Universitat de Girona
Escola Politècnica Superior

Treball final de grau

Estudi: Grau en Disseny i Desenvolupament de Videojocs

Títol: Disseny i implementació d'una plataforma escalable aplicada a un videojoc MOBA

Document: Memòria

Alumne: Aleix Ferré Juan

Tutor: Gustavo Patow
Departament: Informàtica, Matemàtica Aplicada i Estadística
Àrea: Llenguatges i sistemes informàtics

Convocatòria (mes/any): Febrer 2022

Agraïments

A Gustavo Patow, tutor del projecte, pels ànims i optimisme durant tot el procés de desenvolupament d'aquest projecte.

A la meva família per donar-me suport i creure en mi des del primer dia.

Als companys que han dedicat el seu temps en trobar errors el dia del torneig.

Índex

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

1.1 Introducció

Avui en dia el món dels videojocs està evolucionant molt ràpidament i de manera accelerada, tant en l'Apartat gràfic com en nous generes i mecàniques revolucionaries. Un dels gèneres que més m'ha impactat és el MOBA[1], des de la seva aparició a l'any 2003 amb Defense of the Ancients (DOTA) fins a l'actualitat amb el llançament de League of Legends: Wild Rift al 2020. Afegint-s'hi totes les competicions d'esports electrònics i els milions de jugadors diaris, aquests jocs estan sumant moltíssima popularitat. Però, per quin motiu són tan populars aquests jocs? La re-jugabilitat és un factor clau entre els jocs d'aquest gènere. El fet de que hi hagi un mapa amb milers de possibilitats i mecàniques que combinar, sumat amb el sistema multijugador en línia afavoreix a que cada partida sigui única i especial entre els jugadors. Al ser una partida 5 contra 5, la combinació de competitivitat amb la cooperació sempre dona lloc a interaccions molt complexes i interessants. Per exemple, un aliat pot moure un enemic cap a la seva posició amb un ganxo i un altre l'aliat li faci mal cos a cos. En el cas de que no hi hagi un aliat que el mogués, el personatge cos a cos no podria fer res. I això és complica més i més quan hi afegim les mecàniques competitives, fent possible que l'equip enemic, també format per 5 integrants, pugui fer mecàniques similars amb els mateixos o diferents tipus de personatges. En aquestes baralles entre equips o "*team fights*" es concentren la major quantitat d'interaccions entre personatges i d'emocions entre jugadors.

Hi ha hagut moltes entregues de videojocs MOBA, especialment en aquesta ultima dècada, moltes d'elles essent una revolució en el que presenten. Per exemple, Arena of Valor o Vainglory van ser els primers en portar el gènere a dispositius altres del clàssic PC. Aquesta aposta reduïa el nombre d'interaccions possibles i simplificava el joc de manera que els jugadors més casuals poguessin jugar en qualsevol lloc al seu gènere de videojocs preferit.

Per això, s'ha proposat per aquest projecte aconseguir fer una base sòlida i escalable per a un videojoc MOBA amb un sistema d'habilitats, personatges, ítems i totes les mecàniques que proporciona el mapa.

[1] MOBA: Acrònim de "Multiplayer Online Battle Arena", traduït al català seria "Arena de Batalla Multijugador En Linia". Aquest gènere s'explica en profunditat a l'Apartat 4.1.

1.2 Objectius

1.2.1 Estudiar el comportament intern d'un joc MOBA

Per poder implementar un videojoc d'aquest estil primer cal saber quines són totes les mecàniques i com interaccionen tots els elements que el composen.

1.2.2 Estudiar els sistemes multijugador online

La lletra O de les sigles MOBA significa Online, pel que és necessari fer un sistema en xarxa per tal de que el joc tingui sentit i sigui jugable. Cada partida cal estar sincronitzada amb tots els clients amb una latència relativament petita, pel que fa que el joc hagi d'estar optimitzat al màxim possible.

1.2.3 Crear un sistema complex que interconnecti totes les mecàniques

Els videojocs MOBA barregen les entitats controlades per la IA i els combats entre jugadors reals en xarxa. Cal implementar un sistema genèric que entengui cada entitat i quines són les diverses interaccions que l'afecten.

1.3 Quadre d'autovaloració

Estètica	15%
Narrativa	5%
Mecàniques	25%
Tecnologia	55%

Taula 1. Quadre d'autovaloració

A la taula anterior es pot observar que l'objectiu principal d'aquest projecte, al mateix temps de crear un videojoc complet, és crear una plataforma[2] robusta i escalable per anar afegint mecàniques de forma àgil i ràpida mentre el joc va fent-se cada cop més gran.

Al mateix temps, s'ha decidit per una estètica visualment atractiva per tota mena de públic i, sobre tot, que sembli el més professional possible. A l'hora d'escollir recursos de models 3D per l'escena (personatges o objectes del mapa), aquests estan trets de fonts obertes d'internet. Totes les interfícies, tant menús com HUD dins del joc, estan fetes a ma expressament pel projecte.

La narrativa que aquest projecte requereix no és molt complexa. Els personatges es situen en un món distòpic on lluitaran uns contra els altres en una lluita per les bases. Degut al fet de fer servir recursos externs, aquestes històries queden bastant limitades a com siguin aquests models ja donats. Per tant, en aquest

[2] Plataforma: Base robusta i extensible de codi on es basa tot el joc. Aquesta base serà la que sustentarà tots els elements del joc com entitats, ítems, habilitats, etc. Per estendre algun comportament només caldrà crear una nova classe que hereti d'algun comportament similar i/o simplement crear un nou *Scriptable Object* que guardi la informació al disc de forma persistent.

aspecte el prototip té la valoració mínima. Encara que el prototip no es basi en la narrativa, es podria estendre molt fàcilment en qualsevol dels seus àmbits, tant en narrativa del mapa com en història de cadascun dels personatges i com s'interrelacionen.

Les mecàniques en aquest projecte són una part essencial. Aquestes formen la partida i la forma de jugar. Són les regles i fonaments del joc. Les regles d'un MOBA ja venen donades per anteriors entregues d'aquest gènere, però també he volgut donar-li un tomb a aquestes i refer o modificar algunes perquè la experiència del jugador sigui nova i refrescant cada cop que es juga.

Finalment, la tecnologia forma part dels fonaments d'aquest prototip. Sense una bona base, la plataforma perd robustesa al fer-se gran i comencen a aparèixer errors i comportaments inesperats a l'hora de jugar. Això inclou el sistema multijugador en línia, el sistema de personatges modulars, la interacció entre les habilitats dels personatges i les altres entitats del món, les IA de les torretes[3] i dels *minions*[4], l'enviament de missatges entre clients, ítems que canvien les estadístiques del jugador, sistema d'actives i passives, etc.

[3] Torreta: Fortificació que protegeix la base i que bloqueja el traspàs enemic cap a la banda contraria del mapa. Aquesta entitat no es mou i ataca només a una entitat al mateix temps. Totes les torretes excepte el làser que protegeix la base, poden ser destruïdes perquè deixin de tenir efecte.

[4] Minion: Entitat aliada o enemiga que es mou des de la base corresponent fins al camp de batalla amb el propòsit d'atacar a les unitats enemigues que estiguin presents en el carril que li ha estat assignat.

