Universitat de Girona
Escola Politècnica Superior

Treball final de Grau

Estudi: **Grau en Disseny i Desenvolupament de Videojocs**

Títol: Desenvolupament i publicació d'un joc per a mòbils amb components aleatoris

Alumne: David Cano Muñoz

Tutor: Antonio Rodríguez Benítez

Departament: Informàtica, matemàtica aplicada i estadística

Àrea: Llenguatges i sistemes informàtics

Convocatòria (mes/any):09/2022

INDEX

INDEX FIGURES I TAULES

Agraïments

En primer lloc, m'agradaria agrair al meu tutor, Antonio Rodríguez, per el seu assessorament i suport durant tot el projecte, i per atendre les meves inquietuds i mostrar un interès tan gran en el desenvolupament del projecte que sense ella no hauria estat possible.
També m'agradaria donar una part del meu agraïment als meus pares, al meu germans, a la meva avia i als meus amics que van aguantar el meu mal humor durant els moments estressants de la feina, la universitat i el TFG. Moltes gràcies per mostrar tant interès pel que estic fent i pel que faré, els vostres suggeriments i comentaris, molts dels quals han resultat ser molt útils.
A la Gabriela, per animar-me cada dia a treballar amb aquest projecte i per ajudar-me en aspectes que sincerament no son el meu punt fort.
Agrair també als amics fets a la Universitat que en pocs anys hem esdevingut un gran grup dins i fora de l'entorn universitari, gràcies per tots els dies i totes les nits de Discord treballant junts en els nostre projectes i pels plans de desconnexió que hem anat organitzant. Moltes gràcies sobretot a vosaltres Marc, Pau i Quim, sense vosaltres aquest projecte no hagués estat el mateix i potser no hauria arribat a aquest punt final.

1. Introducció i Objectius

Antecedents: En els darrers anys els mòbils han suposat un gran canvi en el nostre dia a dia, d'aquesta manera les companyies s'han aprofitat d'aquesta situació i han desenvolupat mòbils capaços d'executar programes i aplicacions que anys enrere era impensable.
Avui en dia les persones busquen relaxar-se jugant a jocs casuals, on el component de continuar millorant és el principal objectiu i motivació pel jugador, però sense perdre el factor de variabilitat per tal d'aconseguir una fidelització del públic cap al producte. Això ha fet que la indústria dels videojocs per a mòbils creixi a un ritme molt accelerat, superant, d'aquesta manera, als dispositius especialitzats en videojocs.

Objecte: L'objectiu del TFG és poder desenvolupar i publicar a la PlayStore un joc 2D, que tingui com a objectiu completar diferents nivells, que es creïn a partir de sales totalment aleatòries i que cada vegada que es jugui sigui una experiència diferent. Això el permetrà adaptar-se a les necessitats del mercat i ser un joc competent amb els de la mateixa classe.

Abast: Per desenvolupar aquest videojoc s'utilitzarà el programa Unity i alguns editors d'imatges com Photoshop i Krita o, d'altra banda, Gimp per poder fer el disseny de Sprites. Les activitats a realitzar i a estudiar durant aquest projecte seran:

- Fer un estudi de mercat dels videojocs de mòbil per poder-se adaptar i així tenir l'oportunitat de competir contra els jocs més populars de la plataforma.
- Dissenyar i implementar un joc 2D
- Aplicar una part aleatòria i procedural per permetre que cada partida sigui diferent
- Dissenyar i implementar la part artística del joc
- Publicar el joc en una Store oficial

1.1 Marc del projecte

Fa temps que ha deixat de ser una sorpresa que la indústria dels videojocs és la indústria de l'entreteniment que genera més diners, i que fins i tot en certs punts, supera per si mateixa al cine i la música junts. El creixement dels jocs mòbils, concretament, ha sigut molt notòri gràcies a la crescuda en paral·lel dels dispositius mòbils. Aquests estan present en el nostre dia a dia durant moltes hores i, juntament amb les xarxes socials, els jocs ocupen gran part del temps invertit en aquests dispositius, tal i com veiem en la figura 1.

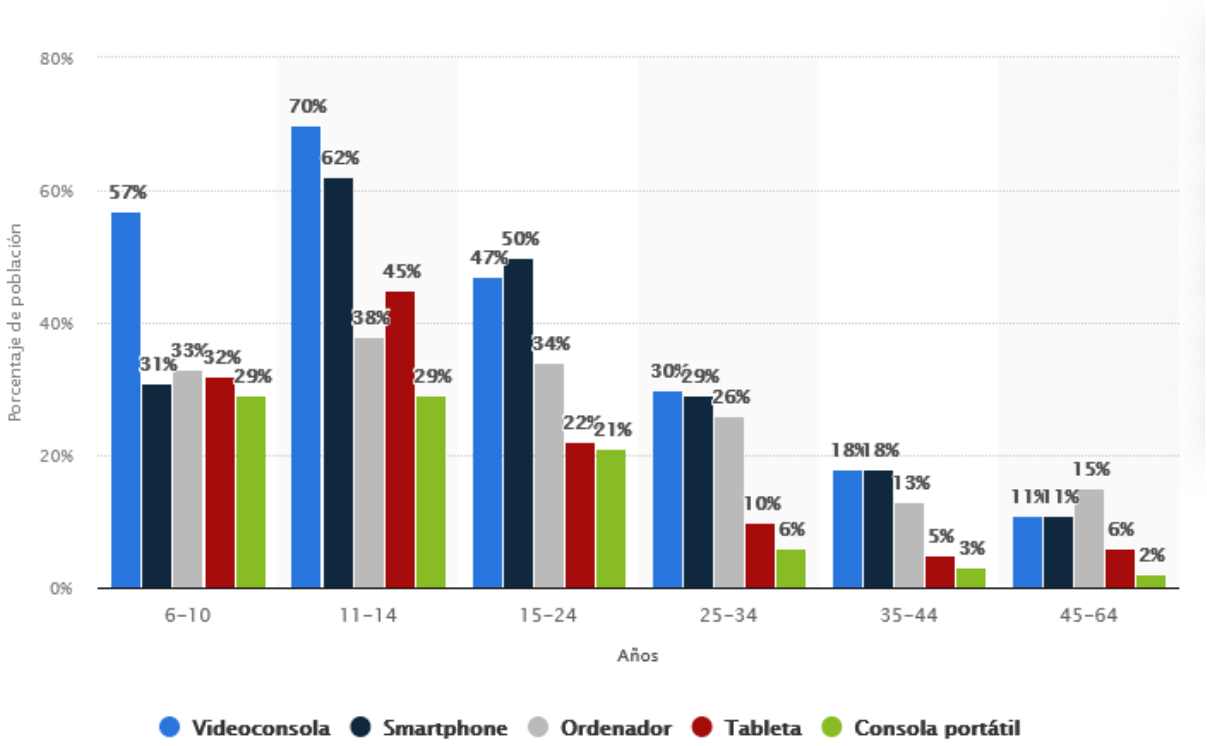


Figura 1: Comparació de plataformes on més es juga segons edats

Els videojocs han esdevingut un dels mitjans d'entreteniment més popular del món, i aquest fet no ha estat casualitat.
Si bé en un principi es relacionaven amb adolescents antisocials que es tancaven a casa tot el dia, han aconseguit canviar aquesta percepció. Primerament, l'interès d'arribar a molt més públic ha desembocat en multitud de gèneres diferents i ha permés l'aparició de jocs molt diversos com per exemple, amb dificultat graduable, molt llargs, frenètics i amb una gran historia (que fins i tot poden estar a l'altura de moltes pel·lícules), jocs on l'interesant es para-se a pensar cada acció, cada puzle o cada minijoc o jocs on les partides duren 3 minuts i són intenses i addictives.

La gran varietat i versatilitat, així com la immediatesa i accessibilitat, ha fet que els videojocs s'hagin expandit molt ràpidament i puguin ser gaudits per un ampli públic com ara bé, gent que vol gaudir d'una llarga història, gent que vol passar una estona amb els amics o gent què, senzillament, s'avorreix a l'autobús de camí al treball i necessita una píldora de distracció i desconnexió.

1.2 Motivacions

Fer un videojoc és per mi tant el cim de la meva etapa com a estudiant universitari, com el que serà la primera rajola en el meu futur com a dissenyador i desenvolupador de videojocs. No hi ha millor manera de posar en pràctica tot l'après durant el grau i demostrar-ho que en el projecte de final de carrera. Aquest serà alhora, la meva millor carta de presentació davant el món dels videojocs. Sempre he desitjat treballar per una empresa de videojocs mòbils, sembla un mercat que no deixa de créixer i que em pot obrir més portes per poder dedicar-me a allò que m'agrada. Per això, després d'una anàlisi de mercat, vaig decidir enfocar-me en intentar pujar a una plataforma mòbil un videojoc acabat.

Figura 2: Logotips icònics per descarregar un joc en les dues plataformes més importants

1.3 Elecció del videojoc

Tot i considerar diferents possibilitats de videojocs, des d'un principi és tenia bastant clar que volia que fos per dispositius mòbils.
Potser perquè des que vaig començar amb els primers mòbils, sempre n'he tingut algun instal·lat, o potser perquè en el meu entorn han estat present. Des de ben petit m'ha agradat jugar a qualsevol joc amb amics i amb el pas del temps, m'he adonat que el que enganxa són les partides que no duren més de 3 minuts, que són curtes però intenses, que cada vegada es posen més difícil i que si ets realment bo i tens el temps necessari, puguis arribar a fer partides més llargues. Per això vaig decidir fer un joc que es creés de forma procedural, que no es fes repetitiu i que fos el mateix jugador qui decideix la durada de la partida. No obstant això, d'acord amb la dificultat del joc, està dissenyat perquè la partida no acostumi a durar més de 5 minuts: temps d'espera mitjana que té la gent.

1.3.1 Per què per a mòbil?

Durant la carrera no he tingut l'oportunitat d'endinsar-me suficient en aquest camp com per saber tot el procediment per crear, desenvolupar, pujar i en cas de necessitat, actualitzar i millorar per mantenir en actiu un joc. Per això vaig decidir començar des de 0, per poder aprendre i testejar cada un dels passos suficients per poder fer arribar al públic de mòbil un videojoc complert.

Com ja s'ha menciona, en l'actualitat els dispositius mòbils han esdevingut la consola portàtil més utilitzada per la gent, ja que són l'aparell més popular i accessible del mercat. Gràcies a la seva portabilitat i senzillesa la gent pot jugar mentre va al metro, espera al bus, o simplement perquè no té res a fer i no és a casa. Això ha comportat que la indústria d'aplicacions i videojocs creixi i s'actualitzi molt en aquest camp fent que la varietat de jocs augmenti cada vegada més.
Aquest ha set el punt detonant per prendre la decisió a fer un joc per aquesta indústria que cada dia és més gran i més potent.

1.4 Propòsit i objectius del projecte

Així donc, l'objectiu principal és arribar a crear un videojoc accessible i per dispositius mòbils, seguint tots els passos necessaris i com si d'un desenvolupament professional es tractés.
La idea és dissenyar un joc que pugui arribar al màxim nombre de plataformes (com Android i/o IOS), dispositius i públic, que segueixi una estratègia que li permeti evolucionar al llarg del temps i s'introdueixi de manera eficient en aquesta indústria.

Addicionalment, s'ha proposat crear una petita versió del joc o prototip que en un futur es pugui reprendre per tal de dur a terme un videojoc complet amb la finalitat de posteriorment publicar-lo.

Per poder assolir aquest propòsit s'han marcat diferents objectius específics.

- Explorar el mercat i definir un públic objectiu
- Definir un guió pel joc
- Definir l'abast del joc
- Dissenyar i implementar el joc preparat per exportar a mòbil
- Exportar el joc a la forma adient per a poder pujar com a app per a mòbil
- Fer un manteniment i correcció de bugs que surtin en pujar-lo i provar-lo en diferents mòbils.

Amb aquests propòsits fixats per desenvolupar un videojoc es duran a terme tots els passos necessaris per saber que es necessita per distribuir un joc per mòbil.

1.5 Quadre d'autoavaluació

Al tractar-se d'un treball individual, el temps dedicat a els següents blocs es definit per les aptituds personals, a més dels requeriments del joc segons l'objectiu final .
És per això que s'ha dedicat i repartit d'aquesta manera el temps, l'energia i els coneixements en els següents apartats:

Narrativa	20 %
Estètica	30 %
Mecàniques	20 %
Tecnologia	30 %

Taula 1: esquematització de temps invertit

Els dos blocs més importants d'un videojoc basat en partides ràpides i de curta duració, és el bon comportament de les mecàniques ingame i una estètica atractiva pel jugador. Per això s'hi ha dedicat un 80% del temps i recursos, per tal que l'experiència del jugador tingui la màxima qualitat possible. S'ha prioritzat que el joc tinguin un funcionament correcte del jugador, del seu moviment i una interacció adient amb el mapa, una creació de nivells adaptada a la dificultat i lliure de bugs o comportaments incoherents.
A més, es va proposar fer el màxim possible de textures, animacions, personatges..., per tal de millorar la tècnica en l'àmbit personal i, com s'ha esmentat, que quedés atractiu i agradable al públic.
D'altra banda, la narrativa no és el punt fort dels jocs de mòbil que finalment triomfen; tot i això, aquests han de tenir una base i una història perquè el jugador quedi immers. És per això que s'ha dedicat un 20%.
Finalment, com no s'ha de crear cap tecnologia, sinó que aquest s'ha de poder adaptar als diferents dispositius mòbils, l'últim apartat té un 0%.

1.6 Organització del document

La Universitat de Girona ha facilitat una guia per poder crear el següent document.

1. Introducció i Objectius
2. Estudi de viabilitat
3. Planificació
4. Disseny del videojoc
5. Implementació i proves
6. Resultats
7. Conclusions
8. Treball futur
9. Bibliografia
10. Manual d'usuari i instal·lació

2. Estudi de viabilitat

Quan s'ha de posar en marxa un projecte amb aquesta ambició és important fer un estudi previ de viabilitat. En aquests casos, se sol fer una inversió significativa per invertir el temps i els recursos adients a més de poder fer una gran campanya de màrqueting. En canvi, en tractar-se d'un projecte personal no es disposen d'aquests recursos i s'ha dedicat el temps lliure que s'ha tingut durant el curs deixant de banda estudis i treball. És crucial recalcar que s'ha utilitzat el màxim de programari lliure i comercial amb la finalitat d'abaratir al màxim el cost.

2.1 Recursos necessaris

2.1.1 Eines

Les eines utilitzades per fer el TFG en la seva totalitat, classificades segons categories, són les següents:

- Tecnologia:
  - Ordinador amb els següents components:
    - **Procesador**: Intel Core i7-10700K 3.80 GHz
    - **Memoria Ram**: DDR4 3600 PC4-28800 16 GB
    - **Tarjeta Gràfica**: GeForce RTX 2060 AMP! 6GB GDDR6
    - **Pantalla**: AOC G2490VXA 23.8" LED FullHD 144Hz FreeSync
  - Teclat i ratolí
  - Tableta Grafica Wacom Intuos
  - Dispositiu Movil Samsung Galaxy A6+ per fer el testing
- Software:
  - **Windows 10**: Sistema Operatiu
  - **Unity**: Motor gràfic per poder desenvolupar el videojoc.
  - **Krita**: Programari de disseny de dibuixos, animacions, hud i tot el 2D.
  - **Visual Studio**: Programari que juntament amb el Unity, és necessari per poder programar mecaniques, estats, funcions del joc…
  - **Google Docs**: Per poder documentar el treball.
  - **Trello**: Servei web de planificació de tasques
- Economia i drets
  - **Campanya de màrqueting**: per arribar al màxim de públic.
  - Totes les imatges, objectes o scripts que han sigut estrets d'internet són d'ús públic i lliure i han sigut modificats i adaptats.

2.2 Recursos humans

El projecte està produït per complet per una sola persona amb l'objectiu de crear currículum i amb fins educatius i sense ànim de lucre. Tot i això, s'ha tractat com a projecte professional amb el propòsit d'algun dia poder monetitzar-lo. En cas de poder tirar-lo endavant com a projecte una mica més gran es podria haver dividit en 3 tipus de treballadors:

- Dissenyador i artista
- Desenvolupador i tester
- Guionista i director del Joc

2.3 Pressupostos inicials

2.3.1 Eines utilitzades

Els preus de cada objecte o programa són orientatius, ja que, gran part dels components que s'utilitzen ja estaven a disposició de l'usuari.

Producte	Cost
Ordinador complet	1400 €
Tableta Grafica Wacom Intuos	80 €
Samsung Galaxy A6+	360 €
Llicencia de Windows 10	20 €
Unity	0 €
Krita	0 €
Visual Studio	0€
Google Docs	0 €
Trello	0 €
Màrqueting	2000€
Total	3860 €

Taula 2: capital invertit en les eines del projecte

A part d'aquesta inversió, s'hauría de sumar les hores invertides pels recursos humans durant aquests mesos.

2.4 Estudi de mercat

Abans de començar un projecte d'aquestes magnituds s'ha de fer un estudi de mercat per valorar si definitivament és viable i si té cabuda dins del mercat.
Per començar s'hauria de definir el projecte i comparar-lo amb els jocs que més han triomfat dins d'aquest sector, amb jocs similars en jugabilitat, en diferents plataformes per poder decidir en quina o quines plataformes és adient publicar el videojoc i sobretot, per encapsular el públic objectiu del nostre projecte.
En aquesta comparació s'analitza:

- El contingut del joc amb relació a les hores que es pot jugar i el contingut dins d'aquest
- El preu del joc de sortida amb relació a l'apartat anterior
- Per últim la qualitat de l'art i com és adaptat amb la història.

Es fa una mitjana d'aquestes tres notes de cada part i aquesta nota es compara amb la del nostre joc.

2.4.1 Estat de l'art

Jocs similars que s'adeqüen a l'estil del nostre projecte.

- **The binding of Isaac**
  The binding of Isaac és un joc 2D llarg i intens amb una historia enrevesada on el jugador haurà de travessar les diferents sales de cada món, matant enemics, agafant tot tipus d'objectes necessaris per sobreviure amb l'objectiu d'arribar al últim Boss que el farà escapar d'una interminable historia.

