presentació) i reduint el nombre de dies dedicat a les animacions i sons dels personatges i enemics.

7.4 Línies de treball futures

Les limitacions de temps han provocat que moltes de les idees o possibilitats de millora que han aparegut durant les diferents iteracions del projecte s'hagin hagut de descartar per motius de temps, són les següents:

7.4.1 Principals

- ***Floating* o *dynamic GamePad*** per oferir un major *gamefeel* amb dispositius andorid. Un molt bon exemple a seguir és el *gamepad* de *Brawl Stars*:

Il·lustració 62 - Gamepad Brawlstars

- **Generació procedural de mapes**. Amb el resultat satisfactori de la configuració de mapes obtingut, seria un desenvolupament relativament senzill escriure les regles per dur a terme la generació procedural. A més, d'aquesta manera es podria allargar molt el *gameplay*.

- **Sistema d'inventari**. El sistema de *Stats* implementat només permet emprar ítems d'un sol ús, però afegir un inventari que permeti canviar les armes o tenir ítems guardats oferiria molta qualitat al resultat final.

- ***Combat Core***. Un cop generat l'inventari, es preveu la necessitat d'implementar una classe que faci de *Core*, a la qual s'hi passi un objecte atacant amb l'estat de l'entitat que ataca (conjunt d'ítems que té, l'arma que té si està emprant alguna habilitat, etc.) i l'estat de l'entitat que rep mal, i que al mateix temps aquesta classe tingui accés a l'estat de la partida per tal que la classe retorni la quantitat de mal que ha de rebre l'entitat atacada i altres possibles dades que poden ser d'interès tant per les entitats involucrades com per l'estat de la partida. Seria una espècie de calculadora que agrupa totes les comprovacions necessàries i simplifiqui el treball de comprovacions de les entitats involucrades en el combat.

7.4.2 Secundàries

- Afegir nous personatges principals.
- Afegir nous enemics amb comportaments diferents.
- Afegir noves animacions i sons per donar un major *gamefeel*
- Millorar l'script de càmera per mòbil

Un cop aconseguits aquest objectius es pretén publicar el joc per *Android*, *IOS*, *PC, Xbox, PS4 i Nintendo Switch*.

Finalment, cal destacar durant aquest projecte s'han generat una gran quantitat de solucions a problemes que podrien ser de molta ajuda per la comunitat i que en un futur es podrien materialitzar en forma de tutorials o cursos a plataformes d'ensenyament.

8. Glossari

GameObject: són els objectes fonamentals de *Unity*. No aconsegueixen molt en si mateixos, però actuen com a contenidors de components, que implementen la funcionalitat real.

GamePad: Fa referència als controls *onScreen* que ofereix un videojoc per mòbil.

RigidBody2D: És el component principal que permet el comportament físic per a un objecte.

Collider: Els components *Collider* defineixen la forma d'un objecte per als propòsits de col·lisions físiques.

Asset: Recurs que pot ser utilitzat en un joc o projecte.

Git: Git és un programari de control de versions, pensant en l'eficiència, la fiabilitat i compatibilitat del manteniment de versions d'aplicacions quan aquestes tenen un gran nombre d'arxius de codi font.

Game Feel: és la sensació intangible i tàctil que s'experimenta en interactuar amb els videojocs.

Stat: Sinònim de característica, habilitat o atribut d'un personatge.

IA: Intel·ligència artificial.

GamePlay: Conjunt d'accions que pot realitzar un jugador per interactuar amb el joc o la forma en què aquest interactua amb el mateix jugador

Sprite: qualsevol imatge de mapa de bits present en pantalla, generalment referint-se als personatges de el joc.

Sprite-sheet: És una imatge que consisteix en diverses imatges més petites (*sprites*) o animacions. La combinació de petites imatges en una imatge gran millora el rendiment del joc, redueix l'ús de memòria i accelera l'inici i el temps de càrrega del joc.

UI: Acrònim del terme anglès *User Interface* (Interfície d'Usuari).

Boss: Enemic o monstre controlat per la CPU que trobem a la fi de les pantalles o fases i que acostuma a ser més difícil de derrotar que la resta d'enemics.

TilePalette: Eina de dibuix que permet seleccionar i pintar els *tiles* seleccionats.