2 Estudi de viabilitat

Tots els projectes de videojocs requereixen de molts recursos per funcionar i, per tant, una inversió inicial per finançar-lo. Si el videojoc és un producte que ha de sortir al mercat, cal que almenys es tingui en compte la programació, l'estètica, la narrativa i el disseny del joc. Però també cal sumar-li el màrqueting necessari i la publicitat per treure el joc al mercat. A *Steam* es publiquen més de vint jocs cada dia. Hem de fer que el nostre videojoc llueixi per damunt de la resta.

Per això cal una bona inversió inicial en tots els camps principals al fer un estudi de viabilitat, mirant detalladament tot el que és necessari per desenvolupar aquest projecte. Estem parlant de tot el cost de personal, software, hardware i eines que s'utilitzaran en el desenvolupament.

2.1 Software

A continuació s'explicaran la majoria de programes que s'han fet servir pel desenvolupament d'aquest projecte.

2.1.1 Unity

Unity és un dels motors de videojocs més famosos. És el programa principal en el que es sustenta aquest projecte. *Unity* porta en funcionament des del 2005 i ha servit a molts desenvolupadors per fer videojocs tan potents com *Hearthstone*, *Mini Metro* o *Monument Valley*. Fa servir C# com a llenguatge de programació principal i és molt fàcil de començar a fer servir amb multitud de tutorials a YouTube de part del canal oficial de Unity o altres creadors. Aquest programari és gratuït amb la llicència personal fins a 100.000$ anuals.

Figura 1: Logotip de Unity

2.1.2 Visual Studio Code

Visual Studio Code és l'editor de codi escollit per treballar en aquest projecte. És lleuger, simple i extensible per defecte. Hi ha un munt d'extensions que es poden fer servir perquè aquest editor es converteixi fàcilment en un IDE[5] funcional amb

[5] IDE: Entorn integrat de desenvolupament. Programa que agrupa totes les funcionalitats necessàries per desenvolupar software en un entorn de treball específic. Normalment, a part de ser editors de text també tenen integrat el compilador i recomanacions de codi, a més de dreceres que fan el desenvolupament més senzill i àgil.

el *framework*[6] de Unity. Per defecte *Unity* fa servir la versió *Visual Studio Community* la qual és un IDE en si mateix per poder treballar amb projectes de C#, però el programa és molt pesat i tarda molt en carregar, pel que s'ha optat per la versió lleugera i portable del programa.

Figura 2: Logotip de Visual Studio Code

2.1.3 Blender

Blender és la eina que s'ha fet servir per generar tots els models 3D que no s'han pogut extreure de fonts externes a causa dels requeriments específics del joc. Aquest programa ens permet modelar, esculpir, generar esquelets, animar, renderitzar tot tipus de models en 3D.

Blender té una llicència de codi obert i es gratuït per a tothom, el qual el fa un molt bon candidat per generar els models necessaris pel desenvolupament del joc de forma senzilla i econòmica.

Figura 3: Logotip de Blender

2.1.4 Adobe Photoshop

Adobe Photoshop és l'editor d'imatges que s'ha triat per aquest projecte. Aquest programari s'ha fet servir per generar tots els dissenys de les interfícies del joc. Aquest permet, de forma intuïtiva i ràpida, editar fotografies o exportar el fitxer en format d'imatge resultant.

Figura 4: Logotip de Adobe Photosop

[6] Framework: Marc de treball. Unity té una forma de treballar que s'ha de seguir per poder treballar amb el motor correctament. Per exemple: els Game Objects, els Components, les funcions Start i Update són una part essencial de Unity que cal aprendre i seguir, tal i com diu la documentació oficial.

2.1.5 Microsoft Word

Microsoft Word és l'editor de text de pagament de Microsoft. Aquest s'ha fet servir per generar tota la documentació del projecte. Aquest permet, de forma molt fàcil i ràpida, editar documents de text amb imatges i generar un fitxer PDF resultant.

Figura 5: Logotip de Microsoft Word

2.2 Recursos

2.2.1 *Hardware* necessari

Per desenvolupar aquest projecte cal un màquina suficientment potent com per executar un videojoc amb gràfics 3D. Part del desenvolupament del joc s'ha fet en un portàtil d'unes característiques mitjanes.

- **Processador**: Intel i7-6700HQ
- **Targeta gràfica**: Gràfics integrats
- **RAM**: 8GB
- **Disc dur**: 1TB HDD

Cap al final del desenvolupament del projecte, es va invertir en una torre molt més potent de les següents característiques:

- **Processador**: Intel i7-11700K 3.60GHz
- **Targeta gràfica**: NVIDIA GeForce RTX 3070
- **RAM**: 16GB DDR4 3600MHz 2x8GB
- **Disc dur**: 500 GB SSD + 3TB HDD

En tots dos casos s'ha utilitzat Windows 10 com a sistema operatiu, encara que al nou ordinador s'hagi pogut actualitzar a Windows 11.

2.2.2 Recursos humans

Els següents perfils de treballadors seran necessaris per completar el projecte. Tots aquests rols s'hauran d'executar per una sola persona alhora, obtenint el rol que s'estigui necessitant en cada moment. La informació següent tracta sous mitjans segons fonts fiables[7]:

[7] Totes les dades de sous estan extretes de la pàgina https://www.jobted.es/. Aquesta et dona un sou anual brut mitjà, o bé nets per mes. Ens centrarem en els nets per més per calcular el preu per hora. Per

- **Programador**: encarregat de mantenir una estructura de codi i fer que el videojoc tingui les funcionalitats requerides.
    - 11 €/h nets
- **Dissenyador**: encarregat del disseny de les mecàniques del mapa, dels personatges i dels ítems. També s'encarregarà del correcte balanceig.
    - 15 €/h
- **Artista 3D**: encarregat del modelat i animació de tots els elements 3D del joc.
    - 14 €/h
- **Artista UI/UX**: Encarregat de fer les interfícies del joc. Això engloba els menús, les animacions que tingui, interfícies[8] diegètiques, espacials, meta, etc.
    - 13 €/h
- **Artista de so**: Encarregat de crear tots els sons (efectes de so) i la música.
    - 12 €/h
- ***Game Tester***: Encarregat de provar el joc a fons per trobar errors en la implementació o formes no esperades de poder passar-se el joc.
    - 8 €/h

El prototip ha ocupat unes 500 hores de feina en total, però podem assumir que, com és un projecte educatiu, el cost humà és zero.

fer-ho multiplicarem 42h setmanals * 4 = 168h/mes. Amb aquesta quantitat podem dividir-la per obtenir el preu per hora de cada lloc de treball.

[8] Existeixen molts tipus d'interfícies: diegètiques, no diegètiques, espacials i meta. Aquestes es complementen entre si per crear una experiència al jugador alhora que l'informen del que està passant a la partida. Per més informació, veure https://www.gamedeveloper.com/design/user-interface-design-in-video-games

2.3 Viabilitat

2.3.1 Cost de software

Tots els programes utilitzats per aquest projecte són, o bé gratuïts, o bé se'n té una llicència gratuïta, per tant el cost total del projecte en aquest àmbit és zero.

Nom	Cost
Unity	0 €, dins de la llicència personal
Visual Studio Code	0 €, a més és open source
Blender	0 €, a més és open source
Photoshop	0 €, ja es tenia llicència abans
Word	0 €, amb la llicència d'estudiant
Total	**0 €**

2.3.2 Cost de recursos

El cost del projecte en l'Apartat de recursos continua sent zero, ja que el cost del hardware que s'ha vist a l'Apartat 2.1.1 s'ha assumit com a cost personal i no per aquest projecte en específic.