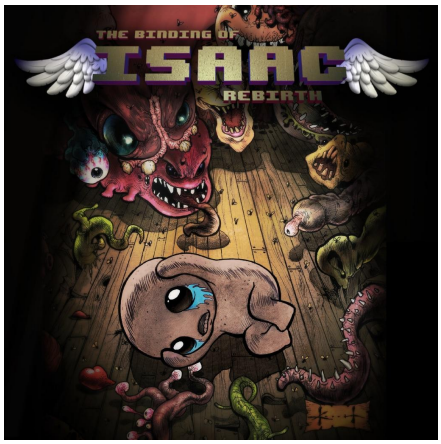


Figura 3: Portada del joc "The Binding of Isaac"

- **Caves**
  Caves of Qud és un videojoc d'aventura roguelike que ens trasllada a un món de ciència-ficció retrofuturista. La missió del jugador no serà una altra que la d'explorar centenars de coves i ruïnes d'una antiga civilització mentre combat contra enemics. Els elements i els nombrosos enemics estan dividits en diferents biomes.

Figura 4:Imatge representativa del joc "Caves"

- **Pathos**
  Pathos és un joc d'aventura roguelike inspirat en el conjunt de regles de Nethack. El jugador podra triar entre 13 classes i viatjar fins a les profunditats de la masmorra. Descendeix a l'infern per a vèncer a la teva nèmesi abans d'escapar de la masmorra amb tot el botí que et puguis emportar!

Figura 5: Imatge representativa del joc "Pathos"

2.4.2 Paràmetres d'avaluació

En la següent taula comparem els jocs anteriors amb el nostre i s'avaluen del 0 al 10 aspectes com el contingut, el preu respecte a les hores de joc, l'art i la integració de la història amb aquest. A continuació, en la cinquena columna es calcula una mitjana que servirà per determinar el nivell del projecte respecte als anteriors. Finalment, com es pot veure, el nostre joc supera la resta, indicador de què pot triomfar sobre els altres, en les plataformes digitals.

#num Joc	contingut	preu	art	mitja
Dungeon Mobile	8	10	7	8.33
The binding of isaac	7	7	6	6.66
Caves	6	8	5	6.33
Pathos	5	10	5	6.66
Enter the Gungeon	7	4	5	5.33

Taula 3: Comparació amb els jocs que s'assimilen del mercat

2.4.3 Perfil de Jugador:

Podem definir “target” com el públic objectiu, el grup de persones al qual han d’anar dirigits i enfocats tots els esforços en termes de màrqueting, per tal d’acostar al públic a la nostra marca.

Definir correctament el *target* és una de les tasques més importants dins de la nostra estratègia, ja que serà molt important de cara a optimitzar els nostres recursos. Això comportarà nombroses avantatges com, per exemple:

- Saber cap a on enfocar l’estratègia de màrqueting
- Gràcies a la segmentació, cada vegada es podrà conèixer millor el públic objectiu i ajustar millor les campanyes.

L'edat, un aspecte clau en la definició del target. Si bé no hi ha una escala o regla exacta que indiqui l'edat, es pot generalitzar i organitzar per determinats rangs com:

➔ Nens (de 0-14 anys)
➔ Adolescents (15-19 anys)
➔ Joves (20-29 anys)
➔ Adults (30-40 anys)
➔ Adults B (41-59 anys)
➔ Ancians (60 o més)

Com que el joc requereix una mica d'habilitat, com per exemple a l’hora de moure’s i disparar (ja que s’han d’utilitzar diversos dits alhora de manera ràpida i en diverses direccions), es podria dir que l'objectiu o target principal són els adolescents i els joves.
Els perfils de jugador esmentats ja estan acostumats a aquest tipus de joc. Són un públic amb, majoritàriament, més temps lliure, que acostuma a jugar jocs de partides curtes i què, a més, solen compartir aquesta afició amb el seu entorn. És aquest fet el que “potenciarà” l’estratègia de màrqueting.
Un altre factor interessant i a tenir en compte és que els joves, estan cada cop més acostumats a efectuar compres dintre dels mateixos i a fer micropagaments. Això obre les portes a una actualització setmanal o mensual amb nous personatges o diferents skins (textures dels personatges), sempre i quan el joc triomfi.

Els videojocs poden oferir diferents tipus de diversió segons els tipus de jugadors. En el llibre “the art of game design” podem veure que el nostre joc ha de complir les condicions següents:

3. Planificació

3.1 Tasques

3.1.1 Art

- **Personatge Principal**
    - Disseny del personatge: pensar una idea de personatge adaptat a una història i un temps específic.
    - Dibuixar: dibuixar el personatge enfocat a poder animar-lo posteriorment amb facilitat.
    - Animar: fer les animacions de moure's, atacar, morir i esperar del personatge principal.

- **Enemic**
    - Disseny dels Enemics i Boos: crear enemics que tinguin concordança en espai i temps amb la història i els altres personatges.
    - Dibuixar: dibuixar els enemics enfocats a poder animar-lo posteriorment amb facilitat.
    - Animar: fer les animacions de moure's, atacar, morir i esperar.

- **Dibuixar textures mapa primer nivell:** Dibuixar textures amb temàtica de "dungeon" per parets, terra i obstacles.
- **Dibuixar Textura mapa segon nivell:** Dibuixar textures amb temàtica de desert per parets, terra i obstacles.
- **Dibuixar Textura mapa tercer nivell:** Dibuixar textures amb temàtica de castell per parets, terra i obstacles.
- **Dibuixar Textura mapa quart nivell:** Dibuixar textures amb temàtica de muntanya per parets, terra i obstacles.
- **Dibuixar Textura mapa cinquè nivell:** Dibuixar textures amb temàtica de casa per parets, terra i obstacles.

3.1.2 Mecaniques/Programació

- **Personatge Principal**
  - Programar el moviment: fer que el personatge es mogui en els eixos X i Y.
  - Fer i rebre mal dels enemics: crear un sistema de vida que es pugui veure quan reps mal i fas mal a enemics.
  - Disparar: programar que el jugador pugui apuntar i disparar cada x temps.
  - Interactuar amb les portes: poder entrar a les portes i que et portin a la sala adient.
  - Agafar objectes: en passar per sobre d'objectes com claus, poder agafar-los.

- **Enemics**
  - Patrullar per la sala: crear circuits aleatoris pels quals els enemics es moguin per la sala.
  - Buscar al personatge principal: buscar al jugador perquè en el moment en el qual es tingui contacte visual ataquin.
  - Seguir al personatge principal: Seguir al personatge per poder atacar-lo.
  - Atacar: si el jugador està el prou a prop atacar i poder impactar o fallar depenent del moviment del personatge principal.

- **Creació sales procedural**
  - Instanciar enemics a les sales: crear enemics a les sales i que no es creïn sobre objectes o posicions inadequades.
  - Instanciar un Boos per Nivell: Instanciar un sol Boos per nivell i a sales finals.
  - Instanciar una clau per poder passar de nivell: instanciar una clau per nivell que permeti obrir la porta final.
  - Instanciar de forma aleatòria objectes i obstruccions per les diferents sales.
  - Instanciar el personatge: instanciar el jugador en una sala central cada vegada que es passa de nivell.
  - Instanciar Power-ups: instanciar amb un percentatge de probabilitat diferent power-ups que puguin ajudar al jugador.
  - Definir i implementar un sistema de puntuació que permeti encoratjar al jugador a superar-se cada vegada que juguí.

- **MiniMapa**
    - Crear instàncies de les sales a la part superior simulant un minimapa
    - Interacció del minimapa amb el personatge i canviar de color segons la sala en què estigui.

- **Puntuació:**
    - Punts per mort, aquests augmenten cada vegada que se superi un nivell.

- **Menús**
    - Crear un menú inicial amb un botó de configuració un botó “Start” i un botó “exit”.
    - Crear un menú de pausa i configuració on es pugui modificar el so.
    - HUD mentre es juga on es pugui veure la vida del jugador, la posició dels joysticks, i un minimapa per saber a quina sala estàs.

- **Música i sons**
    - Implementar diferent música pel menú i pel joc.
    - Implementar sons de trets i impactes

3.2 taula de planificació

Taula necessària per fer el gràfic de grannt inicial on s'indica cada una de les tasque la data d'inici prevista, la data final prevista i els dies que es creuen necessaris, en aquest cas tots tenen dues setmanes per tenir una mica de marge i poder combinar el temps empleat amb la feina i els estudis.

	Nom de l'activitat	Data Inici	Duració en dies	Data Final
Art Personatge Principal (PP)		15/07/2021	15	17-jul
	Disseny del Personatge Principal	17/07/2021	15	01/08/2021
	Dibuixar PP	22/07/2021	15	06/08/2021
	Animar PP	27/07/2021	15	11/08/2021
Art Enemic				
	Disseny dels Enemics i Boos	01/07/2021	15	16/07/2021
	Dibuixar Enemics	03/07/2021	15	18/07/2021
	Animar Enemics	07/07/2021	15	22/07/2021
	Dibuixar textures mapa primer nivell	12/02/2022	15	27/02/2022
	Dibuixar Textura mapa segon nivell	22/02/2022	15	09/03/2022
	Dibuixar Textura mapa tercer nivell	01/02/2022	15	16/02/2022
	Dibuixar Textura mapa quart nivell	13/02/2022	15	28/02/2022
	Dibuixar Textura mapa cinquè nivell	22/02/2022	15	09/03/2022
Personatge Principal				
	Programar el moviment del PP	01/09/2021	15	16/09/2021
	Fer i rebre mal dels enemics	01/10/2021	15	16/10/2021
	Disparar	15/09/2021	15	30/09/2021
	Interactuar amb les portes	12/10/2021	15	27/10/2021
	Agafar objectes			
Enemics				
	Patrullar per la sala	01/06/2022	15	16/06/2022
	Buscar al personatge principal	03/04/2022	15	18/04/2022
	Seguir al personatge principal:	15/04/2022	15	30/04/2022
	Atacar	10/10/2021	15	25/10/2021
Creació sales procedural				
	Instanciar enemics a les sales	15/06/2022	15	30/06/2022
	Instanciar un Boos per Nivell	01/07/2022	15	16/07/2022
	Instanciar una clau per poder passar de nivell	01/08/2022	15	16/08/2022
	Instanciar de forma aleatòria	22/06/2022	15	07/07/2022
	Instanciar el personatge	01/10/2021	15	16/10/2021
	Instanciar Power-ups	15/11/2021	15	30/11/2021
	Definir i implementar un sistema de puntuaci	01/12/2021	15	16/12/2021
MiniMapa				
	Crear instàncies de les sales	01/03/2022	15	16/03/2022
	Interacció del minimapa amb el personatge	15/12/2021	15	30/12/2021
Puntuació				
	Punts per mort			
Menús				
	Crear un menú inicial	15/01/2022	15	30/01/2022
	Crear un menú de pausa	20/05/2022	15	04/06/2022
	HUD	01/11/2021	15	16/11/2021
Música i sons			15	
	Implementar música	15/07/2021	15	30/07/2021
	Implementar sons de trets i impactes			
Memoria				
	Crear la memoria	01/01/2022	246	04/09/2022

Figura 6: Taula de planificació del temps

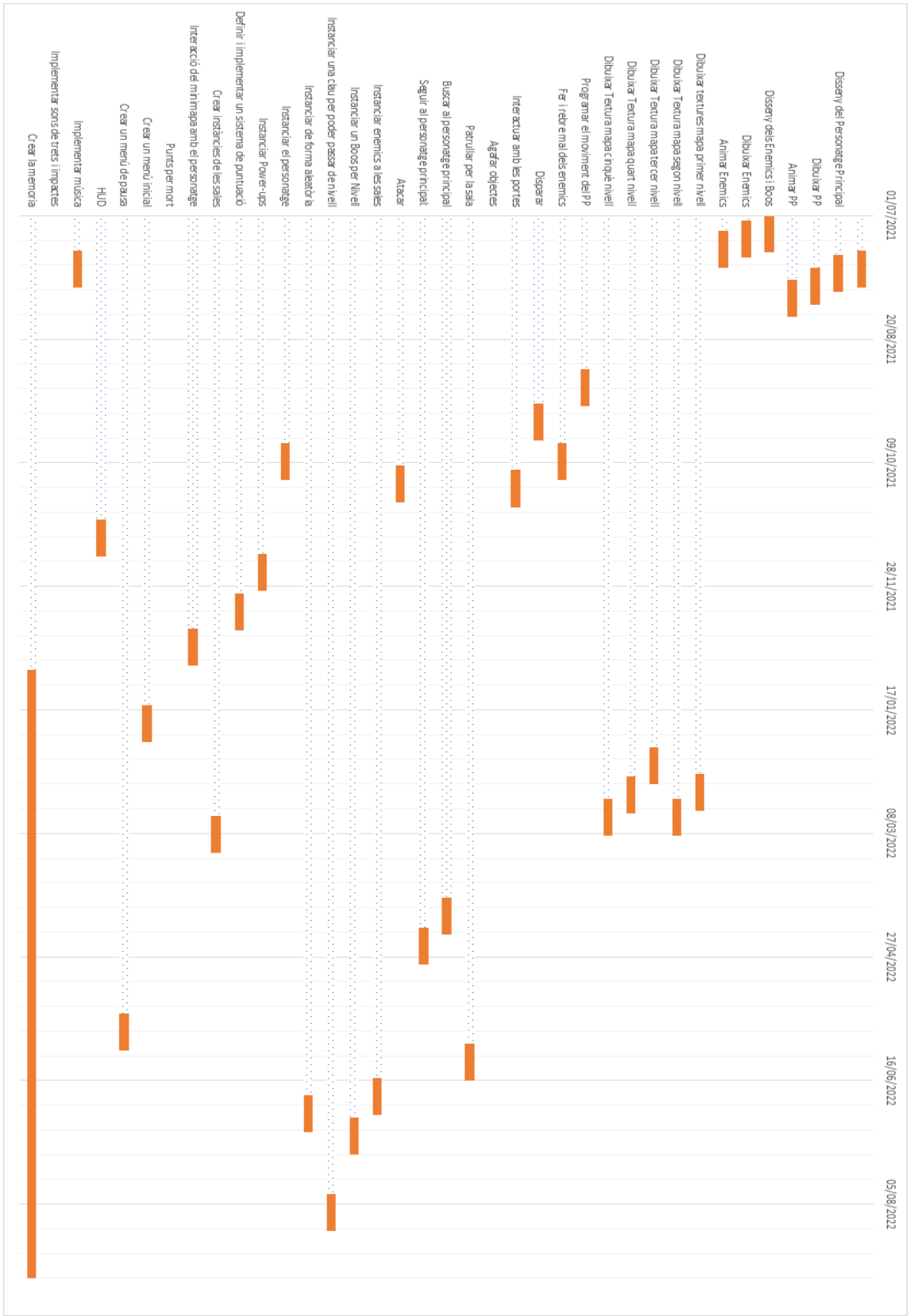


Figura 7: Gràfic de Gannt de la planificació del temps

3.3 Objectius

L'objectiu general i sense entrar en detall del joc, és superar-se a un mateix millorant els records personals i aconseguint el màxim nombre de punts possibles. Aquest fet generarà certa competitivitat entre els jugadors i els mantindrà actius durant més temps.
D'altra banda, l'objectiu principal del joc és recórrer el màxim nombre de mons possibles. Per poder superar-los el jugador haurà de trobar el Boss del nivell pertinent i en matar-lo, s'obrirà una porta, què, en cas que s'hagi trobat la clau adient, conduirà al següent món. El disseny del joc permet que les sales es creen aleatòriament i, per tant, la quantitat de mons és infinita.

4. Disseny del videojoc

4.1 Història dels videojocs mòbils:

Avui dia els videojocs generen molts més diners que el cinema, cada llançament és un esdeveniment mundial i les descàrregues i vendes es fan en qüestió d'hores. Però no sempre va anar així.
El primer a generar molts diners i vendes i que donà origen a la indústria tal com la coneixem, és el joc Odyssey, creat en 1966 per Raph Baer. Es tracta d'un joc tan senzill i divertit que fins i tot es continua jugant avui dia.

El primer joc per a mòbil va ser llançat per Nokia, actualment propietat de Microsoft. Era el joc del *snake*, una serp que havia de menjar fruita per poder créixer sense topar-se amb ella mateixa. Venia dintre del sistema del Nokia 6110.
El joc era senzill. Mentre la gent se sorprenia amb els jocs de la Nintendo 64 amb els seus gràfics millorats, tot dins d'un cartutx, Nokia amb aquest joc canviava totalment l'ús dels mòbils per sempre.

Poc a poc les empreses van afegir jocs als seus mòbils, Nokia va llançar Snake II, i els jocs arcade dels 80, estaven ara en els nostres petits telèfons mòbils.

Però també les empreses de videojocs de tota la vida es van unir al mercat mòbil; Sega va portar Super Monkey Ball de la seva consola als telèfons mòbils, i des de llavors han sorgit empreses que han aconseguit l'èxit amb els seus jocs per als mòbils. I ull, que no parlem de mòbils Full HD o 4K com ara, sinó de píxels de colors.

No és que els jocs de mòbils, siguin avui millor del que eren. És que els jocs són ara més fàcils de trobar, descarregar i jugar, que els d'abans.

4.2 Màrqueting

Després de l'estudi de mercat (2.3) es comença amb el plantejament del pla de màrqueting. En aquest cas, s'ha dut a terme una recerca que, juntament amb la idea inicial de fer un joc mòbil, ha portat a crear un joc 2D d'història i partides curtes.

En l'actualitat hi ha molt pocs jocs 3D que triomfin en les plataformes mòbils, ja que, els dispositius de gamma mitjana i baixa necessiten molta més potència (de la que disposen en l'actualitat) per poder executar amb fluïdesa aquest tipus de jocs. Això ens ha permès poder ampliar el públic objectiu del videojoc.

Així doncs, amb el joc ja en desenvolupament i l'estudi de mercat clar, es decideix començar amb el pla de màrqueting basat en les xarxes socials, que faciliten una via directa cap a les plataformes de distribució (4.1.1). Aquest consisteix en 3 grans punts:

- En primer lloc, destinar una part del capital per crear anuncis atractius en xarxes socials com Instagram, Twitter i sobretot Tiktok, una de les més utilitzades, ara per ara, pel públic objectiu del joc.
- En segon lloc, i com a complement del primer punt s'incitaria a creadors de contingut de plataformes com Twitch o YouTube, on el seu contingut sigui els jocs i sobretot els jocs mòbils, perquè provin el joc i el comparteixin amb els seus seguidors. També dedicant una part del capital destinat al màrqueting.
- En últim lloc, a través de comptes oficials del joc, s'intentarà arribar al màxim públic possible fent servir com a referència tècniques de màrqueting de comptes (en les xarxes) d'empreses com KFC, que, aprofiten al màxim l'edat del seu públic per crear publicacions cridaneres que permeten una interacció directa empresa-client.