TileMap: Graella (*grid*) formada per cel·les regulars que poden contenir una imatge. Permet pintar nivells mitjançant *tiles* i eines de pinzell i definir regles sobre el comportament de les *tiles*.

Tile: Imatge petita de forma regular que s'empra per omplir el fons de l'escena.

Collision Matrix: Defineix quins *GameObjects* poden xocar amb quines capes.

Tags: És una paraula de referència que pot assignar a un o més *GameObjects*. Per exemple, pot definir *Tags* de "Jugador" per als personatges controlats pel jugador i una *Tag* d'"Enemic" per als personatges no controlats pel jugador.

Joystick: Dispositiu de control que permet el jugador interactuar amb el joc.

Screen shake: és una manera popular d'afegir una sensació dinàmica a un joc. Consisteix a desplaçar ràpidament la càmera en petits increments durant un breu temps.

Knockback: cop que empeny a l'adversari cap enrere, generalment deixant-ho també inconscient.

MuzzleFlash: És una lluentor que és visible abans que la bala surti del canó

Pathfinding: Capacitat de la intel·ligència artificial d'un personatge o un altre objecte mòbil d'un joc de trobar el camí més curt entre dos punts i arribar al seu destí.

Grid: Fa referència a una estructura en forma de quadrícula o graella.

Gizmos: s'utilitzen per dibuixar formes a la vista d'escena. Es poden utilitzar aquestes formes per dibuixar informació addicional sobre els *GameObjects*, per exemple el rang de detecció o els *colliders*.

Particle system: és un sistema d'efectes de partícules on es poden simular líquids en moviment, fum, núvols, flames, encanteris màgics i una gran quantitat d'altres efectes.

Health Bar: Barra de vida.

CharacterController: Script principal del jugador

Engine: Sistema dissenyat per a la creació de videojocs que aglutina un conjunt d'aplicacions necessàries per al seu desenvolupament. La seva funció principal és dotar el joc d'un motor gràfic per al renderitzat dels models i animacions que formen el videojoc, encara que sovint els motors incorporen un entorn de desenvolupament format per diverses eines per facilitar als desenvolupadors la feina, com un motor de físiques o un motor de col·lisions.

Framework: És un conjunt estandarditzat de conceptes, pràctiques i criteris per enfocar un tipus de problemàtica particular que serveix com a referència, per enfrontar i resoldre nous problemes d'índole similar.

Dungeons: Nivell o escenari que es caracteritza per les seves formes laberíntiques i que requereix l'exploració del jugador per avançar, generalment sent imprescindible la recerca de claus o altres objectes que permetin l'avanç per certes zones, la resolució de puzles i de l'aniquilació d'un gran nombre d'enemics, en moltes ocasions culminant en un gran cap final que dona per conclosa la masmorra.

Cooldown: Temps d'espera per tornar a fer servir una habilitat.

Script: Fragment de codi que implementa alguna funcionalitat.

Scriptable Object: És una classe que permet emmagatzemar grans quantitats de dades compartides.

Indie: Videojocs desenvolupats per grups reduïts d'individus o petites empreses

Repositori: Un directori o espai d'emmagatzematge on es troba el projecte.

Generació procedural: Generat mitjançant un algoritme aleatori en lloc de manualment o de manera sempre idèntica.

On-screen: En pantalla

C#: Llenguatge de programació.

IDE: Aplicació informàtica que proporciona serveis integrals per facilitar el desenvolupament.

*Moltes d'aquestes definicions s'han obtingut de la pàgina web *Wikipedia* i *GamerDic* citades a la bibliografia.

9. Bibliografia

Pàgines Web

[1] *El videojuego en España factura 1.747 millones de euros en 2020 y aumenta su base hasta casi los 16 millones de usuarios*. europapress.es. (2021). Retrieved 4 June 2021, from https://www.europapress.es/portaltic/videojuegos/noticia-videojuego-espana-factura-1747-millones-euros-2020-aumenta-base-casi-16-millones-usuarios-20210429122400.html.

[2] *GamerDic | Listado de términos sobre videojuegos*. Gamerdic.es. (2021). Retrieved 4 June 2021, from https://www.gamerdic.es/buscar/?q=ide.