Nom	Cost
Ordinador portàtil	800 €, el cost real assumit per aquest projecte és 0 €
Ordinador sobretaula	2000 €, el cost real assumit per aquest projecte és 0 €
Recursos humans	500h * 13€/h[9] = 6500 € (0 € entenent que es treballa de forma gratuïta pel projecte)
Total	**0 €**

[9] Els 13 €/h s'ha extret de fer la mitjana de preus a l'hora dels càrrecs vistos a l'Apartat 2.2.2.

2.4 Estudi de mercat

Abans de començar a desenvolupar el videojoc o, en aquest cas, el prototipus, cal saber l'estat del mercat en aquest àmbit. Cal tenir en compte altres videojocs del mateix estil i veure les diferències i característiques. Finalment cal, en funció d'aquests paràmetres, valorar el projecte i enfocar-lo de la manera més atractiva possible.

2.4.1 Anàlisi de la competència

Cal buscar entre la competència del mercat, basar-se en idees d'altres jocs i muntar-ne un d'interessant i distingible d'entre la resta. Ara es mostraran alguns exemples de videojocs semblants a aquest projecte.

2.4.1.1 League of Legends

Quan pensem en un MOBA, el primer que se'ns ve al cap a la majoria de jugadors assidus d'aquest gènere és el *League of Legends* (col·loquialment conegut com "*LOL*"). Aquest videojoc desenvolupat per *Riot Games* va ser el primer joc d'aquest gènere que va ser mundialment conegut amb més de 180.000.000 de jugadors actius mensuals[10].

Figura 6: Logotip de League of Legends

2.4.1.2 DOTA 2

Un dels jocs pioners a la escena MOBA és, sens dubte el DotA (mencionat a la introducció). El seu successor l'any 2013 és el famós DOTA 2. Aquest joc, desenvolupat per *Valve*, és molt més seriós i difícil d'aprendre que el League of Legends, però a canvi el sentiment al jugar és molt més satisfactori.

Figura 7: Logotip de DOTA 2

[10]Xifres extretes de https://twitter.com/riotgames/status/1455172784938651649?s=20

2.4.1.3 Smite

Smite és un MOBA amb una perspectiva en tercera persona i temàtica grega amb personatges originals i molt divertit. Una bona aposta si es vol passar-ho bé amb els amics.

Figura 8: Logotip de Smite

2.4.1.4 Vainglory

La primera aposta d'una empresa per fer un videojoc MOBA per tauletes va ser *Vainglory*. Desenvolupat per *Super Evil Megacorp*, aquest videojoc va suposar el canvi de mentalitat per molts jugadors respecte que aquest gènere era exclusiu de PC.

Figura 9: Logotip de Vainglory

2.4.1.5 Heroes of the Storm

Una idea de *Blizzard*, enfocat per un públic més casual i simplificant mecàniques establertes pels anteriors títols, *Heroes of the Storm* és un bon iniciador al món dels MOBA.

Figura 10: Logotip de Heroes of the Storm

2.4.1.6 Pokémon Unite

Pokémon revoluciona el mercat amb un nou joc basat en dominar bases marcant punts, en lloc del clàssic mapa on cada equip va prenent dominància de l'altre destruint les torretes fins la base. El videojoc de la ma de *TiMi Studio Group* i *The Pokémon Company (Nintendo)* és bastant semblant a la antiga idea del mode de joc *Dominion*[11] del *League of Legends* de fa uns anys. També és la primera entrega que veiem que els propis personatges dins de la partida evolucionen no només pujant de nivell, sinó que obtenen un kit d'habilitats diferents quan es transformen.

Figura 11: Logotip de Pokémon Unite

2.4.2 Taula comparativa

A partir dels anteriors exemples, podem treure una taula que compari tots els aspectes més importants.

Nom	Càmera	Dificultat	Dispositiu	Mecànica Principal
League of Legends	Isomètrica	Moderada	PC	Carrera de Bases
Dota 2	Isomètrica	Difícil	PC	Carrera de Bases
Smite	3ra Persona	Moderada	PC / Consola	Carrera de Bases
Vainglory	Isomètrica	Moderada	Tablet / Mòbil	Carrera de Bases
Heroes of the Storm	Isomètrica	Fàcil	PC	Carrera de Bases
Pokémon Unite	Isomètrica	Fàcil	*Nintendo Switch* / Mòbil / Tablet	Domini d'objectius
Masters Arena	**Isomètrica**	**Fàcil / Moderada**	**PC**	**Carrera de Bases**

[11] Dominion: Mode de joc de League of Legends presentat oficialment per primera vegada el 26 de setembre de 2011. Aquest mode de joc tractava de controlar les bases dels enemics, eliminant-los o bé obtenint punts de captura que et feien guanyar la partida. Aquest mode va deixar de formar part dels modes del joc des del 22 de febrer de 2016.

A la taula anterior es comparen els diferents aspectes més distingibles de cada videojoc. Aquests es descriuen a continuació:

- **Perspectiva de la càmera**: Si la càmera està en tercera persona o es tracta d'un *top-down*[12] isomètric clàssic.
- **Corba de dificultat**: Si és accessible per jugadors nous i si és satisfactori per jugadors experimentats.
- **Dispositiu principal**: En quin dispositiu es jugarà principalment al joc.
- **Mecànica principal**: Es basa en les mecàniques clàssiques dels MOBA o revoluciona el gènere en algun sentit.

2.5 Públic objectiu i tipus de jugador

A l'hora de dissenyar i presentar el projecte hem de saber a quin públic va dirigit. Per aquesta tasca, durant els anys s'han anat fent servir diversos mètodes, com per exemple, categoritzar per rang d'edat o interessos del jugador.
El públic objectiu s'ha estimat que tingui entre 12 anys en amunt, ja que el joc, encara que no tingui sang explícita, tracta *d'eliminar* enemics (de manera fictícia en un mon utòpic) per aconseguir la victòria de la partida. En el cas de que la mort es tractés més profundament o amb més detall gràfic, aquesta edat oscil·laria entorn els 16 o, fins i tot, 18.

2.5.1 Taxonomia de Bartle

En aquest projecte per categoritzar el públic objectiu, s'ha utilitzat la Taxonomia de *Richard Bartle*, estudiada durant el curs. Aquesta proposa un perfil de jugador respecte el seu tipus de jugabilitat preferit.

En aquesta taxonomia existeixen 4 perfils:

- ***Socializers***: Els agrada contactar amb altres persones i sentir-se que no està sol al món. Els interessa els videojocs de més d'un jugador on poden col·laborar amb els demés per realitzar alguna tasca en concret.
- ***Explorers***: Els agrada explorar un món i la historia del mateix més que l'objectiu final del joc.
- ***Achievers***: Es centren en completar objectius, missions o el que el joc li requereixi en cada moment de la partida.
- ***Killers***: Busquen acció, fer estralls i es diverteixen al lluitar contra enemics, tant controlats per la IA, com altres jugadors en línia.

Abreviat com a *SEAK*, aquesta taxonomia no categoritza a un jugador dins d'un perfil, sinó que cada perfil es determina per un percentatge de cada tipus.

[12] Top-down: Tipus de càmera isomètrica picada, gairebé zenital en alguns casos, on sembla que el jugador està sent gravat per un dron que flota damunt seu.