Plataformes de distribució i consum

Les dos principals plataformes de distribució idònies són Google Play d'Android i App Store per a usuaris de IOS. Tant mateix, degut al baix pressupost i l'elevada demanda (per "publicació" en App Store), en un principi només es llençarà per Google Play.

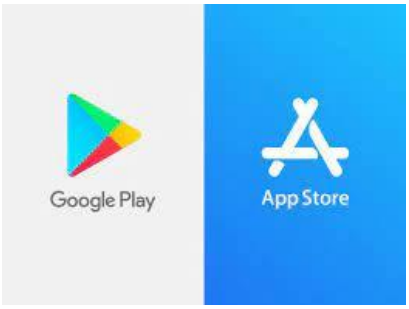


Figura 8: logos de Google Play i App Store

4.3 Espai de joc i mecàniques:

4.3.1 Espai de joc.

L'espai de joc es basa en un conjunt de sales que es creen de forma aleatòria cada vegada que es canvia de nivell. Es genera una sala central (la inicial), que porta a diverses sales en totes direccions. En una sala allunyada, en una de les cantonades, es crea un Boos que en ser eliminat deixa veure una porta cap al següent nivell. Aquesta seria la sala final de cada nivell. Tant mateix, en totes les altres sales apareixen enemics i power-Ups.

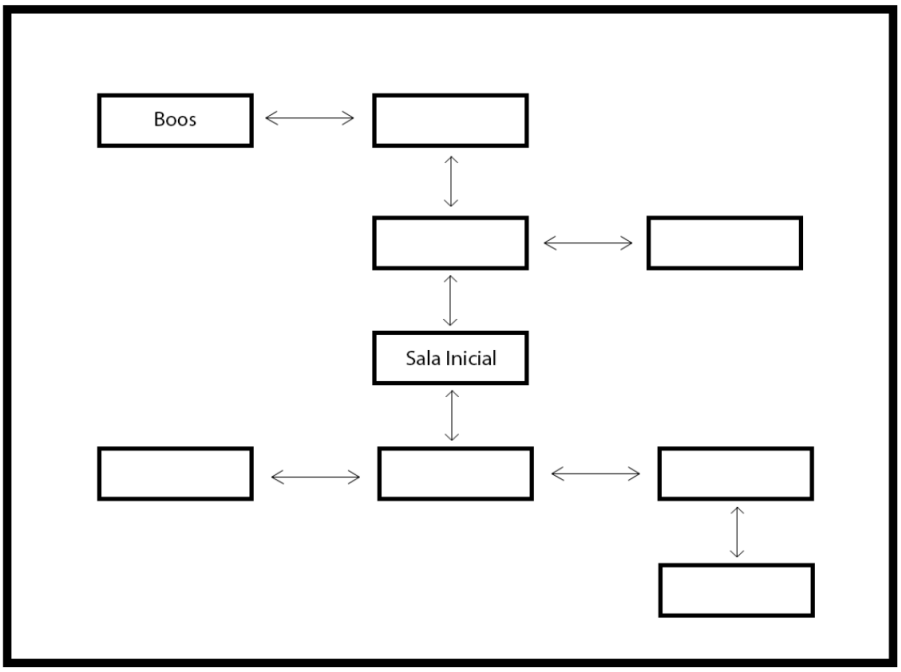


Figura 9: Figura representativa de com seria un nivell

4.3.2 Mecàniques:

Podem identificar les mecàniques com aquelles accions que només el jugador és capaç de fer i amb les quals es basa el joc complementat-se entre elles.

El joc es basa en dos mecàniques basiques

- La mecànica principal, moure's amb el joystick situat a la part esquerra de la pantalla sobre els eixos X i Y
- Seguidament la segona mecànica bàsica és la de disparar i apuntar
- Com a mecàniques auxiliars podem afegir el poder passar per portes, canviar el tipus de dispar o regenerar vida

4.3.3 Reptes:

Es poden desglosar el reptes per nivells, els quals per has d'assolir desde el mes bais al mes alt per poder assolir-los completament.

- El repte Inicial és sobreviure el màxim temps possible i passar el màxim nombre de nivells
- Per poder passar de nivell s'ha de trobar el Boos i matar-lo.
- Per poder trobar el Boos s'ha de moure a través de les sales
- Per poder moure't per les sales s'ha de matar als enemics de cada sala
- Per poder matar s'ha d'apuntar i disparar amb els dos joysticks.

4.3.4 Accions:

Accions són les interaccions més senzilles que pot fer el personatge dins del joc. Dins d'aquestes s'inclouen les següents

- Moure's sobre el pla 2D en totes direccions en els eixos X i Y
- Disparar en totes direccions
- Passar per portes
- Agafar Power-ups, vida i claus.

4.3.5 Objectes interactius

4.3.5.1 Power-ups:

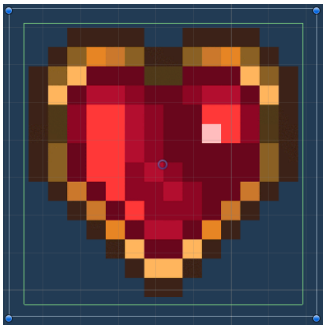
Els power-ups són controlats per l'script del jugador (*controllerPlayer.cs*), ja que s'activen en entrar en contacte amb el jugador, i només afecten aquest, és a dir un enemic no podrà agafar el power-up. Això funciona gràcies a la funció *OnCollisionEnter2D (figura 10)* que detecta si el jugador ha entrar en contacte o no amb el power-up i en cada cas, activa o modifica els valors necessaris del jugador.

```
private void OnCollisionEnter2D(Collision2D collision)
{

    //Debug.Log(collision.gameObject.tag);
    if (collision.gameObject.tag == "vida")
    {
        health += 20;
        vidaUI.setHealth(health);
        Destroy(collision.gameObject);
    }
    else if (collision.gameObject.tag == "speed_hab")
    {
        speed_pl = 3;
        Destroy(collision.gameObject);
        Invoke("changeSpeed1", 10f);
    }else if (collision.gameObject.tag == "multiple_hab")
    {
        multipleShoot = true;
        Destroy(collision.gameObject);
        Invoke("changeMultipleShoot", 15f);
    }
}
```

Figura 10: tros de codi on es detectan els power-ups que agafa el jugador

- Vida:



```
if (collision.gameObject.tag == "vida")
{
    health += 20;
    vidaUI.setHealth(health);
    Destroy(collision.gameObject);
}
```

Figura 11: Sprite del power-up de la vida

Figura 12:funció que modifica els valors del jugador al agafar el power-up de vida

Tal i com podem veure a la figura 12 el power-up de la vida aumenta un 20% del total i actualitza la barra de vida del HUD.

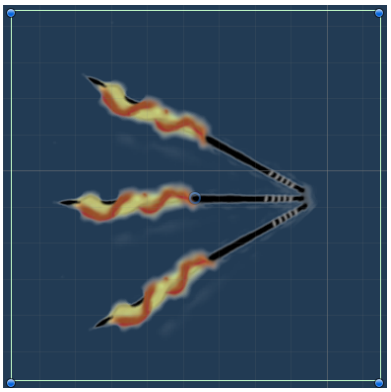
- Velocitat:

```
else if (collision.gameObject.tag == "speed_hab")
{
    speed_pl = 3;
    Destroy(collision.gameObject);
    Invoke("changeSpeed1", 10f);
}
```

Figura 13: Sprite del power-up de velocitat
Figura 14: funció que modifica els valors del jugador al agafar el power-up de velocitat

El Power-up de velocitat augmenta la velocitat del jugador al triple de la velocitat original, aquest al passar un temps determinat.

- Triple dispar:

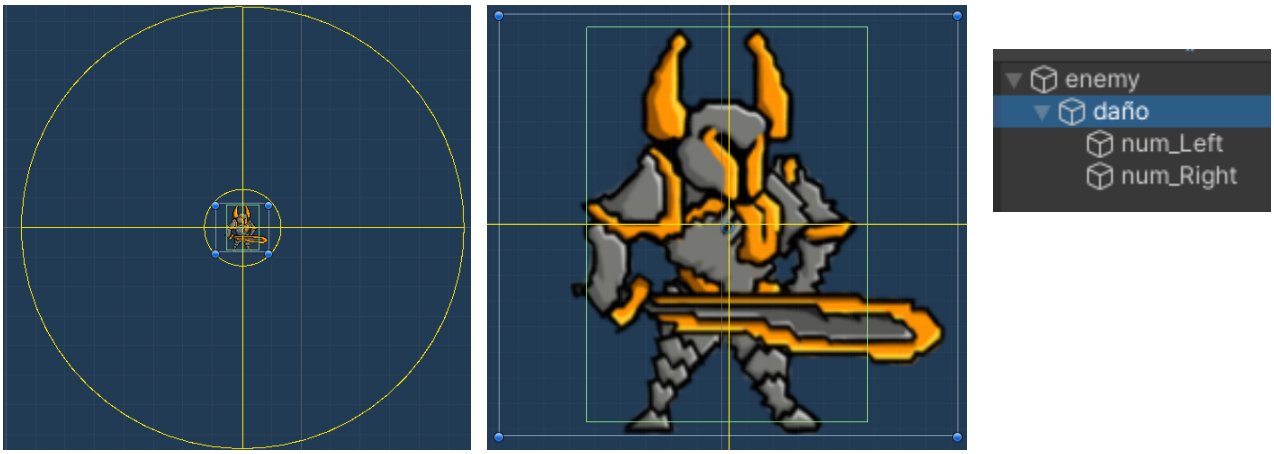


```
}else if (collision.gameObject.tag == "multiple_hab")
{
    multipleShoot = true;
    Destroy(collision.gameObject);
    Invoke("changeMultipleShoot", 15f);
}
```

Figura 15: Sprite del power-up que modifica el tipus de dispars
Figura 16: funció que modifica els valors del jugador al agafar el power-up de triple dispar

El power-up del “triple dispar” canvia el tipus de dispar de simple a triple. Aquest no modifica el dany de forma individual però dóna la possibilitat d’impactar al mateix moment a més d’un enemic o al mateix enemic amb més d’una llança.

4.3.6 Enemics



Figures 17: Imatges del Prefab del enemic amb els seus components

L'enemic base i principal que apareix amb normalitat consta de l'script *enemy.cs*, un sprite Renderer que mostra una Sprite, un Animator per mostrar les animacions necessàries, un BoxCollider i un RigidBody. A més consta de dos cercles que detecten si el jugador entra al rang definit i en aquest cas poder atacar-lo.

- El Boos final de cada nivell consta d'un Script específic, la base d'aquest Script és la mateixa que la d'un Enemic normal, però té alguns valors modificats, com la vida, el dany que pot fer... A més també consta d'alguna funció específica com la següent, que en morir el Boos, s'instància el prefab de la porta i es destrueix l'objecte "Boos".

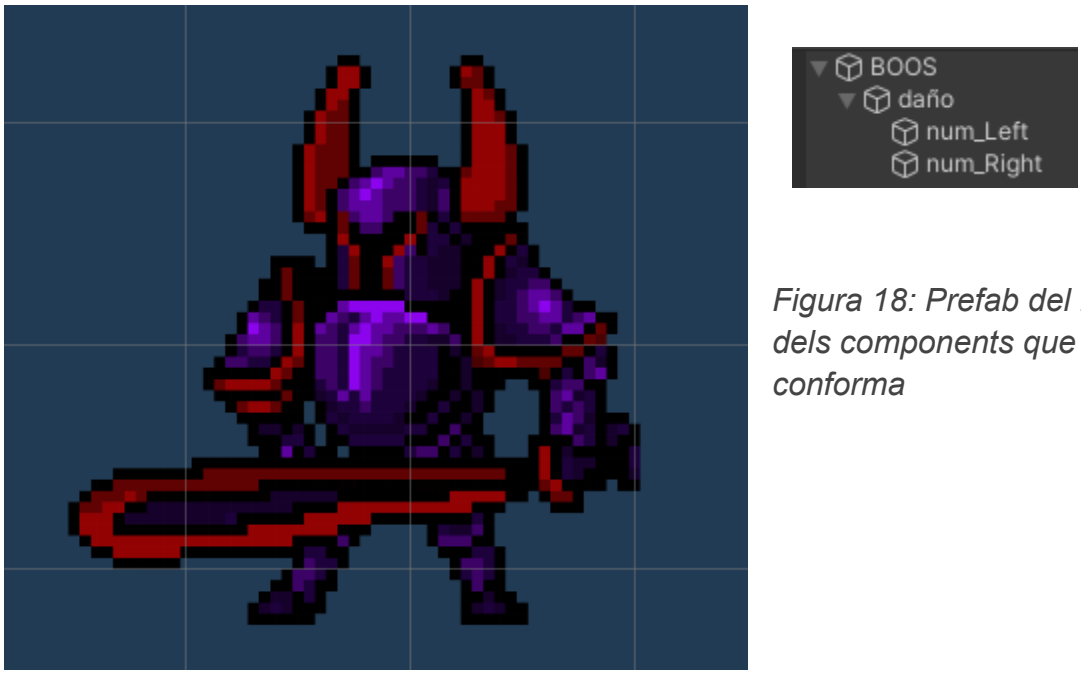


Figura 18: Prefab del Boos i dels components que els conforma

4.3.7 Objectes mapa

- Porta

El prefab DoorTriggerlvl consta del Script "DoorTrigerlvl.cs"(figura 19), un script senzill que detecta si aquest té una col·lisió amb el jugador i si aquest porta la clau d'aquest nivell. La porta estarà oberta per carregar un nou nivell.

```
private void OnTriggerEnter2D(Collider2D collision)
{
    bool keyCatched = GameObject.Find("player").GetComponent<controllerPlayer>().keyAgafada;
    if (collision.tag == "Player" && keyCatched)
    {
        Debug.Log("OnTriggerEnter2D llama a augmenta nivell");
        this.GetComponent<SpriteRenderer>().sprite = openDoor;
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        //SceneManager.LoadScene("scene2");
        _SheetAssigner.augmentanivell();
        lvlUI.text = _SheetAssigner.retornLVL().ToString();

    }
}
```

Figura 19: Codi que detecta quan el jugador entra per una porta

Figura 20: Sprite del prefab de la porta que et permet passar de nivell

- Clau

El prefab “Key” no consta d’un script propi, és a dir, a l’hora de crear-se el mapa es crea de forma aleatoria en aquest pero, és l’script del jugador el que detecta si la clau s’agafa.

Figura 21: Sprite del objecte “Key”

4.3.8 Recursos

- Vida

La vida és un element clau durant tot el joc, el jugador té 100 punts en el moment inicial i només en podrà recuperar 20 per cada cor que trobi al mapa. Tant el Jugador com els enemics fan mal aleatori dins d'un rang determinat, però el normal és que el jugador aguanti entre 3 i 5 cops de melé dels enemics base i l'enemic entre 3 i 4 cops de llança. En canvi, el Boos pot aguantar fins a 10 cops.

4.3.9 Interaccions

- Portes: Algunes portes del joc et permeten moure't a través de les sales del nivell i altres et permeten passar a les sales del següent nivell. Per interaccionar amb aquestes només és necessari col·lisionar amb elles.
- Enemics: Els enemic et perseguiran sempre i quan tinguin contacte visual amb el jugador, aquest t’atacaran si estan lo suficientment prop, això restara al jugador un valor aleatori de vida. El jugador podra disparar i matar-los apuntat amb el joystick dret.
- Power-Ups:
    - Vida: En entrar en contacte augmenta un 20% la vida del jugador
    - velocitat: En entrar en contacte la velocitat de moviment augmenta durant uns segons
    - Triple Dispar: Durant un període de temps el jugador dispara 3 fletxes a la vegada, impactant més d'una fletxa es podrà fer més mal.

4.4 Economia

L'economia del joc es basa en una puntuació que s'obté a partir dels punts totals d'una partida. L'objectiu és intentar superar cada vegada que es juga, el récord personal de punts o també anomenat highscore, a més d'intentar arribar a un nivell més alt. Aquests punts s'obtenen matant enemics i passany nivells. Cada vegada que s'augmenta de nivell es rebran més punts per enemic, ja que és més difícil matar-los.

4.5 Creació de nivells

A mesura que es va matant el Boss de cada nivell s'obre una porta que porta al següent nivell, on canvien les textures simulant cambiar de biomes, edificis… A més, com les sales es creen de forma aleatoria i procedural sempre seran nivells diferents.

4.5.1 Ambients:

- NIVELL 1

Figura 22: imatge in-game del primer nivell

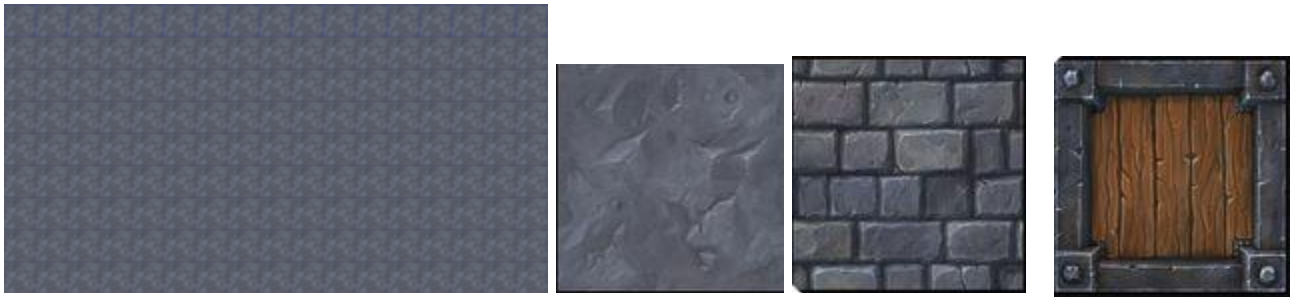


Figura 23: textures del primer nivell

- NIVELL 2:

Figura 24: imatge in-game del nivell 2

Figura 25: textures del segon nivell

- Nivell 3:

Figura 26: imatge representativa in-game del nivell 3

Figura 27: textures del tercer nivell

- Nivell 4:

Figura 28: imatge del nivell 4

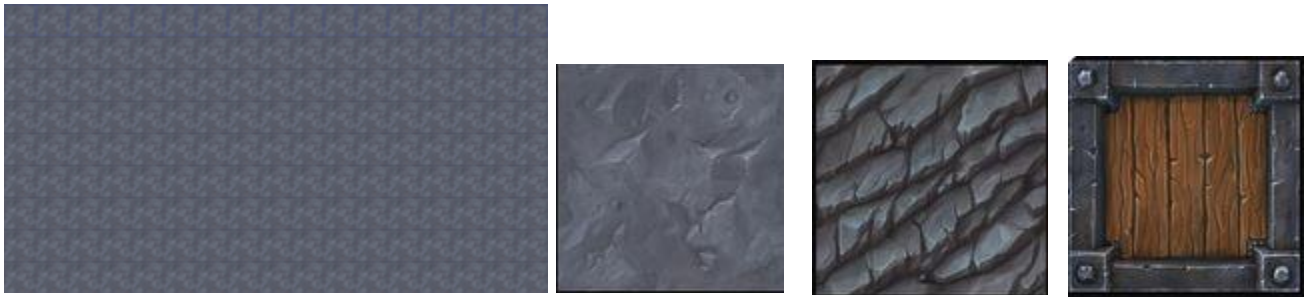


Figura 29: textures del nivell 4

- Nivell 5:

Figura 30: imatge dels nivells 5 en endavant

Figura 31: textures del nivell 5

Disseny d'interfícies i Gamelayouts,:

- Mapa del nivell amb les sales corresponents a la part superior esquerra
- Barra de vida a la part superior dreta
- joystick de moviment a la mitad esquerra
- joystick de apuntar i disparar a la mitad dreta

Narrativa:
El joc es basa en l'edat mitjana, en un món de fantasia, on el personatge principal en un primer instant es troba en un castell encantat on va viatjant per diferents sales. L'objectiu del personatge és sortir amb vida d'aquest castell infernal. Per fer-ho decideix moure's per les diferents sales del castell, matant a tot el que se li posa per davant, ja que si no el mataran a ell. Així descobreix que hi ha un enemic que amaga una porta, que només es pot travessar matant-lo. Quan travessa aquesta porta s'adona que pot viatjar a diferents "biomes" del castell, el que li dona esperança per sortir, però a poc a poc es va adonant que potser sortir d'aquest maleït castell serà impossible, ja que les sales són infinites.