[3] *Glossary of video game terms - Wikipedia*. En.wikipedia.org. (2021). Retrieved 4 June 2021, from https://en.wikipedia.org/wiki/Glossary_of_video_game_terms.

[4] *Interview: Jan Willem Nijman On Nuclear Throne's "Feel"*. Rock Paper Shotgun. (2021). Retrieved 4 June 2021, from https://www.rockpapershotgun.com/interview-jan-willem-nijman-on-nuclear-thrones-feel.

[5] *Introduction to A* Pathfinding*. raywenderlich.com. (2021). Retrieved 4 June 2021, from https://www.raywenderlich.com/3016-introduction-to-a-pathfinding.

[6] *It's all been done before - DESK Magazine*. The DESK Magazine. (2021). Retrieved 4 June 2021, from https://vanschneider.com/blog/its-all-been-done-before/.

[7] *New Unity Input System: Getting Started*. raywenderlich.com. (2021). Retrieved 4 June 2021, from https://www.raywenderlich.com/9671886-new-unity-input-system-getting-started.

[8] *Quick start guide | Input System | 1.0.2*. Docs.unity3d.com. (2021). Retrieved 4 June 2021, from https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/QuickStartGuide.html.

[9] *State · Design Patterns Revisited · Game Programming Patterns*. Gameprogrammingpatterns.com. (2021). Retrieved 4 June 2021, from https://gameprogrammingpatterns.com/state.html.

[10] Technologies, U. (2021). *Unity - Manual: ScriptableObject*. Docs.unity3d.com. Retrieved 4 June 2021, from https://docs.unity3d.com/Manual/class-ScriptableObject.html.

[11] Technologies, U. (2021). *Unity - Manual: System requirements for Unity 2020 LTS*. Docs.unity3d.com. Retrieved 4 June 2021, from https://docs.unity3d.com/Manual/system-requirements.html.

[12] Valdez, N., Valdez, N., & Valdez, N. (2021). The Rule of the Loop (Part 1) - Balangay Entertainment. Balangay Entertainment. Retrieved 4 June 2021, from http://www.balangay.games/2016/05/26/the-rule-of-the-loop-part-1/.

Vídeos i canals de *Youtube*

[13] *Basic Top Down Camera Look Movement! | #Unity Tutorial*. (2021). [Video]. Retrieved 4 June 2021, from https://www.youtube.com/watch?v=LFe017d-S58&list=PLtgzDi-lvNPuYx4Khv3m6mGUGSvTosIr7&index=5&t=599s&ab_channel=TheZBillDyl.

[14] Code Monkey. (2021). *Code Monkey Channel* [Video]. Retrieved 4 June 2021, from https://www.youtube.com/channel/UCFK6NCbuCIVzA6Yj1G_ZqCg.

[15] Garden, D. (2021). *Jan Willem Nijman - Vlambeer - "The art of screenshake" at INDIGO Classes 2013* [Video]. Retrieved 4 June 2021, from https://www.youtube.com/watch?v=AJdEqssNZ-U&t=640s&ab_channel=DutchGameGarden.

[16] Palaje, M. (2021). *Make Games The Vlambeer Way - Resource Drop #3 [Game Design And Development]* [Video]. Retrieved 4 June 2021, from https://www.youtube.com/watch?v=cBPYCmjDNkM&list=PLtgzDi-lvNPuYx4Khv3m6mGUGSvTosIr7&index=3&ab_channel=MatthewPalaje.

[17] Unity. (2021). *Game architecture with ScriptableObjects | Open Projects Devlog* [Video]. Retrieved 4 June 2021, from https://www.youtube.com/watch?v=WLDgtRNK2VE&t=255s&ab_channel=Unity.

[18] *Unity 3D - Make a Basic AI State Machine [How To]*. (2021). [Image]. Retrieved 4 June 2021, from https://www.youtube.com/watch?v=PaLD1t-kIwM&list=PLy78FINcVmjBs__nqXOnyCEDbx9GpkbD6.