En aquest projecte, el rol principal del públic objectiu és el de *Killer*. El MOBA és un gènere que es centra molt en el combat i fer-lo divertit, fluid i satisfactori és clau per l'experiència del jugador. El so, les animacions, els efectes i les interfícies que reflexen el canvi han de demostrar que hi ha impacte en les accions que es fan dins del joc i que el jugador està fent el màxim mal a l'enemic, encara que no sigui del tot així en termes estadístics.

El segon perfil més adequat per aquest gènere és l'*Achiever*. Aquest tipus de jugador li encanta aconseguir objectius i la victòria és una recompensa massa satisfactòria. La mecànica d'anar destruint torretes com a objectiu temporal, a part dels *minions* per guanyar or i anar escalant a la partida també incita a aquest perfil de jugadors a jugar més i més partides passant-s'ho bé.

El perfil de *Socializer* és un que també encaixa molt bé amb aquest gènere. Els aliats poden col·laborar per acabar amb l'equip contrari. A part, també poden interactuar amb l'equip enemic en un xat global de la partida en la gran majoria de jocs del gènere.

Finalment, el perfil d'*Explorer*. Aquest serà el perfil que menys s'explori en aquest gènere. El mapa serà limitat i només es tractarà d'una arena de combat, per tal, qui busqui una experiència de món obert o descobrir zones i misteris, aquesta no serà l'experiència que busca.

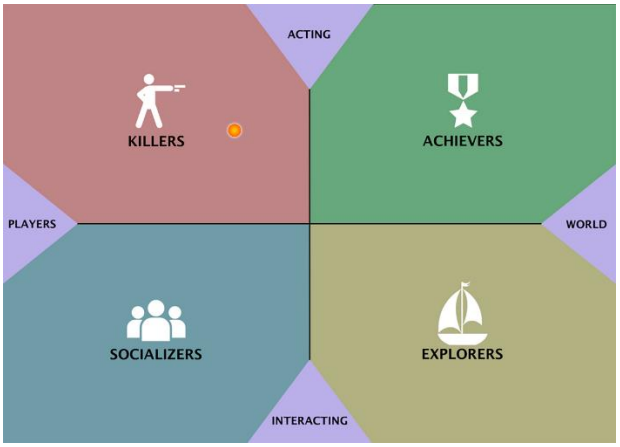

Figura 12: Taxonomia de Bartle per aquest projecte

A la imatge anterior, el punt taronja simbolitza el lloc on estaria el públic objectiu per aquest projecte. Si haguéssim de posar percentatges el perfil aproximat rondaria per un 60% *Killer*, 30% *Achiever*, 10% *Socializer* i 0% *Explorer*.

3 Planificació i metodologia

En aquesta etapa del projecte cal organitzar de forma correcta a tot l'equip de treball. Aquesta planificació és essencial pel bon desenvolupament del projecte i per tenir una idea més detallada del llistat de tasques que cal realitzar per acabar assolint l'objectiu amb èxit.

3.1 Llistat de tasques

Aquestes tasques es poden dividir en diferents categories segons el perfil del treballador que ha de realitzar la tasca:

3.1.1 Tasques de programació

El programador s'encarregarà d'implementar totes les funcionalitats del joc, tant implementar mecàniques, programar les accions de les animacions dels personatges i menús, com dissenyar i implementar un sistema multijugador.

- **Generar un sistema multijugador**: crear un sistema central on es centrin totes les connexions en línia pel joc. Hi ha d'haver usuaris, sales, partides, missatges entre clients, etc.

- **Crear un sistema de personatges modular**: Aquest sistema ha de permetre que el joc funcioni, sense importar quin dels personatges s'esculli i que s'adapti a cadascun.

- **Sistema d'ítems a la botiga modular**: A la botiga hi ha d'haver un llistat d'ítems que es puguin comprar i que funcionin amb qualsevol personatge sense importar les seves característiques.

- **IA de les entitats controlades per la màquina**: Aquestes entitats, com poden ser els *minions* o les torretes repartides pel mapa, han de ser independents a qualsevol jugador i actuar en tots els clients de la mateixa forma per assegurar la coherència dins les partides.

- **Totes les entitats del mapa són variacions d'una sola cosa**: Una entitat és qualsevol cosa que té vida i que (si no té cap efecte especial) pot morir quan aquesta arribi a 0. Les entitats seran fills d'aquesta única classe pare, probablement, abstracta[13]. Cada fill haurà d'implementar la funcionalitat del què passa quan cada acció de l'entitat s'activa.

[13] Classe abstracta: Classe de programació que no pot ser instanciada en cap moment i només serveix de model o referencia per les seves variacions (els seus fills). És similar a les interfícies amb la peculiaritat que aquestes últimes només defineixen i no implementen cap funcionalitat per defecte.

- **Programació de tots els botons de la interfície d'usuari**: Tota la funcionalitat dels botons dels menús com les interfícies dins del joc.

- **Atac bàsic del personatge**: Cada personatge té la mateixa manera d'atacar però no totes les animacions de cada personatge tarden el mateix. A més, cada personatge tindrà una velocitat d'atac diferent. El personatge es podrà moure en un rang mentre ataca però si surt del rang, l'atac es cancel·larà.

3.1.2 Tasques de dissenyador

Dissenyarà totes les habilitats dels personatges i estadístiques dels ítems de la botiga. La mecànica principal del joc (la carrera de bases) serà la mateixa que tots els altres videojocs del gènere.

- **Dissenyar les habilitats dels personatges**: Dissenyar el què faran les quatre habilitats de cadascun dels personatges del joc i donar-los una identitat.

- **Dissenyar els ítems de la botiga**: A la botiga hi haurà diferents ítems que es podran comprar per millorar les diferents estadístiques de cada personatge.

- **Dissenyar les interaccions bàsiques**: Per jugar s'ha de poder interactuar amb les altres entitats.

- **Dissenyar les mecàniques principals**: Com es pot guanyar o perdre? Què fa que un jugador sigui millor o pitjor que un altre? Com es pot aprofitar mecàniques que el joc brinda per tenir més probabilitats de guanyar?

- **Dissenyar el comportament de les entitats controlades per la IA**: Els *minions* i les torretes tindran un comportament determinat. Tindran un ordre de prioritat i atacaran a l'entitat amb més prioritat en cada moment.

- **Economia del joc**: El joc tindrà un sistema molt bàsic de monedes que es guanyen al matar entitats enemigues. Aquestes monedes es poden gastar a la botiga per comprar ítems per millorar les estadístiques d'un personatge.

3.1.3 Tasques d'artista 3D

Aquest serà l'encarregat de generar tots els models en 3D que hagin d'estar dins del joc.

- **Crear els personatges**: La creació i animació de tots els personatges serà una part vital pel joc. Aquesta part serà substituïda per la plataforma *Mixamo* que ja ens dona models animats optimitzats per videojocs de forma gratuïta.

- **Crear l'escenari del joc**: L'artista hauria de crear un escenari que mantingués les proporcions donades pel dissenyador però afegint-hi detalls i que fes servir un toc visual coherent per tot el joc.

- **Crear els efectes**: Totes les habilitats dels personatges, com efectes dels atacs de les entitats, han de tenir una entitat pròpia.

- **Crear les torretes, nexe i *minions***: Aquesta és una part essencial del joc. Cal un model per cada entitat.

3.1.4 Tasques d'artista UI/UX

La feina de l'artista d'interfícies serà crear tot el que inclouen els menús i totes les interfícies dins del joc.