4.6 Estudi i disseny de personatges.

4.6.1 Personatge Principal

Figura 32: sprite del personatge principal i les seves animacions

El personatge principal és un lluitador de l'edat mitjana, que es troba tancat en una dimensió 2D, aquest amb l'ansietat de voler escapar, comença a matar a tot el que es posa davant i comença a viatjar per diferent portes amb l'ambició de trobar una sortida.
Aquest personatge és d'estatura mitjana, bastant hàbil i amant de les llances. Sempre ha tingut una mentalitat molt lluitadora i sempre té al cap defensar la seva pàtria, per això els colors de la seva llança juntament amb la capa.
L'estil de dibuix és cartoon sense ser píxelart però donant-li tocs realistes com per exemple les ombres. Les proporcions no són realistes perquè li he volgut donar un toc característic cartoon, per exemple té un cap i un cos gran per poder remarcar la força que té el personatge i, en canvi, té uns braços i no té cames per donar-li més agilitat. També es pot remarcar el fet que no té cames, només té peus. Gràcies a això es pot remarcar el pit i el tronc del personatge

4.6.2 Enemics

Figura 33: sprite de l'enemic i les seves animacions

L'enemic simula un minotaure, el casc té la forma i les banyes d'un bou. Aquest només té atac cos a cos, però que l'espasa sigui tan gran i llarga, permet al minotaure arribar més lluny. Aquest fet contraresta el fet que sigui un personatge pesat amb una mobilitat bastant reduïda a l'hora d'atacar. L'estil de dibuix és igual al del Personatge Principal, un estil cartoon que no és píxel art, però amb tocs semi-realistes, com les ombres i els canvis de colors. En diferencia del personatge principal té cames i braços més grans per remarcar que és un personatge lent i poc hàbil però fort i assetjador.

4.6.3 Boos

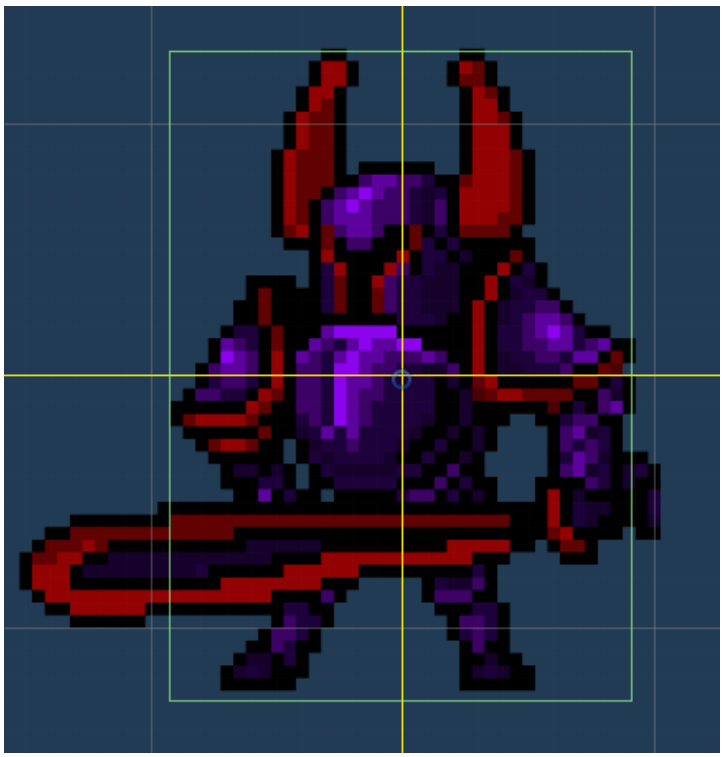


Figura 34: Prefab del Boss final de cada nivell

El Boos és l'enemic gran, amb més vida i estèticament és igual als enemics normals. És diferència del minotaure pel filtre lila i la diferencia dimensió.
Aquest enemic sempre està a una sala final on només es pot entrar per una porta.

4.7 Menus

Dungeon Mobile consta de tres menús, el menú inicial, aquell que es veu al obrir el joc i cada vegada que es el jugador mor, el menú de configuración i el menú de pausa, que són aquells que et permeten pausar el joc, modificar el volum i tornar al joc.

El Menú inicial (figura 35) per poder començar a jugar consta de tres botons:

- Start, que inicia el joc
- Exit, tanca l'aplicació
- Configuració, que obre un pop-up on pots modificar el volum de la música.

Figura 35: captura in-game del menú d'inici

El menú de configuració és el mateix que el de pausa(figura 36) consta d'un slider per modificar la música i un botó per tornar al joc.

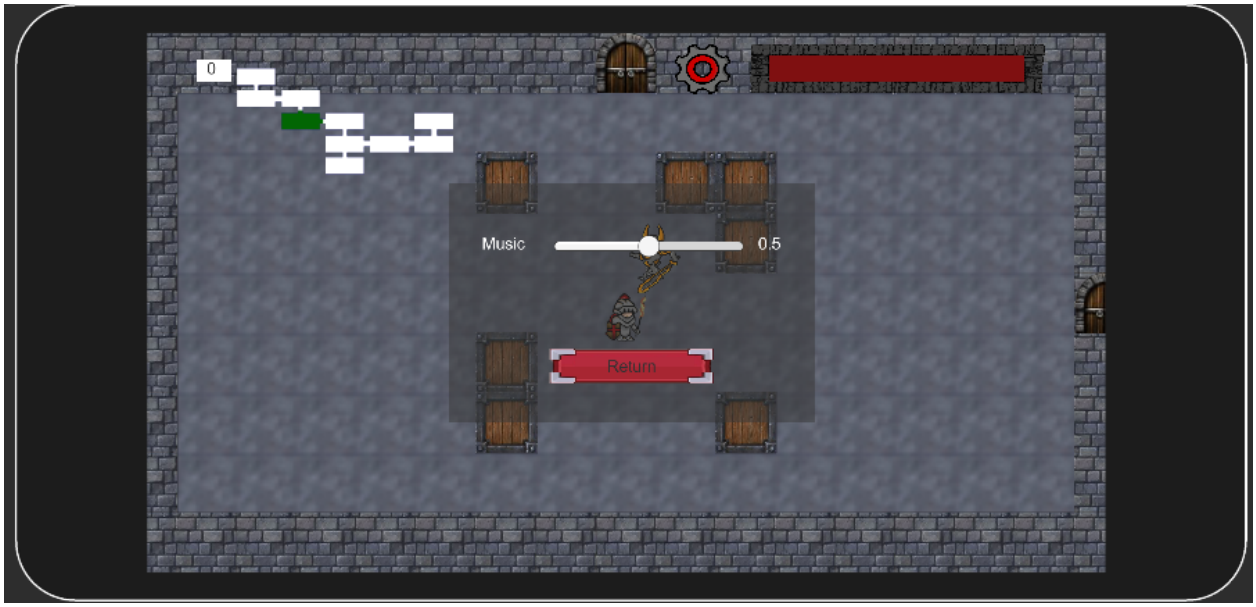


Figura 36: captura in-game del menú de pausa i configuració del só

5. Implementació i proves

5.1 Unity Assets Utilitzats:

Joystick Pack

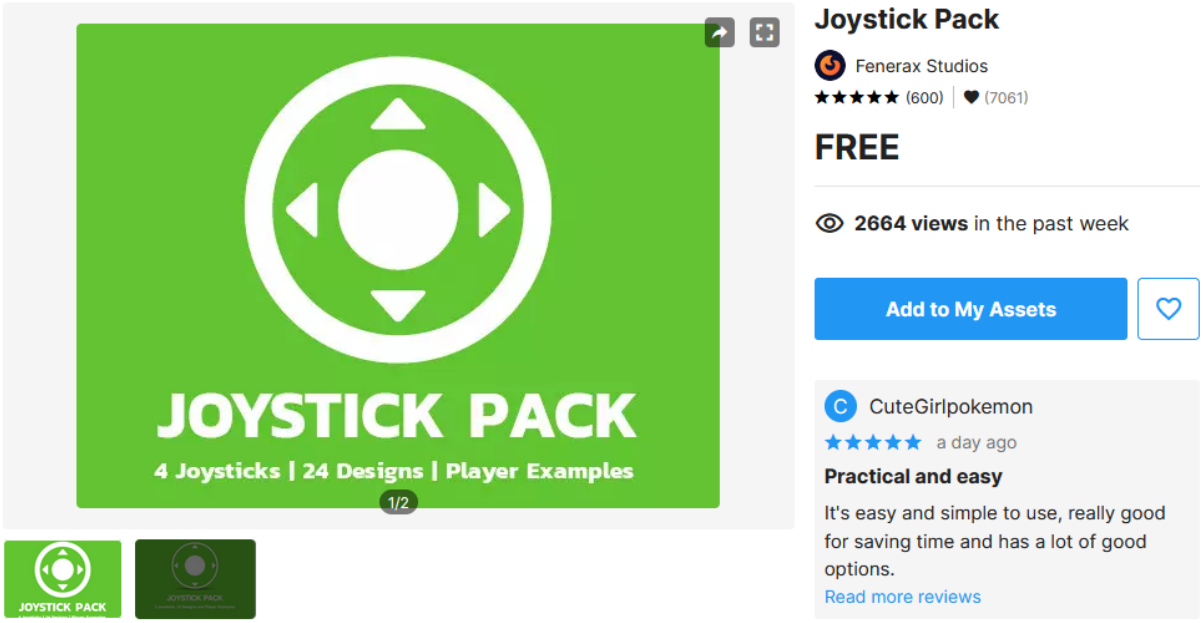


Figura 37: Imatge representativa del asset "Joystick pack"

Per l'implementació del joystick amb els que el jugador pot moure, apuntar i disparar s'ha utilitzat el asset d'unity "Joystick Pack" que ens incorpora cuatre tipus de joysticks diferents:

- Dynamic Joystick: Joysticks amb posició variable segons la posició on es clica, a més si mantens clicada la pantalla i vas canviant de posició el joystick es va movent cap a la posició del dit.
- Fixed Joystick: Són joysticks amb posició fixa i on només es pot interaccionar amb el joystick si cliques sobre ell.
- Variable Joystick: També són joysticks amb posició fixa, però es pot definir una àrea, on no cal que es cliqui exactament sobre el joystick sinó que depenent de la distància sobre el joystick ja detecta la direcció sobre aquest.
- Floating Joystick: Els joysticks flotants són els que finalment s'ha implementat, ja que s'ha decidit que només es veurà el joystick, si es clica a la pantalla per poder tenir més camp de visió del joc i el joystick es veurà a la posició de la pantalla que sigui clicada del jugador perquè s'adapti a més tipus de mans.

5.2 Scripts principals

5.2.1 Generació de sales i nivells (LevelGeneration.cs)

Per poder generar correctament les sales de cada nivell hem de definir el màxim de sales i un màxim en l'eix X i Y. Seguidament es creen les sales; s'assignen les portes adients i els objectes, es dibuixa el mapa a partir de les sales i finalment se li adjudiquen objectes, power-ups, enemics..

```
void Start () {
    if (numberOfRooms >= (worldSize.x * 2) * (worldSize.y * 2)){ // asegurarnos ed que les sales no es poden fer mes enlla del mon marcat
        numberOfRooms = Mathf.RoundToInt((worldSize.x * 2) * (worldSize.y * 2));
    }
    gridSizeX = Mathf.RoundToInt(worldSize.x);
    gridSizeY = Mathf.RoundToInt(worldSize.y);
    CreateRooms(); //crea la forma del mapa
    SetRoomDoors(); //assigna les portes adients
    DrawMap(); //instancia les imatges per crear el mapa

    GetComponent<SheetAssigner>().Assign(rooms); //passa la informacio de les "rooms" als scripts adients
}
```

Figura 38: Base del script on es criden per ordre a les diferents funcions necessàries per crear el nivell.

Una de les funcions principals és *createRooms* (figura 39). Aquesta crea sales a partir d'una instancia d'un prefab "sala". Mitjançant un número aleatori, se li assigna una posició dins el nivell, es comprova que aquesta és possible i es crea la nova sala.

```
void CreateRooms(){
    //setup
    rooms = new Room[gridSizeX * 2,gridSizeY * 2];
    rooms[gridSizeX,gridSizeY] = new Room(Vector2.zero, 1);
    takenPositions.Insert(0,Vector2.zero);
    Vector2 checkPos = Vector2.zero;
    //magic numbers
    float randomCompare = 0.2f, randomCompareStart = 0.2f, randomCompareEnd = 0.01f;
    //add rooms
    for (int i =0; i < numberOfRooms -1; i++){
        float randomPerc = ((float) i) / (((float)numberOfRooms - 1));
        randomCompare = Mathf.Lerp(randomCompareStart, randomCompareEnd, randomPerc);
        //grab new position
        checkPos = NewPosition();
        //test new position
        if (NumberOfNeighbors(checkPos, takenPositions) > 1 && Random.value > randomCompare){
            int iterations = 0;
            do{
                checkPos = SelectiveNewPosition();
                iterations++;
            }while(NumberOfNeighbors(checkPos, takenPositions) > 1 && iterations < 100);
            if (iterations >= 50)
                print("error: could not create with fewer neighbors than : " + NumberOfNeighbors(checkPos, takenPositions));
        }
        //finalize position
        rooms[(int) checkPos.x + gridSizeX, (int) checkPos.y + gridSizeY] = new Room(checkPos, 0);
        takenPositions.Insert(0,checkPos);
    }
}
```

Figura 39: funcio createRooms del script "LevelGeneration.cs"

La funció *CreateRooms* crida a 2 funcions més, *NewPosition*(figura 40) i *SelectiveNewPosition* (figura 41), la primera per crear una nova posició i la segona per crear una posició a partir d'una sala amb un un sol veí.

```
Vector2 NewPosition(){
    int x = 0, y = 0;
    Vector2 checkingPos = Vector2.zero;
    do{
        int index = Mathf.RoundToInt(Random.value * (takenPositions.Count - 1)); // pick a random room
        x = (int) takenPositions[index].x;//capture its x, y position
        y = (int) takenPositions[index].y;
        bool UpDown = (Random.value < 0.5f);//randomly pick wether to look on hor or vert axis
        bool positive = (Random.value < 0.5f);//pick whether to be positive or negative on that axis
        if (UpDown){ //find the position bnased on the above bools
            if (positive){
                y += 1;
            }else{
                y -= 1;
            }
        }else{
            if (positive){
                x += 1;
            }else{
                x -= 1;
            }
        }
        checkingPos = new Vector2(x,y);
    }while (takenPositions.Contains(checkingPos) || x >= gridSizeX || x < -gridSizeX || y >= gridSizeY || y < -gridSizeY);
    return checkingPos;
}
```

Figura 40: Funció NewPosition de l'script "LevelGeneration.cs"

```
Vector2 SelectiveNewPosition(){ // method differs from the above in the two commented ways
    int index = 0, inc = 0;
    int x =0, y =0;
    Vector2 checkingPos = Vector2.zero;
    do{
        inc = 0;
        do{
            //instead of getting a room to find an adject empty space, we start with one that only
            //as one neighbor. This will make it more likely that it returns a room that branches out
            index = Mathf.RoundToInt(Random.value * (takenPositions.Count - 1));
            inc ++;
        }while (NumberOfNeighbors(takenPositions[index], takenPositions) > 1 && inc < 100);
        x = (int) takenPositions[index].x;
        y = (int) takenPositions[index].y;
        bool UpDown = (Random.value < 0.5f);
        bool positive = (Random.value < 0.5f);
        if (UpDown){
            if (positive){
                y += 1;
            }else{
                y -= 1;
            }
        }else{
            if (positive){
                x += 1;
            }else{
                x -= 1;
            }
        }
        checkingPos = new Vector2(x,y);
    }while (takenPositions.Contains(checkingPos) || x >= gridSizeX || x < -gridSizeX || y >= gridSizeY || y < -gridSizeY);
    if (inc >= 100){ // break loop if it takes too long: this loop isnt garuanteed to find solution, which is fine for this
        print("Error: could not find position with only one neighbor");
    }
    return checkingPos;
}
```

Figura 41: Funció SelectiveNewPosition de l'script "LevelGeneration.cs"

La funció *NumberOfNeighbors* es crida varies vegades durant tot l'script per comprovar quants veïns té la sala.

```
int NumberOfNeighbors(Vector2 checkingPos, List<Vector2> usedPositions){
    int ret = 0; // start at zero, add 1 for each side there is already a room
    if (usedPositions.Contains(checkingPos + Vector2.right)){ //using Vector.[direction] as short hands, for simplicity
        ret++;
    }
    if (usedPositions.Contains(checkingPos + Vector2.left)){
        ret++;
    }
    if (usedPositions.Contains(checkingPos + Vector2.up)){
        ret++;
    }
    if (usedPositions.Contains(checkingPos + Vector2.down)){
        ret++;
    }
    return ret;
}
```

Figura 42: Funció NumberOfNeighbors de l'script "LevelGeneration.cs"

La funció *Drawmap* (figura 43) que recorre les sales i crea l'objecte identificador de cada una en el mapa.

```
void DrawMap(){
    foreach (Room room in rooms){
        if (room == null){
            continue; //skip where there is no room
        }
        Vector2 drawPos = room.gridPos;
        drawPos.x *= 16;//aspect ratio of map sprite
        drawPos.y *= 8;
        //create map obj and assign its variables
        MapSpriteSelector mapper = Object.Instantiate(roomWhiteObj, drawPos, Quaternion.identity).GetComponent<MapSpriteSelector>();
        mapper.type = room.type;
        mapper.up = room.doorTop;
        mapper.down = room.doorBot;
        mapper.right = room.doorRight;
        mapper.left = room.doorLeft;
        mapper.gameObject.transform.parent = mapRoot;
        room.AssignMapper(mapper);
    }
}
```

Figura 43: Funció DrawMap de l'script "LevelGeneration.cs"

La següent funció és *SetRoomDoors* (figura 44), una funció sencilla pero clau a l'hora de crear el nivell ja que es la que comprova quants veïns te cada sala i li assigna una porta a cada un d'aquests.