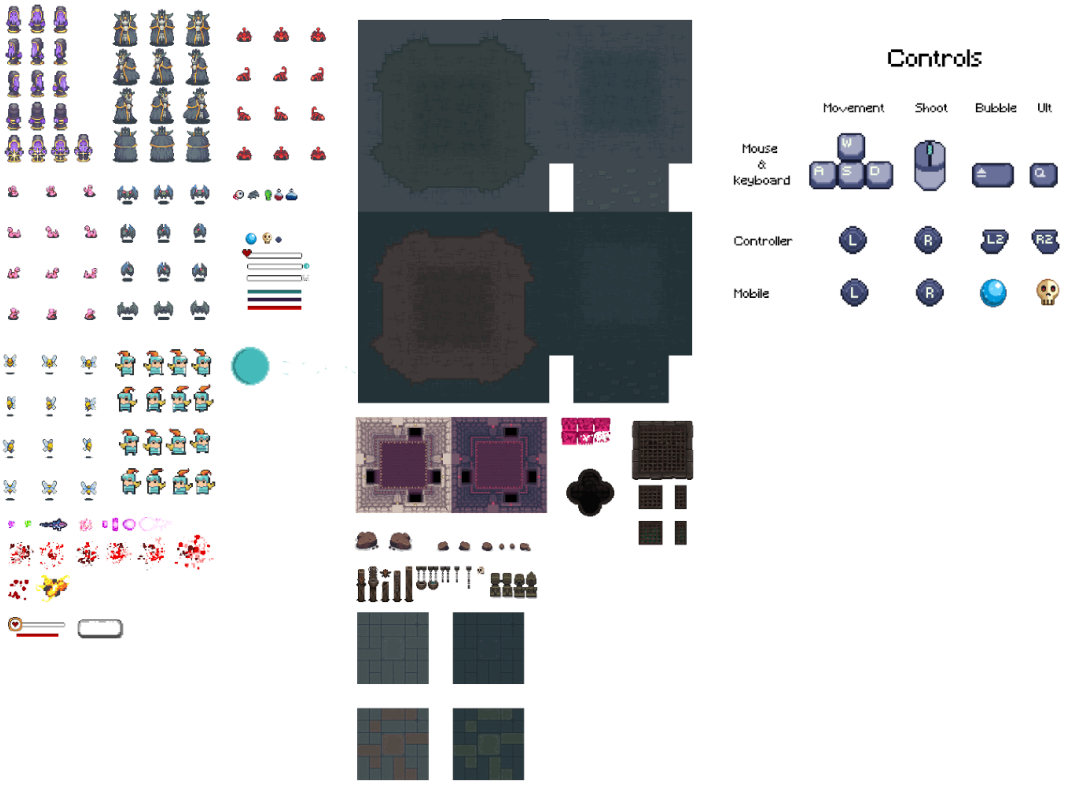
[19] Zotov, A. (2021). *Alexander Zotov channel* [Video]. Retrieved 4 June 2021, from https://www.youtube.com/channel/UCYgUFrFyfr5IyL8HfTTiWhA.

Llibres

Nystrom, R. (2014). *Game programming patterns*. Genever Benning.

Schell, J. *The art of game design*.

10. Annexos



Il·lustració 63 - Sprite-Sheet complet

10.1 Tests d'usuaris

El tests d'usuaris s'han realitzat per la plataforma mòbil, i els resultats són els següents:

Nom del participant	**Miquel Àngel Perelló**	**Sebastià Sansó**	**Maria del Mar Perelló**
Edat del participant	15 anys	26 anys	20 anys
Experiència prèvia amb els videojocs	Bastanta experiència amb jocs de consola, com *Call of Duty* i *Fortnite.*	Bastanta experiència en tot tipus de jocs.	No té gaire experiència amb videojocs. Només ha jugat al *Clash of clans* i al *Clash royale*.
Duració de la prova (*gameplay*)	3' 38''	3' 11''	2' 15''
Nivell	Últim nivell, joc superat.	Últim nivell, joc superat.	Segon nivell, joc no superat.
Comentaris	Bon joc, divertit i original.	M'agradaria destacar la intel·ligència enemiga, proporciona molt de dinamisme.	Els *items* i les habilitats són una bona aportació. Joc agradable i bonic.
Aspectes a millorar	El *joystick* podria tenir un rang més ampli.	Excés de screen shake.	Podria incloure més habilitats defensives.