- **Generar tots els menús del joc**: Aquests menús inclouran la següent llista.
    - Pantalla d'inici.
    - Menú principal.
    - Menú de col·lecció.
    - Menú de crear partida.
    - Menú de la llista de partides actives.
    - Lobby per escollir el personatge.
    - Pantalla d'estadístiques després de la partida.
    - Menú d'opcions.
    - Pantalla de càrrega.

- **Generar totes les interfícies dins del joc**: Aquests apareixeran dins del joc i són els següents.
    - HUD de la partida. Aquest inclou:
        - Temps actual de la partida
        - Per cada equip:
            - *Minions* eliminats
            - Torretes eliminades
            - Enemics eliminats
    - HUD del personatge. Aquest inclou tot el grup de:
        - Icona del personatge actual
        - Nivell actual
        - Progrés del nivell actual
        - Habilitats i l'estat actual
        - Vida i manà (actual i màxima). Aquests també inclouen un text per la regeneració d'aquestes estadístiques cada segon.
        - Inventari. Inclou tots els ítems comprats i els diners actuals.

3.1.5 Tasques d'artista de so

La feina de l'artista de so serà generar tots els sons del joc. En aquest cas, el prototipus no té cap tipus de so en el desenvolupament, per tant no s'ha inclòs cap tasca en aquest Apartat. Si s'hagués implementat algun sistema de so, les tasques serien:

- **Generar tots els sons de cada habilitat**: Aquests inclourien les 8 habilitats programades pel prototipus amb un so únic i distingible.

- **Crear tots els sons de les interfícies**: Cada botó tindrà un so quan es cliqui o quan es passi el ratolí per damunt. Totes les interfícies dins del joc també tindran aquest efecte de so.

- **Sons dels personatges**: Cada personatge tindrà unes frases doblades que s'hauran de gravar i editar per tal que entonin amb l'ambient del joc. Cada personatge cridarà quan es mori i quan alguna entitat li faci mal. Depèn de la quantitat de mal es reproduirà un so o un altre.

- **Sons de les entitats**: Totes les entitats del mapa (tant *minions* com torretes) tindran un so per cada acció que facin.

- **Música del joc**: El joc tindrà musica pròpia i, depenent de si s'està al menú principal, al Lobby o dins del joc, es reproduirà una de diferent.

3.1.6 Tasques de *Game Tester*

Aquest s'encarregarà de provar totes les funcionalitats del joc amb cada personatge i notificar si alguna cosa va malament.

3.1.7 Altres tasques

En aquí s'inclouran totes les tasques que no estan dins de cap dels grups esmenats anteriorment.

- **Documentació**: S'ha de generar una tasca per la feina que porta crear aquest document i la documentació alternativa del codi.

- **Direcció**: Per assegurar-nos de que tot el projecte funciona correctament hi ha d'haver un càrrec de responsabilitat que s'asseguri que el projecte va en els terminis que toca i que s'està fent el que es demanava.

3.2 Metodologia de treball

Per crear el prototipus, el desenvolupament s'ha partit en tres parts essencials: disseny, implementació i correcció d'errors. Dins d'aquests tres grups de tasques s'ha anat creant el projecte.

Per organitzar la feina s'ha fet servir el mètode *Kanban*. Aquesta és una metodologia derivada de la sèrie de pràctiques escrites al *Manifest Àgil*[14]. Aquests descriuen detalladament un conjunt de dotze principis per escriure millor codi més ràpid, amb l'objectiu principal de lliurar al client programari que aporti valor de forma ràpida i continua.

El primer que s'ha fet és crear un tauler a *Trello*[15]. Aquest és molt útil per mantenir un ordre i saber el que està passant en tot moment del desenvolupament del projecte.

Per cada tasca prou gran, s'ha establert unes dates límit per acabar-les. Aquestes dates s'han d'anar complint al peu de la lletra per poder acabar el projecte correctament a temps.

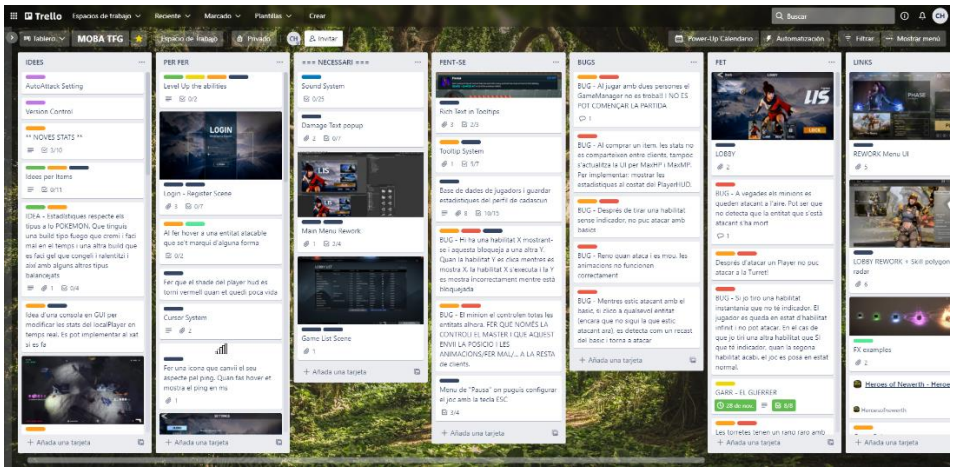

Figura 13: Captura de pantalla del tauler de Trello

[14] El Manifest Àgil es pot veure al següent enllaç: https://agilemanifesto.org/iso/ca/principles.html

[15] Trello: Una plataforma que fa servir la metodologia *Kanban* per organitzar projectes mitjançant cartes (que representen l'estat de cada tasca) en columnes. En aquest projecte s'han creat altres columnes auxiliars a part de les clàssiques "Per fer", "Fent-se" i "Fet".

3.3 Diagrama de Gantt

Per assignar cada grup de tasques en un temps s'ha creat el diagrama de Gantt següent:

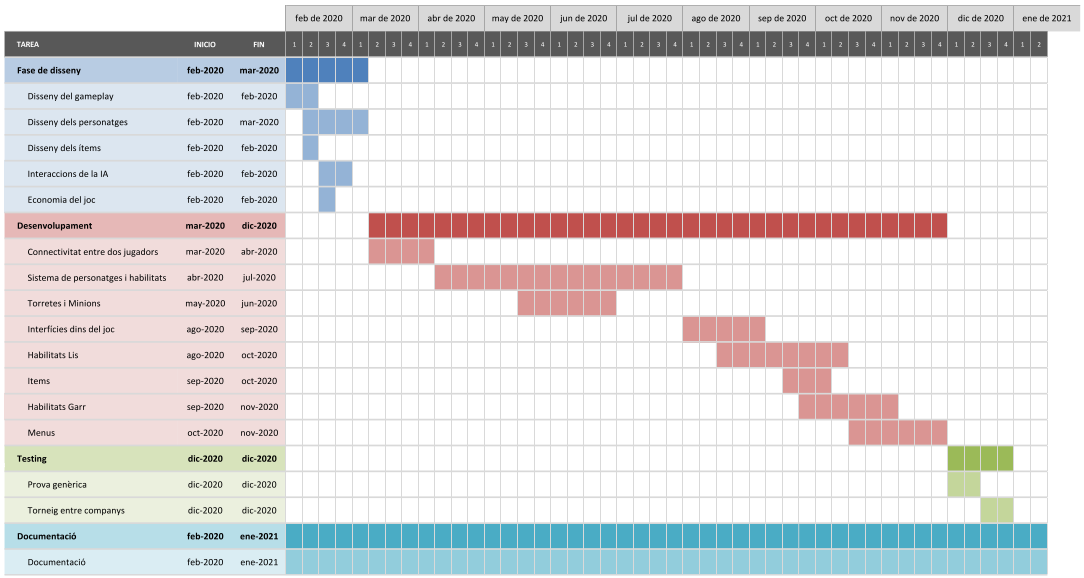

Figura 14: Diagrama de Gantt

4 Marc de treball i conceptes previs

En aquest Apartat s'entra en detall sobre el joc d'una manera relativament superficial per entendre el funcionament bàsic. En primer lloc s'explicaran els conceptes principals d'un videojoc MOBA, en segon i últim lloc s'entrarà en el vocabulari específic dels programes a fer servir.