Finalment es crida al script *SheetAssigner.cs* que asigna tots els objects, enemics, power-ups de cada sala.

```
void SetRoomDoors(){
    for (int x = 0; x < ((gridSizeX * 2)); x++){
        for (int y = 0; y < ((gridSizeY * 2)); y++){
            if (rooms[x,y] == null){
                continue;
            }
            Vector2 gridPosition = new Vector2(x,y);
            if (y - 1 < 0){ //check above
                rooms[x,y].doorBot = false;
            }else{
                rooms[x,y].doorBot = (rooms[x,y-1] != null);
            }
            if (y + 1 >= gridSizeY * 2){ //check bellow
                rooms[x,y].doorTop = false;
            }else{
                rooms[x,y].doorTop = (rooms[x,y+1] != null);
            }
            if (x - 1 < 0){ //check left
                rooms[x,y].doorLeft = false;
            }else{
                rooms[x,y].doorLeft = (rooms[x - 1,y] != null);
            }
            if (x + 1 >= gridSizeX * 2){ //check right
                rooms[x,y].doorRight = false;
            }else{
                rooms[x,y].doorRight = (rooms[x+1,y] != null);
            }
        }
    }
}
```

Figura 44: Funció SetRoomDoors de l'script "LevelGeneration.cs"

5.3 Minimapa:

Figura 45: minimapa una vegada ja creat

Per crear el minimapa es crida al Script *MapSpriteSelector.cs* on la funció principal és PickSprite (figures 46 i 47) que segons les portes que tinguin la sala, selecciona el sprite adequat per afegir-lo al minimapa.

```
void PickSprite(){ //picks correct sprite based on the four door bools
    if (up){
        if (down){
            if (right){
                if (left){
                    rend.sprite = spUDRL;
                }else{
                    rend.sprite = spDRU;
                }
            }else if (left){
                rend.sprite = spULD;
            }else{
                rend.sprite = spUD;
            }
        }else{
            if (right){
                if (left){
                    rend.sprite = spRUL;
                }else{
                    rend.sprite = spUR;
                }
            }else if (left){
                rend.sprite = spUL;
            }else{
                rend.sprite = spU;
            }
        }
        return;
    }
    if (down){
```

```
        if (down){
            if (right){
                if(left){
                    rend.sprite = spLDR;
                }else{
                    rend.sprite = spDR;
                }
            }else if (left){
                rend.sprite = spDL;
            }else{
                rend.sprite = spD;
            }
            return;
        }
        if (right){
            if (left){
                rend.sprite = spRL;
            }else{
                rend.sprite = spR;
            }
        }else{
            rend.sprite = spL;
        }
    }
```

Figures 46 i 47: funció PickSprite del Script MapSpriteSelector.cs

5.3.1 Personatge Principal (controllerPlayer.cs)

Per actualitzar el moviment del personatge principal aprofitarem el Joystick Pack dels Assets d'Unity per detectar els tocs en pantalla. S'ha dividit en dues parts la pantalla del movil i a cada una se li assigna un Joystick flotant que a partir de la posició on el jugador toca amb cada dit es tracta de manera diferent: la part esquerra tracta la direcció de moviment del personatge i la part dreta tracta la direcció on apunta el personatge.

```
horizontalMove = joystick.Horizontal * velocityMovement;
transform.position += new Vector3(joystick.Horizontal, 0, 0) * Time.deltaTime * velocityMovement;
Anim.SetFloat("movX", horizontalMove);
verticalMove = joystick.Vertical * velocityMovement;
transform.position += new Vector3(0, joystick.Vertical, 0) * Time.deltaTime * velocityMovement;
Anim.SetFloat("movY", horizontalMove);
```

Figura 48: input que tracta el moviment del personatge

```
Vector2 lookDir2 = new Vector2(direccion.Horizontal, direccion.Vertical) - new Vector2(transform.position.x, transform.position.x);
float anglePlayer = Mathf.Atan2(direccion.Vertical, direccion.Horizontal) * Mathf.Rad2Deg;//-90f
transform.eulerAngles =  new Vector3(0,0, anglePlayer);
```

Figura 49: input que tracta el angle

```
private void FixedUpdate()//movement of the player by the joystick
{
    horizontalMove = joystick.Horizontal * velocityMovement;
    transform.position += new Vector3(joystick.Horizontal, 0, 0) * Time.deltaTime * velocityMovement;
    Anim.SetFloat("movX", horizontalMove);
    verticalMove = joystick.Vertical * velocityMovement;
    transform.position += new Vector3(0, joystick.Vertical, 0) * Time.deltaTime * velocityMovement;
    Anim.SetFloat("movY", horizontalMove);

    if (horizontalMove !=0 || verticalMove != 0)
    {
        Anim.enabled = true;
        Anim.gameObject.GetComponent<Animator>().enabled = true;
        Anim.SetBool("is_moving", true);

    }
    else
    {
        Anim.SetBool("is_moving", false);
        Anim.gameObject.GetComponent<Animator>().enabled = false;
        Anim.gameObject.GetComponent<Animator>().enabled = true;

    }


    Anim.SetFloat("speed", Math.Abs(horizontalMove + verticalMove));
    Anim.gameObject.GetComponent<Animator>().enabled = false;
    Anim.gameObject.GetComponent<Animator>().enabled = true;
    Anim.SetBool("alive", isAlive());



    Vector2 lookDir2 = new Vector2(direccion.Horizontal, direccion.Vertical) - new Vector2(transform.position.x, transform.position.x);
    float anglePlayer = Mathf.Atan2(direccion.Vertical, direccion.Horizontal) * Mathf.Rad2Deg;//-90f
    transform.eulerAngles =  new Vector3(0,0, anglePlayer);


    if (CanShot && anglePlayer != 0)
    {
        shoot();
        CanShot = false;
        Invoke("coolDownShot", 1f);
    }
}
```

Figura 50: funció FixedUpdate de l'script "ControllerPlayer''

La versió fixedUpdate es una funció que s'executa per cada frame que s'actualitza del joc i que en primer moment actualitza la posició del jugador a partir del joysticks mencionats anteriorment, i a més. actualitza les animacions del personatge principal.

Els power-up es detecten quan el personatge colisiona amb aquests:

```
private void OnCollisionEnter2D(Collision2D collision)
{

    //Debug.Log(collision.gameObject.tag);
    if (collision.gameObject.tag == "vida")
    {
        health += 20;
        vidaUI.setHealth(health);
        Destroy(collision.gameObject);
    }
    else if (collision.gameObject.tag == "speed_hab")
    {
        speed_pl = 3;
        Destroy(collision.gameObject);
        Invoke("changeSpeed1", 10f);
    }else if (collision.gameObject.tag == "multiple_hab")
    {
        multipleShoot = true;
        Destroy(collision.gameObject);
        Invoke("changeMultipleShoot", 15f);
    }
}
```

Figura 51: funció OnCollisionEnter2D de l'script "ControllerPlayer"

La funció *OnTriggerEnter2D* serveix per comprovar a quina porta entra i que ha de fer en cada cas, a més aquest script també es el que controla el nivell, el dispars i el temps entre aquests, el mal que pot fer i rebre, etc.

Aquesta funció detecta les portes a les quals entra el personatge comparant la posició de el personatge amb el centre de la càmera. És a dir, quan toca una porta, la posició del personatge en comparació a la porta serà negativa o positiva en els eixos X i Y.

```
Vector3 tempPos = GameObject.Find("MainCamera").GetComponent<Camera>().transform.position;

string PuertaEntrada = "ninguna";

float tileSize = 16;

if (transform.position.x > maincam.transform.position.x + 20)//Right door
{
    //Debug.Log("Right door");
    PuertaEntrada = "Right";
    tempPos += Vector3.right * horMove * moveJump.x; //jump bnetween rooms based opn input

}
else if (transform.position.x < maincam.transform.position.x - 20)//Left door
{
    //Debug.Log("Left door");
    PuertaEntrada = "Left";
    tempPos += Vector3.right * horMove * moveJump.x; //jump bnetween rooms based opn input
    //tempPos += Vector3.up * vertMove * moveJump.y;
}
else if (transform.position.y > maincam.transform.position.y + 20)//Up door
{
    //Debug.Log("Up door");
    PuertaEntrada = "Up";
    //tempPos += Vector3.right * horMove * moveJump.x; //jump bnetween rooms based opn input
    tempPos += Vector3.up * vertMove * moveJump.y;
}
else if (transform.position.y < maincam.transform.position.y - 20)//down door
{
    //Debug.Log("down door");
    PuertaEntrada = "Down";
    // tempPos += Vector3.right * horMove * moveJump.x; //jump bnetween rooms based opn input
    tempPos += Vector3.up * vertMove * moveJump.y;
}

//--------------------------------------mover la camara a la pantalla que toque--------------------
//Debug.Log("tempPos2"+ tempPos);
GameObject.Find("MainCamera").GetComponent<Camera>().transform.position = tempPos;
transform.position = tempPos;
```

Figura 52: primera part de la funció OnTriggerEnter2D del script controllerPlayer

Seguidament, com ha guardat a quina porta ha entrat a la sala anterior, posa el personatge a la posició adequada de la sala per simular que les portes estan connectades.

```
//---------------------------------------mover al lado que toque de la pantalla-----------------
if (PuertaEntrada == "Right")
{
    Vector3 aux = tempPos;
    aux.x = aux.x - (tileSize * 7);
    aux.z = 0;
    transform.position = aux;
}
else if (PuertaEntrada == "Left")
{
    Vector3 aux = tempPos;
    aux.x = aux.x + (tileSize * 7);
    aux.z = 0;
    transform.position = aux;
}
else if (PuertaEntrada == "Up")
{
    Vector3 aux = tempPos;
    aux.y = aux.y - (tileSize * 2);
    aux.z = 0;
    transform.position = aux;
}
else if (PuertaEntrada == "Down")//fet
{
    Vector3 aux = tempPos;
    aux.y = aux.y + (tileSize * 2);
    aux.z = 0;
    transform.position = aux;
}
else//mover al centro de la pantalla
{
    Vector3 aux = tempPos;
    aux.y = aux.y - 24;
    aux.z = 0;
    transform.position = aux;
}
```

Figura 53: segona part de la funció OnTriggerEnter2D del script controlerPlayer

La funció shoot es crida si es manté apretat el joystick de la dreta. Depenent de si està activat o no el power-up que modifica els trets, cridarà la funció *“bullet_shoot()”* (figura 55)o la funció *“bullet_3shoot()”* (figura 56).

```
public void shoot()
{
    //horMove = joystick.Horizontal;//capture input
    //vertMove = joystick.Vertical;
    Anim.SetBool("shooting", true);

    // bullet_shoot();
    if (multipleShoot) bullet_3shoot();
    else bullet_shoot();
    //raycastShoot();

}
```

Figura 54:funció shoot del script controllerPlayer

La funció *bullet_shoot* (figura 55) instancia el prefab de la llança en la posició i orientació del jugador, a més de afegir-li la força en l'angle corresponent.

```
void bullet_shoot()
{

    //Vector3 direccio = new Vector3(joystick.Horizontal, joystick.Vertical, 0);
    Vector2 lookDir2 = new Vector2(direccion.Horizontal, direccion.Vertical) - new Vector2(transform.position.x, transform.position.x);
    float anglePlayer = Mathf.Atan2(direccion.Vertical, direccion.Horizontal) * Mathf.Rad2Deg;
    float direccio = joystick.Horizontal + joystick.Vertical;
    //Debug.Log(gun.rotation);
    GameObject bullet =  Instantiate(bulletPrefab, gun.position, gun.rotation);
    Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();
    rb.AddForce(gun.right * 30f, ForceMode2D.Impulse);

    Destroy(bullet, 7f);
    Anim.SetBool("shooting", false);
    Anim.gameObject.GetComponent<Animator>().enabled = false;
    //Destroy(bullet, 5f);


}
```

Figura 55: funció bullet_shoot del script controllerPlayer

En canvi, la funció *bullet_3shoot* (figura 56) instancia 3 llances en 3 direccions diferents adaptades a la posició i direcció a la que el jugador esta apuntant.

```
void bullet_3shoot()
{

    //Vector3 direccio = new Vector3(joystick.Horizontal, joystick.Vertical, 0);
    Vector2 lookDir2 = new Vector2(direccion.Horizontal, direccion.Vertical) - new Vector2(transform.position.x, transform.position.x);
    float anglePlayer = Mathf.Atan2(direccion.Vertical, direccion.Horizontal) * Mathf.Rad2Deg;
    float direccio = joystick.Horizontal + joystick.Vertical;

    GameObject bullet = Instantiate(bulletPrefab, gun2.position, gun2.rotation);
    GameObject bullet2 = Instantiate(bulletPrefab, gun.position, gun.rotation);
    GameObject bullet3 = Instantiate(bulletPrefab, gun3.position, gun3.rotation);

    Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();
    rb.AddForce(gun2.right * 30f, ForceMode2D.Impulse);

    Rigidbody2D rb2 = bullet2.GetComponent<Rigidbody2D>();
    rb2.AddForce(gun.right * 30f, ForceMode2D.Impulse);

    Rigidbody2D rb3 = bullet3.GetComponent<Rigidbody2D>();
    rb3.AddForce(gun3.right * 30f, ForceMode2D.Impulse);

    Destroy(bullet, 7f);
    Destroy(bullet2, 7f);
    Destroy(bullet3, 7f);
    Anim.SetBool("shooting", false);
    Anim.gameObject.GetComponent<Animator>().enabled = false;
    //Destroy(bullet, 5f);

}
```

Figura 56: funció bullet_3shoot del script controllerPlayer

5.3.2 SheetAssigner.cs

Aquest Script es basa en una funció principal on es passen les instàncies de les portes i els prefabs necessaris dels objectes i enemics que poden aparèixer en cada sala.

```
foreach (Room room in rooms){
    //skip point where there is no room
    if (room == null){
        continue;
    }
    //pick a random index for the array
    int index = Mathf.RoundToInt(Random.value * (sheetsNormal.Length -1));
    //find position to place room
    Vector3 pos = new Vector3(room.gridPos.x * (roomDimensions.x + gutterSize.x), room.gridPos.y * (roomDimensions.y + gutterSize.y), 0);
    RoomInstance myRoom = Instantiate(listRoomObj[controllerPlayer.lvlActual], pos, Quaternion.identity).GetComponent<RoomInstance>();
    //lvlActual = gameObject.GetComponent<controllerPlayer>().returnlvl();
    //Debug.Log("lvlActual: " + lvlActual);
    myRoom.Setup(sheetsNormal[index], room.gridPos, room.type, room.doorTop, room.doorBot, room.doorLeft, room.doorRight, prefabBoos,
        prefabVida, prefabMoreShoots, prefabMprespeed, room.MapSprite, spritesParedsXLevel);
}
```

Figura 57: funció assign del script SheetAssigner

5.3.3 RoomInstance.cs

En un inici es guarden tots els valors, objectes, prefabs i textures necessàries amb aquestes i a base de diferents funcions s'instancien diferents prefabs.

```
Setup(Texture2D _tex, Vector2 _gridPos, int _type, bool _doorTop, bool _doorBot, bool _doorLeft, bool _doorRight,
```

```
, GameObject _prefBoos, GameObject _prefVida, GameObject _prefmoreshoots, GameObject _moreSpeed, MapSpriteSelector _mapSprite, GameObject[] _spritesPared)
```

Figura 58 i 59: paràmetres necessaris que arriben a la funció setup

La funció *setup* es crida desde *sheetAssigner* i és des d'on s'inicial·litzen tots els valors de l'escenari per cada una de les instàncies de les sales.

```
myRoom.Setup(sheetsNormal[index], room.gridPos, room.type, room.doorTop, room.doorBot, room.doorLeft,
```

```
room.doorRight, prefabBoos, prefabVida, prefabMoreShoots, prefabMprespeed, room.MapSprite, spritesParedsXLevel);
```

Figura 60 i 61: paràmetres necessaris que se li passen a la funció setup

Seguint amb la funció Setup, es guarden tot els valors necessaris per crear una sala, aquests valors es passen des de la funció "SheetAssigner". Seguidament, a partir d'un nombre aleatori entre el 0 i el 8, es decideix si s'instanciarà un power-up i en aquest cas s'escull quin power-up sortirà en aquella sala. En concret, si surt un 1, s'instància el power-up de vida, si surt el 2, s'instància el power-up que canvia a triple tret, si surt el 3, s'instància el power up de més

velocitat de moviment i la resta de números no instància res perquè sigui menys probable que una sala tingui power-ups i sigui més complicat el joc. Seguidament, es comprova que s'hagi creat un Boos en aquell nivell, en cas de no haver-hi cap i estar en una instància d'una sala que sigui un extrem del mapa, es crea un Boos, en cas contrari es creen enemics a la sala.

```
tex = _tex;
gridPos = _gridPos;
type = _type;
doorTop = _doorTop;
doorBot = _doorBot;
doorLeft = _doorLeft;
doorRight = _doorRight;
BOOS = _prefBoos;
vida = _prefVida;
moreshoots = _prefmoreshoots;
moreSpeed = _moreSpeed;
MapSprite = _mapSprite; //assignra la textutra del mapa
spritesParedsXLevel = _spritesPared;
MakeDoors();
GenerateRoomTiles();

int num = Random.Range(0, 8);
if (num == 1) InstantiateVida();
else if(num == 2) InstantiateShotsHab();
else if (num == 3) InstantiateSpeedHab();

bool boosCreat = GameObject.FindGameObjectWithTag("BOOS");
//Debug.Log(boosCreat);
if (!boosCreat)
{
    bool boosBool = InstantiateBOOS();//instantiate the enemies in the map
    if (!boosBool) InstantiateEnemies();//instantiate the enemies in the map
}
else
{
    InstantiateEnemies();//instantiate the enemies in the map
}
```