4.1 Què és un MOBA?

En aquest Apartat explicarem el gènere MOBA per a què tothom ho pugui entendre.

4.1.1 Objectiu de la partida

El concepte bàsic de la jugabilitat del MOBA és bastant accessible en quant s'entén el disseny des de la base. Existeixen dos equips de cinc jugadors en cadascun. Els equips es troben en mapa quadrangular on hi ha tres línies[16] i tenen la forma de la Figura següent:

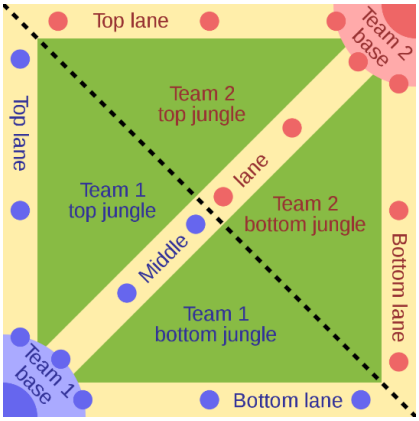

Figura 15: Mapa clàssic d'un MOBA

L'objectiu principal de la partida és acabar amb el nexe enemic situat a la base enemiga. El nexe és una estructura situada a la base de cada equip, que no ataca (en la majoria de MOBA) i que l'únic objectiu a la partida és que aquesta acaba quan és destruït. El primer equip que acabi amb el nexe enemic, guanya la partida.

Figura 16: Nexe de l'equip vermell de League of Legends

[16] Les línies es solen anomenar pel nom en anglès: TOP, MID (de *middle*) i BOT (de *bottom*).

4.1.2 Torretes

Per poder arribar al nexe enemic s'haurà de travessar qualsevol de les tres línies derruint totes les torretes de la línia fins a arribar a la base enemiga. A cada línia hi ha torretes (normalment tres, com es mostra a la *Figura 14*). Aquestes s'encarreguen de bloquejar el pas als enemics ja que cadascuna d'aquestes atacarà a tots els enemics dins d'un rang.

Figura 17: Torreta de l'equip vermell de League of Legends

4.1.3 Minions

Els *minions* són entitats que surten des de cada base i van directes a la línia a la que tenen designada. Cada *minon* té un únic objectiu: caminar recte per la línia fins a trobar un objectiu al que atacar. Si en té múltiples, atacarà al que tingui més prioritat.

Figura 18: Minions de League of Legends

Per evitar que la torreta ataqui a un personatge, s'ha de deixar que els *minions* vagin davant. Com la torreta només pot tenir un objectiu alhora, mentre aquesta ataca a *minions* aliats, el jugador pot anar fent mal amb els atacs bàsics i així treure-li vida fins a poder avançar a la següent torreta, fins a arribar al nexe.

Figura 19: Torreta de l'equip vermell atacant a uns minions de l'equip blau a League of Legends

4.1.4 Personatges

A cada equip hi ha cinc jugadors, els quals controlen un personatge. Aquest personatge s'escull abans de començar la partida i sempre serà el mateix durant aquesta. Un personatge té diferents maneres d'interaccionar amb el món.

- Es poden moure pel mapa
- Poden atacar amb un atac bàsic que pot ser de dos tipus:
    - Atac cos a cos
    - Atac a distància

Figura 20: Indicador del rang de l'atac bàsic a distància a League of Legends

- Tenen un conjunt de cinc habilitats (quatre normals i una passiva[17])

Figura 21: Habilitats de Jinx a League of Legends

Cadascuna de les habilitats es pot prémer per activar una certa funcionalitat. Per exemple, la R de *Jinx* (vista a la Figura 19) llença un coet que surt del personatge en direcció al ratolí. Si aquest toca un personatge enemic, aquest explota i li fa mal. A part, quan explota fa mal a totes les entitats (personatges o *minions*) enemigues en un radi.
Cada habilitat té altres estadístiques a part de la acció que fa al activar-la.

- El *manà*[18] que gasta al activar-la
- El temps[19] que s'ha d'esperar per tornar a activar-la

Figura 22: Habilitats en espera de League of Legends

4.1.5 Estadístiques de les entitats

Cada entitat al mapa té unes estadístiques específiques. Aquestes són, entre altres, les següents:

- **Vida**: Quan aquesta arriba a zero, l'entitat es considera morta.
- **Mana**: Aquesta representa la quantitat d'habilitats que pot fer servir.
- **Mal físic**: Representa el mal que farà l'entitat amb els atacs bàsics.
- **Mal màgic**: Mal que farà amb les habilitats.
- **Armadura**: Resistència al mal físic. Aquesta funciona com un filtre de reducció del mal entrat quan s'aplica l'atac bàsic. Per exemple, si l'atac bàsic enemic hauria de fer 100 de mal i el personatge aliat té 20 d'armadura, el mal realment aplicat seria de 80[20].
- **Resistència màgica**: El mateix que l'armadura però reduint el mal màgic.
- **Velocitat de moviment**: Rapidesa amb la que es mou l'entitat pel mapa.

[17] Les habilitats passives són efectes per al personatge que s'està controlant que s'activen quan un event es dispara. Per exemple: "Guanya 10 de mal físic durant 2 segons cada cop que es rep mal". Cada personatge en té una de pròpia i única.
[18] El manà és l'àlies de la clàssica "*stamina*" que es gasta cada cop que es fan servir habilitats i es regenera en el temps.
[19] "*Cooldown*" en anglès, fa referència a el refredament que ha de tenir l'habilitat abans de poder-se emprar un altre cop.
[20] Si es fes servir la formula de restar quantitats. Al joc final s'ha fet servir una altra, explicada a l'Apartat 6.3.3.

4.1.6 Economia del joc

Al matar entitats, cada jugador obté una moneda, en aquest cas "or". Aquest es pot anar acumulant a l'inventari o es pot gastar a la botiga situada a la base de cada equip. Aquesta botiga ofereix ítems a canvi de l'or i aquests ítems donen un potenciat a les estadístiques del personatge. Alguns ítems també obtenen passives que es sumen a la del personatge que s'estigui jugant en aquell moment.

Figura 23: Inventari i comptador d'or de League of Legends

4.1.7 Estratègia de partida

Hi ha cinc jugadors i tres línies, llavors, com es disposen pel mapa per organitzar-se? A l'esquema clàssic de MOBA existeixen cinc posicions que es traslladen a cinc rols (o també dit *posicions*) dins la partida. Aquests rols depenen del kit d'habilitats de cada personatge. Els rols són els següents:

- **El TOP**: Va a la línia del TOP i sol ser un guerrer o tanc. Aquests solen ser personatges que fan mal cos a cos i tenen una quantitat de vida alta.

- **El JUNGLA**: Aquest tipus va per la jungla matant entitats neutrals per aconseguir experiència i, a vegades, surt de la jungla i ajuda a alguna de les línies aportant un avantatge temporal[21] a causa de ser un més (els personatges que ja van a la línia i el jungla que acaba d'arribar).

- **El MID**: Va la línia del MID i sol ser o bé un assassí físic o màgic. Pot ser que vagin cap a altres línies de forma temporal[22] ja que, a l'estar al mig del mapa, poden fer-ho amb facilitat.

- **L'ADC**: Va la línia del BOT i sol ser un tirador a distància que fa mal físic. Aquest té poca vida però fa molt de mal per segon.

[21] Aquest desavantatge temporal té un nom en anglès molt utilitzat i és "*to gank*" o mal traduït a l'argot dels jugadors: "*gankejar*". Ben traduït es diria "*plantar una emboscada*". Aquesta estratègia consisteix en agafar desprevinguts als enemics per l'esquena quan el *jungla* aliat ve a ajudar amb la possibilitat d'eliminar a l'enemic o als enemics que hi hagi en aquella línia.

[22] D'aquest moviment se'n diu "to roam" o "roaming". Traduït correctament es diria "vagar" o recórrer el mapa en busca de possibles enemics a eliminar. No necessàriament ha de ser el MID el qual ho faci, ho poden fer tots els jugadors de la partida, però per exemple poques vegades un ADC anirà al TOP.

- **El SUPPORT**: Va a la línia del BOT i acompanya l'ADC. Sol curar i posar escuts[23] per ajudar que el tirador faci la màxima quantitat de mal sense exposar-se per la poca vida que té.

Figura 24: Disposició de posicions a League of Legends

Pot ser confús per nous jugadors el fet de que algunes línies i rols tinguin el mateix nom, però només cal pensar en si a la línia hi ha un jugador o no. En el primer cas tindrà el nom de la línia, en cas contrari no podem posar el mateix nom perquè hi hauria dos posicions amb el mateix nom.

[23] Els escuts són vida extra que ignora la vida màxima de l'entitat. Es considera una barra de vida extra que té prioritat quan l'entitat rep mal. Normalment els escuts són temporals i no duren més de dos o tres segons. També pot ser que l'escut només absorbeixi un tipus de mal o tots.

4.2 Vocabulari sobre el marc de treball de Unity

A l'estar treballant amb Unity, aquest ofereix un marc de treball amb un cert vocabulari especialitzat per cada concepte dins del motor. A continuació s'explicarà cada concepte:

- **Scene**: Una Scene, o escena en català, és un grup d'elements que formen un nivell. Tots els elements, tant el jugador com el terreny, estan dins d'aquesta escena. Cada escena és independent així que s'assegura que, entre un nivell i un altre, no es manté cap mena d'informació local a aquell nivell. Es pot tenir més d'una escena oberta concurrentment, però la forma per defecte de treballar és només mantenir una escena al mateix temps.

- **GameObject**: Un GameObject, o objecte del joc en català, és qualsevol entitat que tingui un eix de coordenades dins de l'escena. Aquest eix es defineix amb la posició, rotació i escala de l'objecte respecte el (0,0) de la escena.

- **Component**: Un Component és tot el que es pot afegir al GameObject perquè tingui un cert comportament. Per exemple, es pot afegir un BoxCollider a un GameObject perquè els demés objectes col·lideixin amb aquest.

- **Script**: Un Script, o fitxer de codi, és un fitxer de text que descriu el comportament de qualsevol entitat dins de Unity. Realment qualsevol Component és un Script, però n'hi ha que estan implementats nativament a Unity amb la finalitat de facilitar el desenvolupament.

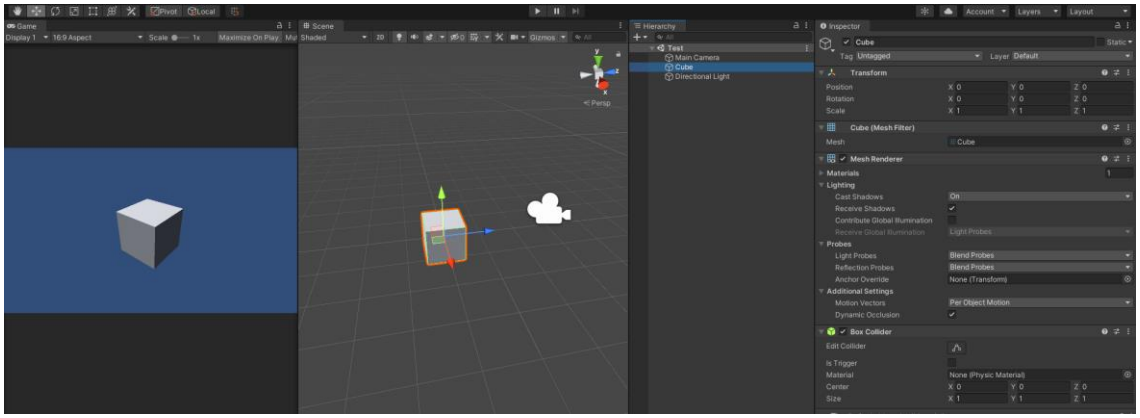

Figura 25: Captura de pantalla de Unity

Per exemple, a la Figura 25, tenim a la part esquerra la vista del joc. Aquesta és una simulació de com es veuria el joc des de la perspectiva de la càmera en aquell moment. Més a la dreta tenim la vista de la escena, vista des d'una càmera virtual que ens permet moure'ns lliurement i seleccionar objectes. Més a la dreta tenim la vista de la Jerarquia. Aquesta disposa tots els elements dins de l'escena activa en aquell moment. En aquest cas tenim seleccionat el *GameObject* "*Cube*".

A la part més dreta de la imatge tenim la vista de l'*Inspector*. Aquesta vista permet veure els components del *GameObject* que tenim seleccionat en aquell moment. En aquest cas, com tenim seleccionat l'objecte "Cube", veiem els *Components*. Aquests són:

- **Transform**: Aquest *Component* és obligatori per qualsevol *GameObject* i descriu la posició, rotació i escala de l'objecte a l'escena respecte el pare.

- **Mesh Filter**: Aquest manté els vèrtex, les arestes i les cares del polígon. Si es vol mostrar alguna altra forma, només cal canviar l'objecte "*Mesh*" que es passa com a paràmetre i es mostrarà una altra malla.

- **Mesh Renderer**: Aquest descriu com s'ha de renderitzar la malla descrita pel Mesh Filter.

- **Box Collider**: Aquest *Component* fa que l'objecte sigui sòlid i pugui col·lidir amb altres objectes amb qualsevol tipus de *collider*. Si s'activa la opció "*is Trigger*", aquesta propietat física desapareix i es transforma en un activador de funcions. Per exemple, si estem al *Mario Kart* i volem comprovar quan el jugador passa per la línia de meta, podem fer us d'un *Box Collider* amb aquesta opció activa per comprovar-ho (el jugador també tindrà un *Collider*, pot ser *trigger* o no).

5 Disseny del videojoc

En la creació d'un videojoc és important tenir en compte l'etapa de disseny. En aquest capítol es recullen les parts més importats del document de disseny inicial presentat per al projecte.