Figura 62: funció Setup del script “RoomInstance”

Dins d’aquesta es criden a les següents funcions que instancien els prefabs adequats.

```
private void InstantiateVida()
{
    //vida = Resources.Load("Prefabs/vida") as GameObject;
    //vida = Resources.Load<GameObject>("ASSETS/Prefabs/vida");
    //Debug.Log(roomSizeInTiles.y);
    Instantiate(vida, transform.position + Vector3.up * (1 * tileSize) , Quaternion.identity);
}

private void InstantiateShotsHab()
{

    Instantiate(moreshoots, transform.position + Vector3.up * (1 * tileSize), Quaternion.identity);
}

private void InstantiateSpeedHab()
{

    Instantiate(moreSpeed, transform.position + Vector3.up * (1 * tileSize), Quaternion.identity);
}
```

Figura 63: funció que Instancien power-ups del script "RoomInstance"

```
private void InstantiateEnemies()
{
    int n = Random.Range(1, 5);
    for (int i = 0; i < n; i++)
    {

        bool enemicCreat = false;
        int z = 0;
        while (!enemicCreat && z< 15) {
            //int x = Random.Range(1, tex.width);
            int x = Random.Range(-7, 7);
            //int y = Random.Range(1, tex.height);
            int y = Random.Range(-3, 3);
            //Debug.Log("x: " + x + "y: " + y);
            //buscar posicion aleatoria vacia(sin objetos)----------------------------------------------------------------------------------------------------
            Color pixelColor = tex.GetPixel(x, y);
            int signe = Random.Range(0, 4);
            if (signe == 0)
            {
                pixelColor = tex.GetPixel(x , y);

                Vector3 auxX = transform.position + Vector3.up * (y * tileSizey) + Vector3.right * (x * tileSizex);

                if (pixelColor.a == 0)
                {

                    GameObject auxEnemy = Instantiate(enemy, transform.position + Vector3.up * (y * tileSizey) + Vector3.right * (x * tileSizex), Quaternion.identity);
                    new WaitForSeconds(1);
                    Enemy auxEnemy_2 = auxEnemy.GetComponent<Enemy>();
                    auxEnemy_2.definirMaxSala(transform.position.x, transform.position.y);
                    if (auxEnemy_2.detectColisionwithObjectMapa()) Destroy(auxEnemy);
                    else  enemicCreat = true;
                }
                else { z++; }
            }
            else if (signe == 1)
```

```
else if (signe == 1)
{
    pixelColor = tex.GetPixel(-x, y);

    Vector3 auxX = transform.position + Vector3.up * (y * tileSizey) - Vector3.right * (x * tileSizex);

    detection(auxX);

    if (pixelColor.a == 0)
    {
        GameObject auxEnemy = Instantiate(enemy, transform.position + Vector3.up * (y * tileSizey) + Vector3.left * (x * tileSizex), Quaternion.identity);
        new WaitForSeconds(1);
        Enemy auxEnemy_2 = auxEnemy.GetComponent<Enemy>();
        auxEnemy_2.definirMaxSala(transform.position.x, transform.position.y);
        if (auxEnemy_2.detectColisionwithObjectMapa()) Destroy(auxEnemy);
        else enemicCreat = true;
    }
    else { z++; }
}

else if (signe == 2)
```

```
else if (signe == 2)
{
    pixelColor = tex.GetPixel(x, -y);

    Vector3 auxX = transform.position + Vector3.down * (y * tileSizey) + Vector3.right * (x * tileSizex);

    if (pixelColor.a == 0)
    {
        GameObject auxEnemy = Instantiate(enemy, transform.position + Vector3.down * (y * tileSizey) + Vector3.right * (x * tileSizex), Quaternion.identity);
        new WaitForSeconds(1);
        Enemy auxEnemy_2 = auxEnemy.GetComponent<Enemy>();
        auxEnemy_2.definirMaxSala(transform.position.x, transform.position.y);
        if (auxEnemy_2.detectColisionwithObjectMapa()) Destroy(auxEnemy);
        else enemicCreat = true;
    }
    else { z++; }

}
else
{
```

```
            else
            {
                pixelColor = tex.GetPixel(-x , -y);

                Vector3 auxX = transform.position + Vector3.down * (y * tileSizey) - Vector3.right * (x * tileSizex);
                auxX.z = 2f;

                if (pixelColor.a == 0)
                {
                    GameObject auxEnemy = Instantiate(enemy, transform.position + Vector3.down * (y * tileSizey) + Vector3.left * (x * tileSizex), Quaternion.identity);
                    new WaitForSeconds(1);
                    Enemy auxEnemy_2 = auxEnemy.GetComponent<Enemy>();
                    auxEnemy_2.definirMaxSala(transform.position.x, transform.position.y);
                    if (auxEnemy_2.detectColisionwithObjectMapa()) Destroy(auxEnemy);
                    else enemicCreat = true;
                }
                else { z++; }

            }

        }
    }
}
```

Figures 64, 65, 66, 67: funció InstantiateEnemies del script "RoomInstance"

La funció *InstantiateBOOS* (figura 68) es crida quan ja s'ha comprovat que compleix el requisit per instanciar un BOOS final de nivell.

```
private bool InstantiateBOOS()
{
    int i = 0;
    if (doorTop) i++;
    if (doorLeft) i++;
    if (doorRight) i++;
    if (doorBot) i++;

    if (i == 1)
    {
        //BOOS = Resources.Load<GameObject>("ASSETS/Prefabs/BOOS");
        //var textFile = Resources.Load<TextAsset>("Text/textFile01");

        Instantiate(BOOS, transform.position , Quaternion.identity);
        return true;
    }
    else
    {
        return false;
    }


}
```

Figura 68: funció InstantiateBoos del script "RoomInstance"

La funció *makeDoors*(figura 69) posa les instancies necessaries de les portes per cada sala perque el jugador es pugui moure entre sales.

```
void MakeDoors(){
    //top door, get position then spawn
    Vector3 spawnPos = transform.position + Vector3.up*(roomSizeInTiles.y/4 * tileSize) - Vector3.up*(tileSize/4);
    PlaceDoor(spawnPos, doorTop, doorU);
    //bottom door
    spawnPos = transform.position + Vector3.down*(roomSizeInTiles.y/4 * tileSize) - Vector3.down*(tileSize/4);
    PlaceDoor(spawnPos, doorBot, doorD);
    //right door
    spawnPos = transform.position + Vector3.right*(roomSizeInTiles.x * tileSize) - Vector3.right*(tileSize);
    PlaceDoor(spawnPos, doorRight, doorR);
    //left door
    spawnPos = transform.position + Vector3.left*(roomSizeInTiles.x * tileSize) - Vector3.left*(tileSize);
    PlaceDoor(spawnPos, doorLeft, doorL);
}
```

Figura 69: funció MakeDoors del script "RoomInstance"

la funció GenerateRoomTiles (figura 70) recorre l'imatge de la sala on té diferents colors que permeten identificar el tipus d'objecte que s'ha de colocar i aquesta crida a la funció GenerateTile que posa el tile adequat per cada color(obstacle, pared,...)

```
void GenerateRoomTiles(){
    //loop through every pixel of the texture
    for(int x = 0; x < tex.width; x++){
        for (int y = 0; y < tex.height; y++){
            GenerateTile(x,y);
        }
    }
}
void GenerateTile(int x, int y){
    Color pixelColor = tex.GetPixel(x,y);
    //skip clear spaces in texture
    if (pixelColor.a == 0){
        return;
    }
    //find the color to math the pixel
    /*int i = 0;
    foreach (ColorToGameObject mapping in mappings)
    {
        if (i == 0) mapping.prefab = spritePared;
    }*/
    mappings[0].prefab = spritesParedsXLevel[controllerPlayer.lvlActual];
    //mappings[0].prefab = spritesParedsXLevel[1];
    foreach (ColorToGameObject mapping in mappings){
        if (mapping.color.Equals(pixelColor)){
            Vector3 spawnPos = positionFromTileGrid(x,y);
            Instantiate(mapping.prefab, spawnPos, Quaternion.identity).transform.parent = this.transform;//Es crean els tiles de la sala
            //Instantiate(spritesParedsXLevel[1], spawnPos, Quaternion.identity).transform.parent = this.transform;//Es crean els tiles de la sala
        }
    }
}
```

Figura 70: funcions GenerateTiles i GenerateTile del script "RoomInstance"

La funció positionFromTileGrid (figura 71) rep dos valors de posicio i retorna un vector3 amb la posició de la celda.

```
Vector3 positionFromTileGrid(int x, int y){
    Vector3 ret;
    //find difference between the corner of the texture and the center of this object
    Vector3 offset = new Vector3((-roomSizeInTiles.x + 1)*tileSize, (roomSizeInTiles.y/4)*tileSize - (tileSize/4), 0);
    //find scaled up position at the offset
    ret = new Vector3(tileSize * (float) x, -tileSize * (float) y, 0) + offset + transform.position;
    return ret;
}
```

Figura 71: funció positionFromTileGrid del script "RoomInstance"

La funció OnTriggerEnter2D(figura 72) detecta que el jugador entra a la sala per poder cambiar de color al minimapa la sala indicada.

```
private void OnTriggerEnter2D(Collider2D collision)
{

    if (collision.tag == "Player")
    {
        //Debug.Log(collision.tag);
        type = 1;
        MapSprite.PickColor(type);
        this.GetComponent<BoxCollider2D>().enabled = false;
    }

}
```

Figura 72: funció OnTriggerEnter2D del script "RoomInstance"

Les dues següents funcions (figura 73) canvien el color de l'instancia de la sala del minimapa i els valors que detecten si está activada o no.

```
public void changeTypeTo0()
{
    //Debug.Log("changeTypeTo0");
    type = 0;
    MapSprite.PickColor(type);
    this.GetComponent<BoxCollider2D>().enabled = true;
}

public void changeTypeTo1()
{
    //Debug.Log("changeTypeTo1");
    type = 1;
    MapSprite.PickColor(type);
    this.GetComponent<BoxCollider2D>().enabled = false;
}
```

Figura 73: funcions changeTypeTo0 i changeTypeTo1 del script "RoomInstance"

5.3.4 Enemics(Enemy.cs)

Els enemics sempre creen un raycast fins el personatge, en cas què el primer objecte sigui el personatge i estigui dins del rang de moviment, l'enemic es mou en direcció al personatge, en canvi, si l'enemic troba un objecte diferent que no sigui el personatge o simplement el personatge no està dins del rang de moviment del enemic, aquest espera fins que el personatge s'apropi per començar a moure´s. També com l'enemic té un atac cos a cos, quan el personatge està dins del rang d'atac, aquest ataca.

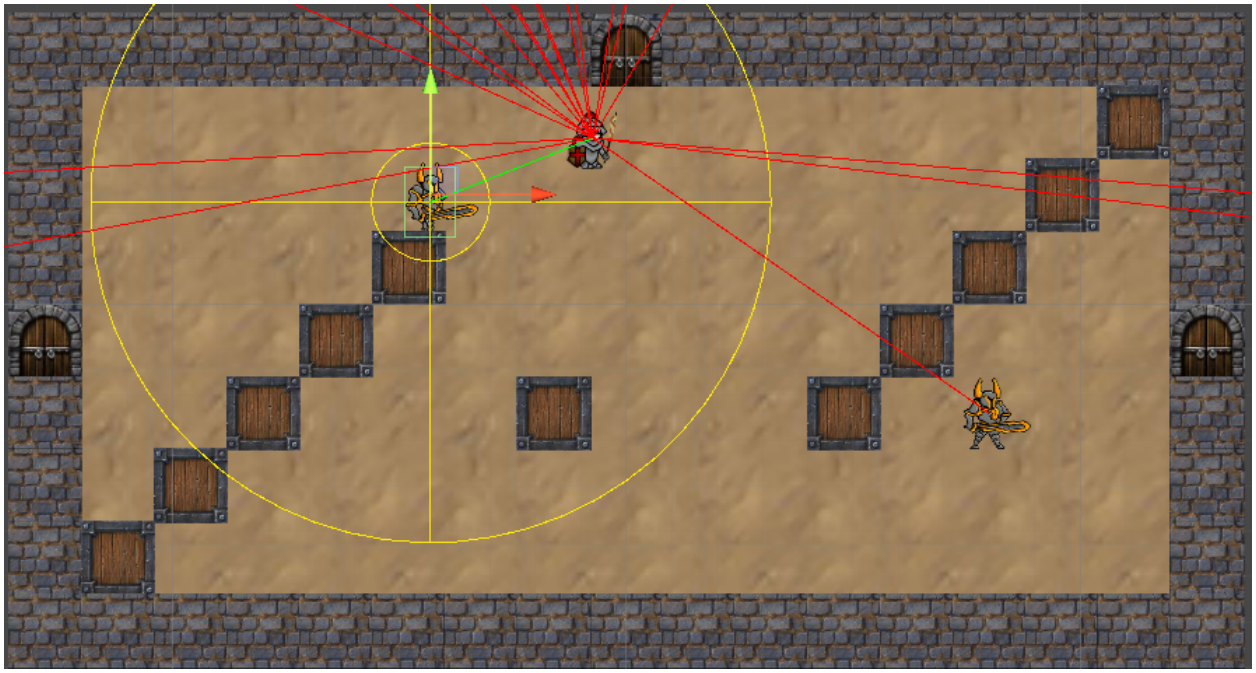


Figura 74: Exemple de l'atac del enemic

```
Vector3 target = initialPosition;

RaycastHit2D hit = Physics2D.Raycast(transform.position, player.transform.position - transform.position, VisionRadius, 1 << LayerMask.NameToLayer("Default"));

Vector3 forward = transform.TransformDirection(player.transform.position - transform.position);

Debug.DrawRay(transform.position, forward, Color.red);
```

Figura 75: codi on es crea el raycast cap al personatge

Si el raycast troba l'enemic el target canvia a la posició del personatge.

```
//si el jugador se encuentra
if(hit.collider != null) {
    if(showDebug)Debug.Log("if: " + hit.collider.tag);
    if (hit.collider.tag == "Player")
    {
        target = player.transform.position;
    }
}
else
{
    if (showDebug) Debug.Log("else: " + hit.collider.tag);
}
```

Figura 76: tros de codi on detecta si l'enemic veu al jugador

En el cas de que el personatge entre al rang, es mou i en el cas d'haver-se mogut torna a la posició inicial. Cada vegada que es mou o ataca s'activa animacions diferents.

```
//si esta en rango de ataque
if (target != initialPosition && distance < AttackRadius)
{
    if (CanAttack)
    {
        attack();
        CanAttack = false;
        Invoke("coolDownShot", 1f);
    }

    Anim.SetBool("is_attacking", true);
    Anim.SetBool("is_moving", false);

}
else
{
    rb2D.MovePosition(transform.position + dir * speed * Time.deltaTime);
    Anim.SetBool("is_attacking", false);
    Anim.SetBool("is_moving", true);
    Anim.SetFloat("movX", dir.x);
    Anim.SetFloat("movY", dir.y);

}

if (target == initialPosition && distance < 0.02f)
{
    transform.position = initialPosition;
    Anim.SetBool("is_moving", false);


}

Debug.DrawLine(transform.position, target, Color.green);
```

Figura 77: tros de codi que serveix per detectar si el jugador esta dins de la distància necessària per atacar

L’enemic no sempre estara al mateix lloc de la sala. Per poder afegir-li moviment als enemics perquè puguin trobar al jugador més fàcilment cada 10 segons l’enemic canvia de posició dins la sala a partir de la funció canviaposicio (figura 78).

```
void canviaPosicio()
{

    float x = UnityEngine.Random.Range(maxSalaX - 112, maxSalaX + 112);
    float y = UnityEngine.Random.Range(maxSalaY - 48, maxSalaY + 48);
    target = new Vector3(x, y, 0);
    initialPosition = target;
    Invoke("canviaposiciobool", 10.0f);
    canviaposicio = false;
}

void canviaposiciobool()
{

    canviaposicio = true;
}
```

Figura 78: funció canviaPosicio per ubicar una nova posició objectiu

5.4 Testejar el joc

5.4.1 Unity Remote 5

Gràcies a l'aplicació “unity remote 5” podem testejar al moment el nostre joc als dispositius mòbils amb sistema operatiu Andro. L’aplicació funciona de forma que executa a l'ordinador el joc pero permet visualitzar-lo desde el dispositiu mòbil, a més de poder interactuar desde la pantalla i botons del dispositiu sense haber de instalar-lo cada vegada que es vol fer un testeig. Això ens dona agilidesa a l’hora d’executarlo moltes vegades però amb l’inconvenient de que compta amb una mica de lag ja que no s’executa desde el mòbil directament.

Per poder executar el joc amb unity remote 5 només cal conectar el dispositiu a l'ordinador, posar el mòbil en mode desenvolupador, obrir l'aplicació al mòbil i donar-li al botó de play a l’unity.

Figura 79: imatge representativa de unity remote 5

5.4.2 Mitjançant una build (APK)

Fer una build del nostre joc i instalarlo a un dispositiu Android ens permet executar el joc en el dispositiu directament i testejar realment si tot funcionará correctament cuan els jugadors se l'instalin directament desde la PlayStore.

Per poder crear una build només cal anar al apartar de *File* d'Unity, dins d'aquest a *Build Settings,* seguidament ens assegurarem que estem en el mode d'android, apretem el botó *Build* de la part inferior i seleccionar la carpeta de destí, això crearea un arxiu que mourem a una carpeta del nostre dispositiu Android. Al obrir aquest arxiu ens instalara el joc i l'inicará per poder jugar i testejar-lo directament al dispositiu.

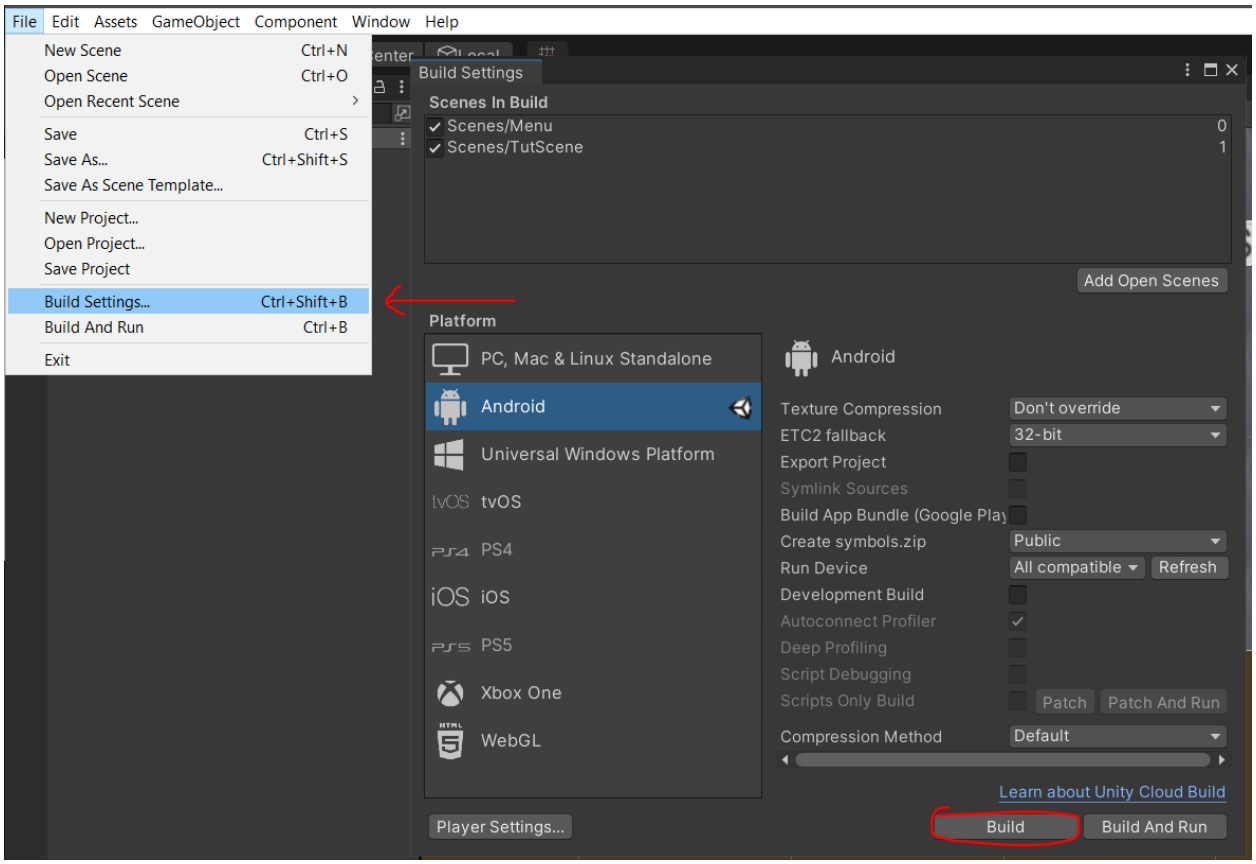


Figura 80: finestra "build settings" de l'unity

5.5 Pujar el joc a Google Play

Per poder pujar el joc a Google play s'han hagut de seguir els següents passos:

- Pas 1

En primer lloc s'ha de crear compte de desarrollador google play i per fer això s'ha de fer un compte de google i pagar 25 dòlars.

link per crear un compte desenvolupador: https://play.google.com/console/

- Pas 2

En segon lloc s'ha de crear els detalls de la app que es vol pujar a partir d'emplenar el següent formulari, on demana dades bàsiques per poder diferenciar l'app, com per exemple, el nom de la app, idioma, si es un joc o una app, si es gratis, entre d'altres() :

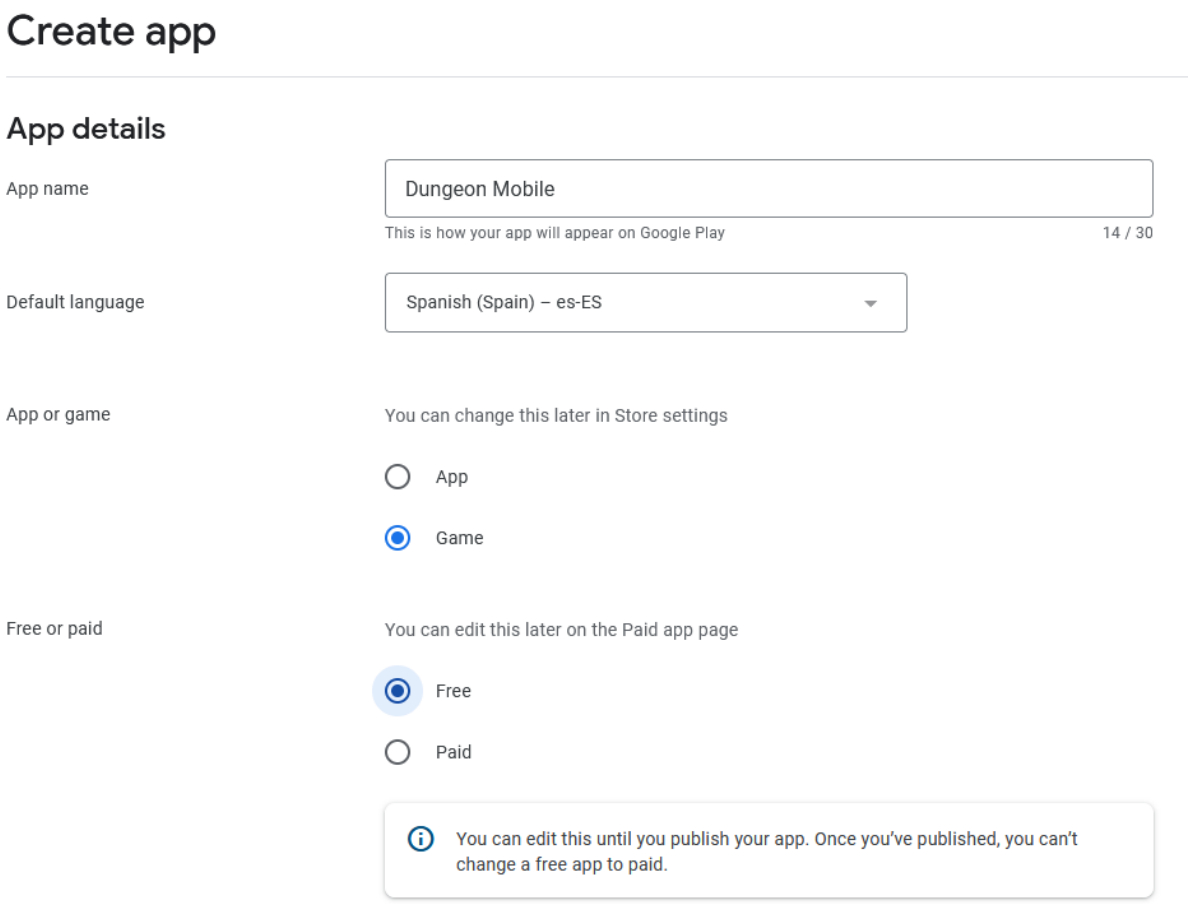


Figura 81: formulari inicial per crear una app a Google Play

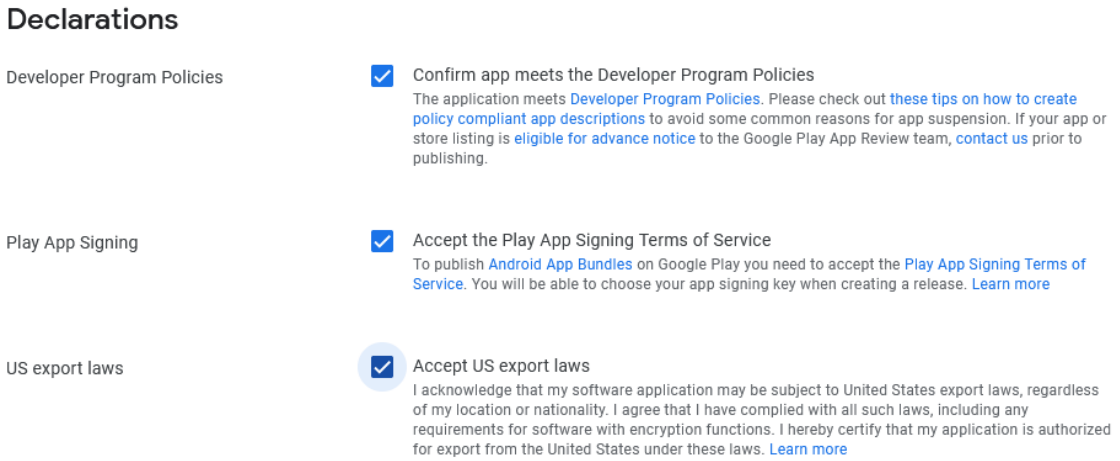


Figura 82: segona part del formulari necessari per pujar un joc a Google Play

- Pas 3

Seguidament s'ha d'emplenar cadascun dels apartats necessaris perquè es pugui identificar quin tipus d'app o joc és, a quin tipus de persones es pot recomanar(PEGI), quins tipus de permisos necessita i a quin tipus de dades pot accedir l'aplicació. També s'ha d'especificar a quines funcions del mòbil accedirà l'aplicació, fotos, camara, arxius específics…

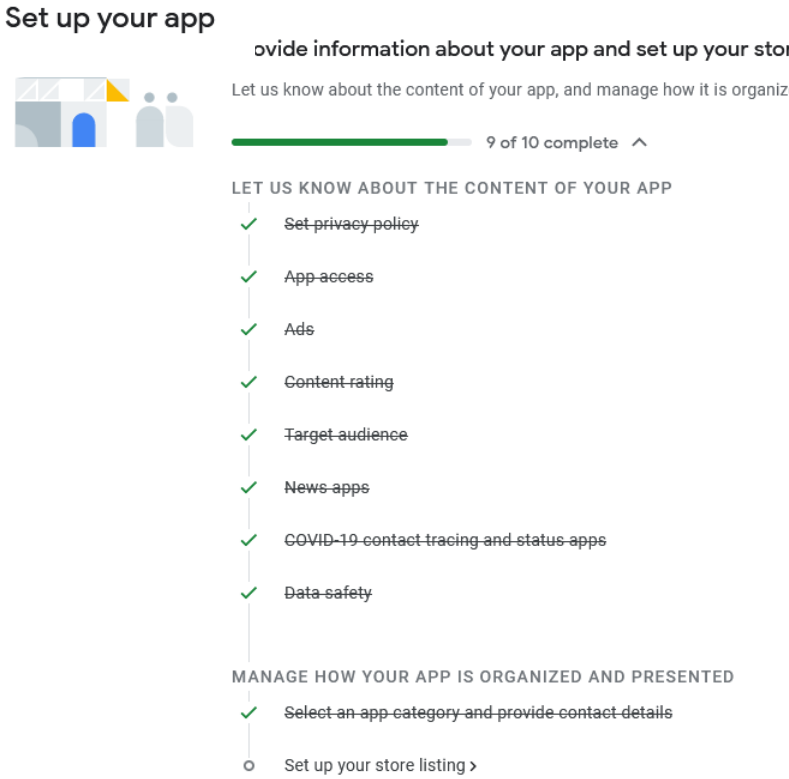


Figura 83: formularis necessari per plenar amb informació secundaria del joc

- Pas 4

Un pas obligatori per poder tenir un joc a la Play Store es pujar el arxiu corresponent a la build del videojoc (arxiu .aab), a més de un nom i/o versió i una descripció explicant quines són les millores implementades. Això es fa cada vegada que es vol pujar l'app o el joc amb actualitzacions.

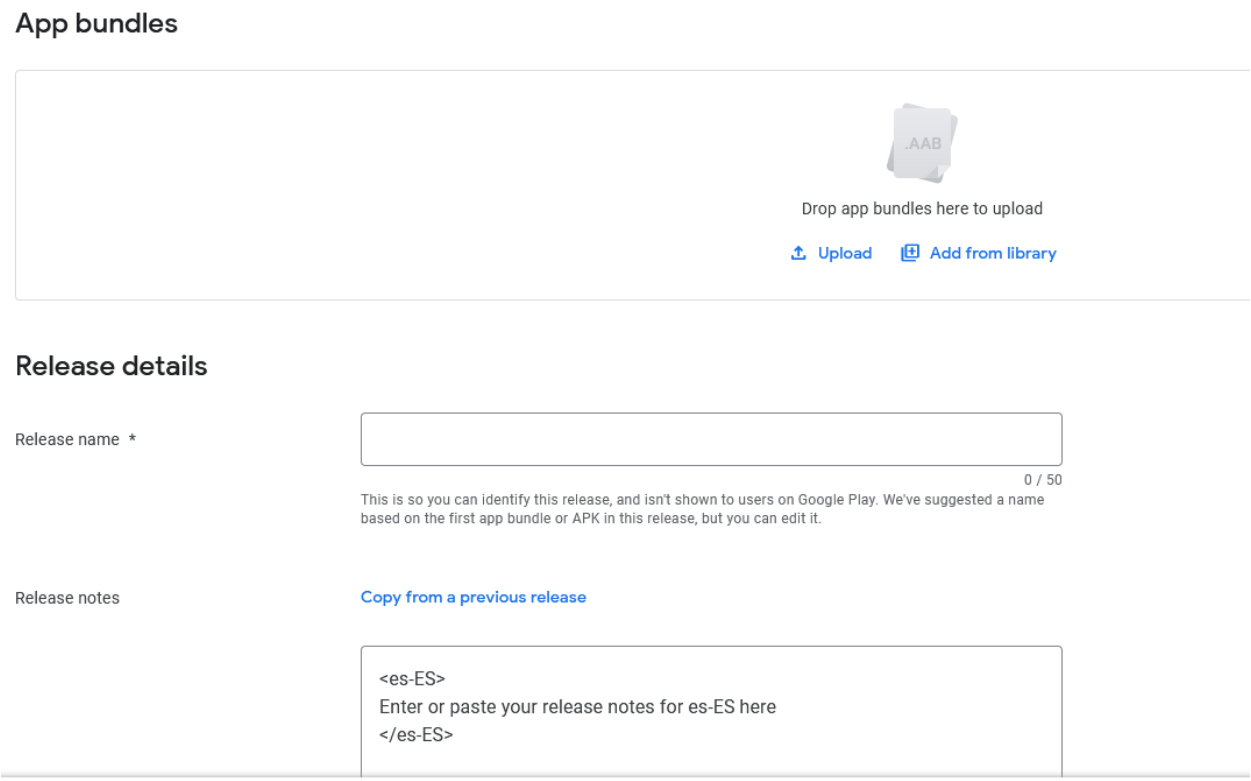


Figura 84: formulari necessari per pujar l'arxiu .aab amb el joc

- Pas 5

Després d'emplenar totes les dades necessàries, pujar els arxius, es publica, pero per aconseguir això serà necessari que ens ho aprovin. A dia d'avui encara esta en revisió per es seguira treballant fins que s'aprovi.

Figura 85: resultat al plenar tots els formularis necessaris i passar l'aplició a finalitzada

6. Resultats

En aquest apartat mirarem els objectius que es van marcar en un principi i llavors farem valoració de si els hem assolit i amb quin grau de compliment.
Es recorda que la finalitat principal era dissenyar un videojoc, amb la condició de simular un desenvolupament professional. L'altra condició més important que marcava el marc del projecte era que havia de ser publicat com a mínim a una plataforma oficial mòbil, Play Store per a Android o App Store per a Apple.

Per a tals qüestions, es va decidir marcar uns objectius específics:

- **Fer un estudi de mercat dels videojocs de mòbil per poder-se adaptar i així tenir l'oportunitat de competir contra els jocs més populars de la plataforma.**
  S'ha dedicat un apartat sencer a desenvolupar l'estudi de mercat per al videojoc, en concret l'apartat 2: Estudi de viabilitat.
  En l'apartat mencionat, es fa un estudi de mercat per saber quins jocs hi ha al mercat que puguin fer competència a Dungeon Mobile, com en comparació amb aquest Dungeon Mobile és millor i que ofereix el joc que no ofereix la competència.
  A partir d'aquí obtenim un públic objectiu a través d'un anàlisis de competència sumat al coneixement adquirit durant la carrera i a l'experiència adquirida a través dels anys jugant diferents estils de joc.
- **Dissenyar i implementar un joc 2D.**
  Durant tots els apartats 4 i 5, Disseny del videojoc i Implementació i proves s'explica el funcionament del videojoc i l'assoliment dels objectius marcats per aquest, es fa un recorregut pels scripts creats per les mecàniques, pel disseny del mapa i dels personatges i la implementació d'aquests.
- **Aplicar una part aleatòria i procedural per permetre que cada partida sigui diferent.**
  Dins l'apartat 4.4 Creació de nivells podem veure com queda cada nivell amb els seus sprites. Podem veure com va evolucionant les sales per cada nivell i com podem distingir-lo. Per l'altra part a l'apartat 5.2.1 Generació de sales i nivells es pot trobar els scripts que expliquen el funcionament i la jerarquia que s'empra per crear cada nivell, les sales que els conformen, amb els diferents objectes, power--ups i enemics, a més s'explica com crear el minimapa amb la forma de les sales corresponent.
- **Dissenyar i implementar la part artística del joc.**
  Al llarg de tot l'apartat 4 es pot veure com s'han creat els personatges, els enemics i les seves animacions corresponents també podem fer un

recorregut pels sprites que formen cada nivell, els sprites dels objectes i power-ups i finalment els menús, amb els seus botons i característiques.

- **Publicar el joc en una Store oficial.**
  Com a objectiu final després d'haver creat el joc, és publicar-lo. Per aquest motiu com a finalitat per al joc Dungeon Mobile era com a mínim publicar-lo a una botiga oficial. S'ha decidit publicar-lo a la Play Store d'Android per què és la més econòmica. Per publicar-lo s'ha fet una guia a l'apartat 5.4 on espot veure els passos necessaris per pujar una app o joc a la Play Store.

Com a resultat final podem veure el nostre joc ja pujat a la Play Store (figura 86):

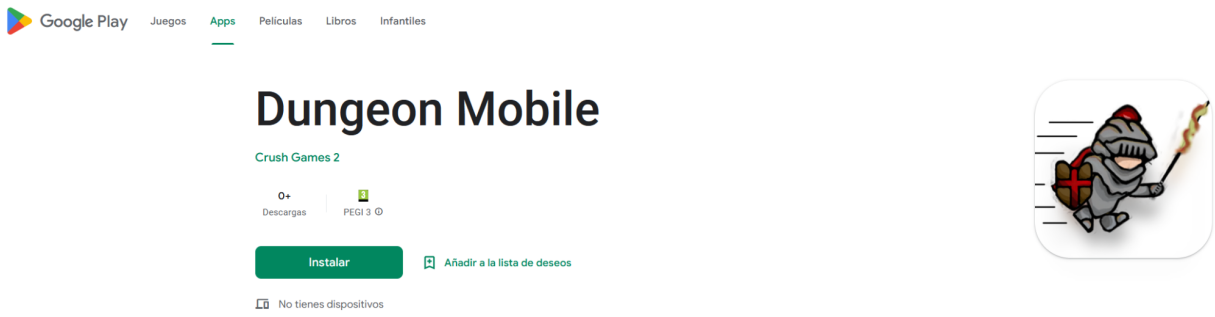


Figura 86: imatge del joc ja disponible a la Play Store

Seguidament podem veure unes captures fetes ja en un dispositiu android de com es veu Dungeon Mobile.

Figura 87: Captura del menú inicial del joc en un dispositiu mòbil

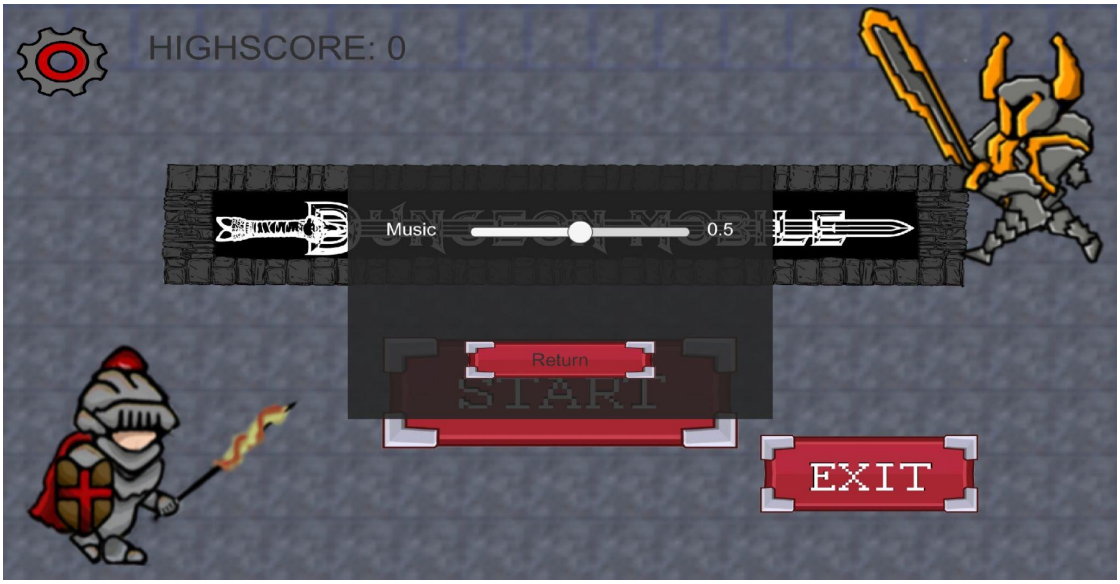


Figura 88: Captura del menú pausa i menú de configuració del joc en un dispositiu mòbil

Figura 89: Captura del menú inicial del joc en un dispositiu mòbil

7. Conclusions

Al llarg d'aquests mesos he pogut treballar en tots, o gairebé tots els passos necessaris dins la producció d'un videojoc. He aplicat tot l'après durant la carrera i sobretot m'he enfrontat a reptes i problemes que no han sorgit durant els estudis.

Primer de tot es pot dir que fer un videojoc una sola persona mentre s'estudia i es treballa és molt complicat perquè si han de dedicar moltes hores, que hagués volgut tenir, per poder polir i arreglar moltíssims detalls, tot i això, estic bastant orgullós del resultat al qual ha arribat el projecte.

"Dungeon Mobile" era la idea d'un videojoc que feia temps que volia dur a terme, Segurament hauria acabat portant a terme en algun moment de la meva vida, però gràcies al projecte he pogut aprofitar-ho per a realitzar aquest treball també i que segurament actualitzaré amb el temps per millorar-lo el màxim possible

Fer el document informatiu també s'ha endut una gran part del temps, tot i això, el document ha servit com a guia i per tenir una estructura de tot el que estava implementat en cada moment, a més de fer de guia sobretot dels scripts per l'estructura i recordar de quina forma es criden aquests.

Fent un repàs podem veure quines eren les tasques marcades en un inici i podem veure quines d'aquest s'han pogut implementar finalment. Les tasques marcades amb un tic verd “✅” s'han pogut dur a terme, en canvi, les marcades amb una creu vermella “❌” no:

- ✅ **Personatge Principal**
  - ✅ **Disseny del personatge**: pensar una idea de personatge adaptat a una història i un temps específic.
  - ✅ **Dibuixar**: dibuixar el personatge enfocat a poder animar-lo posteriorment amb facilitat.
  - ✅ **Animar:** fer les animacions de moure’s, atacar, morir i esperar del personatge principal.

- ✅ **Enemic**
  - ✅ **Disseny dels Enemics i Boos:** crear enemics que tinguin concordança en espai i temps amb la història i els altres personatges.
  - ✅ **Dibuixar:** dibuixar els enemics enfocats a poder animar-lo posteriorment amb facilitat.

    - ✅ **Animar:** fer les animacions de moure's, atacar, morir i esperar.

- ✅ **Dibuixar textures mapa primer nivell**: Dibuixar textures amb temàtica de "dungeon" per parets, terra i obstacles.
- ✅ **Dibuixar Textura mapa segon nivell**: Dibuixar textures amb temàtica de desert per parets, terra i obstacles.
- ✅ **Dibuixar Textura mapa tercer nivell**: Dibuixar textures amb temàtica de castell per parets, terra i obstacles.
- ✅ **Dibuixar Textura mapa quart nivell**: Dibuixar textures amb temàtica de muntanya per parets, terra i obstacles.
- ✅ **Dibuixar Textura mapa cinquè nivell**: Dibuixar textures amb temàtica de casa per parets, terra i obstacles.
- ✅**Personatge Principal**
    - ✅ **Programar el moviment:** fer que el personatge es mogui en els eixos X i Y.
    - ✅ **Fer i rebre mal dels enemics:** crear un sistema de vida que es pugui veure quan reps mal i fas mal a enemics.
    - ✅ **Disparar:** programar que el jugador pugui apuntar i disparar cada x temps.
    - ✅ **Interactuar amb les portes:** poder entrar a les portes i que et portin a la sala adient.
    - ✅ **Agafar objectes:** en passar per sobre d'objectes com claus, poder agafar-los.

- ✅ **Enemics**
    - ✅ **Patrullar per la sala:** crear circuits aleatoris pels quals els enemics es moguin per la sala.
    - ✅ **Buscar al personatge principal:** buscar al jugador perquè en el moment en el qual es tingui contacte visual ataquin.
    - ✅ **Seguir al personatge principal:** Seguir al personatge per poder atacar-lo.
    - ✅ **Atacar:** si el jugador està el prou a prop atacar i poder impactar o fallar depenent del moviment del personatge principal.

- **Creació sales procedural**
    - ✅ **Instanciar enemics a les sales:** crear enemics a les sales i que no es creïn sobre objectes o posicions inadequades.

    - ✅ **Instanciar un Boos per Nivell:** Instanciar un sol Boos per nivell i a sales finals.
    - ✅ **Instanciar una clau per poder passar de nivell:** instanciar una clau per nivell que permeti obrir la porta final.
    - ✅ **Instanciar de forma aleatòria** objectes i obstruccions per les diferents sales.
    - ✅ **Instanciar el personatge**: instanciar el jugador en una sala central cada vegada que es passa de nivell.
    - ✅ **Instanciar Power-ups:** instanciar amb un percentatge de probabilitat diferent power-ups que puguin ajudar al jugador.
    - ❌ **Definir i implementar un sistema de puntuació** que permeti encoratjar al jugador a superar-se cada vegada que juguí.(S'ha definit un sistema de puntuació i s'ha implementat pero no funciona correctament i s'ha decidit treure'l fins que tingui un funcionament correcte).

- **MiniMapa**
    - ✅ **Crear instàncies de les sales** a la part superior simulant un minimapa
    - ✅ **Interacció del minimapa amb el personatge** i canviar de color segons la sala en què estigui.

- **Puntuació:**
    - ❌ **Punts per mort,** aquests augmenten cada vegada que se superi un nivell.

- **Menús**
    - ✅ **Crear un menú inicial** amb un botó de configuració un botó "Start" i un botó "exit".
    - ✅ **Crear un menú de pausa** i configuració on es pugui modificar el so.
    - ✅ **HUD** mentre es juga on es pugui veure la vida del jugador, la posició dels joysticks, i un minimapa per saber a quina sala estàs.

- **Música i sons**
    - ✅ **Implementar** diferent **música** pel menú i pel joc.
    - ❌ **Implementar sons de trets i impactes** (no s'han pogut implementar sons als enemic ni als dispars per falta de temps)

- **Memoria**
    - ✅ **Crear la memoria** del projecte

Seguidament, podem veure com ha sigut el resultat final del diagrama de gannt. Podem observar com els dies d'inici s'han respectat pero hi han tasques que s'ha hagut més del temps del estimat, ja sigui per sobrecàrrega de feina o per compaginar amb factors extern però també podem veure que hi han tasques que s'esperaba tardar més i s'han pogut acabar dins del termini. Hi han tasques que no s'han pogut implementar i no surten en aquest graf de Gannt.

	Nombre de Actividad	Fecha Inicio	Duración en dias	Fecha fin
Art Personatge Principal (PP)		15/07/2021	12	17-jul
	Disseny del Personatge Principal	17/07/2021	15	01/08/2021
	Dibuixar PP	22/07/2021	15	06/08/2021
	Animar PP	27/07/2021	15	11/08/2021
Art Enemic				
	Disseny dels Enemics i Boos	01/07/2021	12	13/07/2021
	Dibuixar Enemics	03/07/2021	15	18/07/2021
	Animar Enemics	07/07/2021	15	22/07/2021
	Dibuixar textures mapa primer nivell	12/02/2022	14	26/02/2022
	Dibuixar Textura mapa segon nivell	22/02/2022	13	07/03/2022
	Dibuixar Textura mapa tercer nivell	01/02/2022	13	14/02/2022
	Dibuixar Textura mapa quart nivell	13/02/2022	13	26/02/2022
	Dibuixar Textura mapa cinquè nivell	22/02/2022	13	07/03/2022
Personatge Principal				
	Programar el moviment del PP	01/09/2021	22	23/09/2021
	Fer i rebre mal dels enemics	01/10/2021	20	21/10/2021
	Disparar	15/09/2021	13	28/09/2021
	Interactuar amb les portes	12/10/2021	12	24/10/2021
	Agafar objectes			
Enemics				
	Patrullar per la sala	01/06/2022	15	16/06/2022
	Buscar al personatge principal	03/04/2022	14	17/04/2022
	Seguir al personatge principal:	15/04/2022	14	29/04/2022
	Atacar	10/10/2021	12	22/10/2021
Creació sales procedural				
	Instanciar enemics a les sales	15/06/2022	17	02/07/2022
	Instanciar un Boos per Nivell	01/07/2022	12	13/07/2022
	Instanciar una clau per poder passar de nivell	01/08/2022	12	13/08/2022
	Instanciar de forma aleatòria	22/06/2022	16	08/07/2022
	Instanciar el personatge	01/10/2021	13	14/10/2021
	Instanciar Power-ups	15/11/2021	15	30/11/2021
	Definir i implementar un sistema de puntuaci	01/12/2021	11	12/12/2021
MiniMapa				
	Crear instàncies de les sales	01/03/2022	26	27/03/2022
	Interacció del minimapa amb el personatge	15/12/2021	22	06/01/2022
Puntuació				
	Punts per mort			
Menús				
	Crear un menú inicial	15/01/2022	12	27/01/2022
	Crear un menú de pausa	20/05/2022	11	31/05/2022
	HUD	01/11/2021	14	15/11/2021
Música i sons				
	Implementar música	15/07/2021	12	27/07/2021
	Implementar sons de trets i impactes			
Memoria				
	Crear la memoria	01/01/2022	246	04/09/2022

Figura 90: taula dels dies empleats per cada tasca finalment

Figura 91: gràfic de Grannt final amb el temps empleat finalment

S'han tingut problemes sobretot a l'hora de pujar el joc a Google play, ja que a partir de l'agost del 2022 és necessari que el joc o l'aplicació estigui disponible com a mínim per l'api 31 i en ser una API tan nova no hi ha suficient informació per poder compilar el joc correctament. També han sorgit dificultats que poden sorgir en qualsevol projecte com poden ser conflictes a l'hora d'afegir diferents funcionalitats com per exemple el minimapa i les portes, conflictes a l'hora d'instanciar objectes, enemics power-ups de forma aleatòria i que no se sobreposin,… Tot i això, la majoria s'han pogut resoldre satisfactòriament.

Per concloure, crec que s'ha complert amb els objectius del treball. En aquest treball es volia publicar en una plataforma oficial el videojoc. En aquest cas per tema de recursos només s'ha decidit publicar-lo a la Play Store d'Android.
El disseny del joc conté tot el que en un principi s'ha pensat, el joc té les parts per mi, necessàries, adequades i amb les quals es pot entendre la idea del joc. Es pot jugar una partida sencera i es pot veure que el focus del joc està en el fet que és un joc infinit de creació aleatòria i procedural, apte per a mòbils i de partides curtes.

8. Treball futur

Des de fa uns anys quasi tots els videojocs acostumen a sortir al mercat amb el joc base i poques "skins". A mesura que va passant el temps van sortint més modes de joc i més "skins" per mantenir als jugadors enganxats. Un gran exemple és el Fortnite que després de 5 anys continua triomfant. Es vol aprofitar aquesta metodologia per poder afegir funcionalitats i millores.
Com a treball futur m'agradaria poder assolir els següents punts per fer que el joc estigui complet i com desitjaria que hagués quedat:

- Polir l'art:
    - Millorar les animacions dels personatges: Les animacions avui en dia no estan completament polides i de vegades es veuen de forma extranya. S'hauria de refer algunes o modificar les ja fetes, perquè no es vegi el tall entre animacions o que es vegin de forma més lineal amb el moviment.
    - Afegir varietat d'enemics i atacs: Ara mateix només hi ha dos tipus d'enemic i un atac. M'agradaria implementar més enemics amb diferents atacs pels diferents nivells i diferents skins per cada un dels enemics.
    - Afegir més varietat de nivells. Ara mateix ja hi ha 5 diferents nivells, però poder estaria bé fer un conjunt de sprites i que per cada nivell s'escollissin diferents sprites, així hi hauria més varietat.
    - Afegir skins per poder monetitzar el videojoc. No hi cap manera en la qual monetitzar el joc. S'han pensat dues maneres de poder monetitzar-lo, en primer lloc, cada vegada que es mori el jugador, si es vol una vida extra es mostra un anunci i revius amb la vida al 50%. La segona manera de monetitzar-lo seria implementant skins, les quals es podran adquirir pagant-les directament o de forma aleatòria en una ruleta. Això comportaria modificar el PEGI del joc.

- Mecàniques:
    - Afegir diferents atacs pel personatge principal o més d'un personatge jugable: La idea seria tenir més d'un personatge, un amb més vida, un altre amb més velocitat, inclús un amb dany a melé i un altre amb dany a distància i poder agafar com a base un d'aquest i després a mesura que matis enemics, a final de cada partida rebre diferents punts que es poguessin intercanviar per modificar habilitats o a atacs o fins i tot per poder comprar de nous.

- Afegir diferents modes de joc: La idea seria poder implementar algun mode de joc cooperatiu i/o competitiu que permetés jugar amb els teus amics online.

9. Bibliografia

-Dades sobre els videojocs
https://es.statista.com/estadisticas/879172/dispositivos-usados-por-la-poblacion-para-jugar-a-videojuegos-por-edad-espana/
-Descripcio joc Caves, mencionat a l'estat de l'art
https://vandal.elespanol.com/juegos/pc/caves/51938#p-13
-Descripcio del joc Pathos, mencionat a l'estat de l'art
https://play.google.com/store/apps/details?id=com.x10host.pathos&hl=es&gl=US
-Definició target
https://mglobalmarketing.es/blog/marketing-para-empresas-como-se-define-el-target/
-Historia dels vieojocs 1
https://3cero.com/historia-juegos-moviles/
-Historia dels videojocs 2
https://www.hobbyconsolas.com/reportajes/juegos-moviles-mejor-modo-historia-pasar-entretenido-cientos-horas-902413
-fix Target API 31
https://forum.unity.com/threads/android-targeting-api-level-31-makes-the-game-freeze-on-android-12.1237576/
-Procedural Generation Map
https://www.youtube.com/watch?v=nADIYwgKHv4&list=PL5hTSKx61-Etf8W6GeJAyrqQVKnOpJaZT

10. Manual d'usuari i d'instal·lació

Per instal·lar el joc és necessari tenir un dispositiu Android, ja sigui un móvil, una tablet o qualsevol dispositiu amb Android superior al 8.0 Oreo. També és necessari que el dispositiu tingui espai de memoria suficient per poder instal·lar-lo.
Seguidament entrarem a l'aplicació "Play Store" que ve ja instalada amb el mòbil i buscarem al buscador superior, Dungeon Mobile i cliquem sobre la imatge de la figura 92 i cliquem al boto "instalar". Una vegada instalat ja podrem obrir-lo i començar a jugar.

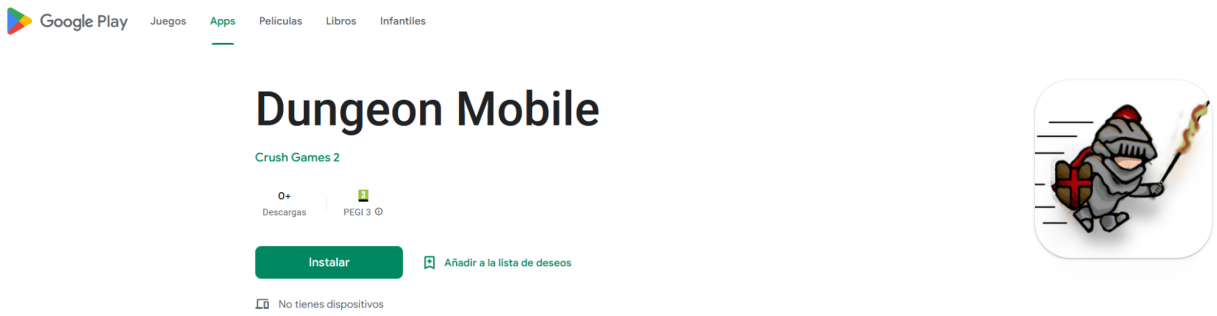


Figura 92: imatge del joc a la Play Store