5.1 Objectiu del projecte

L'objectiu principal d'aquest projecte és implementar un videojoc del gènere MOBA utilitzant la plataforma que es descriurà a l'Apartat 6. Aquesta haurà de ser flexible a possibles canvis i escalable per noves mecàniques i personatges. També haurà d'implementar totes les funcionalitats bàsiques d'un videojoc MOBA com es descriu a l'Apartat 4.1.

5.2 Llistat de mecàniques

Cada videojoc oferta al jugador amb una sèrie de mecàniques que són les accions que pot fer al món per canviar l'estat amb el fi de passar-se un nivell o guanyar una partida. Aquestes mecàniques han d'estar acord amb el gènere que s'està plantejant.

5.2.1 Espai

El joc ha d'estar contingut en un espai. Aquest pot ser limitat o infinit (si és procedimental[24]). En aquest cas existirà una arena on els jugadors podran lluitar entre si. Es pot veure la forma que té a la Figura 26.

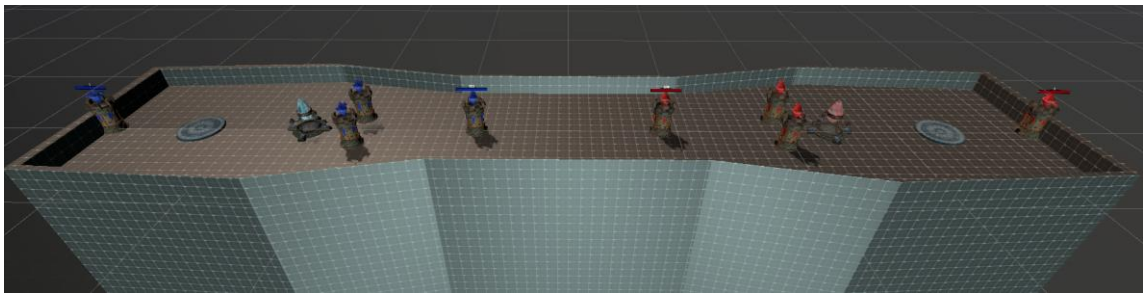

Figura 26: Vista aèria de l'arena de batalla

En aquest disseny s'ha escollit una variant del mapa clàssic de tres línies per un de simplificat, on només hi ha una línia i només hi haurà un combat 1 contra 1. Aquest mapa consta de dues bases, dos nexes, tres torretes i un làser per cadascun dels equips. Existeixen les parets que limiten al jugador que surti del mapa. Les torretes fan de paret lògica i guien al jugador cap a la base enemiga. Si el jugador intenta travessar-la sense destruir-la, aquesta l'atacarà i el jugador quedarà exposat davant l'enemic, fent-lo una víctima molt vulnerable. La càmera es quedarà tal i com és als videojocs MOBA clàssics. Una perspectiva *top-down* com la explicada a l'Apartat 2.4.2.

[24] Els espais procedimentals són aquells que no estan fixes a un contingut en concret, sinó que es guien per un algorisme que genera el món de forma pseudo-aleatòria. Un exemple molt famós d'espai procedimental és *Minecraft*.

5.2.2 Escollir personatge

El personatge dins de la partida ha de ser dinàmic i ha de poder ser escollit pel jugador abans de començar cada partida. Hi haurà una pantalla on es vegin totes els possibles personatges amb la possibilitat d'escollir-ne només un. Aquesta pantalla servirà per determinar l'estil de joc del jugador a la partida respecte al personatge escollit, com es pot veure a la Figura 27.

Figura 27: Pantalla de Lobby

Dins del joc aquests canvis hauran d'estar reflectits i que totes les funcionalitats del personatge siguin les escollides pel jugador, com es pot veure a la Figura 28.

Figura 28: HUD dinàmic del jugador dins la partida

Aquest personatge serà independent al que esculli l'enemic. Qualsevol combinació de personatges serà vàlida en aquest prototip. D'aquesta manera es podran comprovar la funcionalitat de totes les interaccions entre personatges.

5.2.3 Personatge

El personatge serà l'entitat del mapa que el jugador controlarà i podrà moure's pel mapa de forma lliure. Tindrà unes estadístiques que aniran canviant durant la partida i aquestes afectaran al rendiment del personatge durant la partida. A més el personatge podrà córrer mantenint la tecla SHIFT que multiplicarà la velocitat de moviment per una certa quantitat.

5.2.3.1 Estadístiques bàsiques

Cadascun dels personatges tindrà unes certes estadístiques bàsiques. Aquestes seran les descrites a continuació:

- **Vida Actual**: La entitat es considerarà morta quan arribi a 0.

- **Vida Màxima**: La entitat es regenerarà com a màxim aquesta quantitat. Quan el personatge reaparegui a la base, ho farà amb aquesta quantitat.

- **Regeneració de Vida**: Cada cinc segons el personatge es regenerarà una certa quantitat de vida automàticament.

- **Mana Actual**: La capacitat de tirar habilitats. Aquesta barra es gastarà cada cop que s'activi una habilitat i anirà regenerant-se en el temps.

- **Mana Màxim**: La quantitat màxima de mana que pot tenir el personatge. Quan el personatge reaparegui a la base, ho farà amb aquesta quantitat.

- **Regeneració de Mana**: Cada cinc segons el personatge es regenerarà una certa quantitat de mana automàticament.

5.2.3.2 Moviment

El personatge podrà moure's de dues maneres diferents: caminant o corrent. Aquest mode podrà ser alternat mantenint la tecla SHIFT del teclat. Aquesta mecànica presenta dues estadístiques més:

- **Velocitat de moviment**: Unitats per segon que es mourà el personatge pel mapa caminant.

- **Multiplicador d'esprint**: Aquesta unitat es multiplicarà a la velocitat de moviment mentre la tecla SHIFT es mantingui premuda.

5.2.3.3 Progressió de nivell

També existirà una progressió de nivell que farà que el personatge vagi adquirint experiència en el temps i, per tant, pujant de nivell i adquirint millores en les seves estadístiques base.

- **Nivell**: Aquest designa el nivell de poder que té el personatge. Cada cop que aquest pugi de nivell, les estadístiques base augmentaran. Hi ha un nivell màxim i és el 15.

- **XP**: Progressió del nivell actual. Les estadístiques només canvien quan aquesta XP arriba al límit requerit per aquest nivell. Cada nivell requereix cada cop més XP i podrà ser calculada amb la fórmula següent ($\boldsymbol{xp}$ és la funció d'experiència respecte $\boldsymbol{l}$, el nivell de la entitat):

$$\boldsymbol{xp}(\boldsymbol{l}) = ((\boldsymbol{l} - \mathbf{1}) * \mathbf{100}) + \mathbf{250}$$

Cada cop que el personatge puja de nivell, aquest guanya estadístiques base. Aquestes venen descrites per les següents estadístiques:

- **Augment de Vida Màxima**: Descriu la quantitat de vida màxima que se li suma al personatge cada cop que puja de nivell.

- **Augment de Mana Màxim**: Descriu la quantitat de mana màxim que se li suma al personatge cada cop que puja de nivell.

- **Augment de Regeneració de Vida**: Descriu la quantitat de regeneració de vida que se li suma al personatge cada cop que puja de nivell.

- **Augment de Regeneració de Mana**: Descriu la quantitat de regeneració de mana que se li suma al personatge cada cop que puja de nivell.

5.2.3.4 Atac bàsic

Els personatges tenen la possibilitat d'atacar amb un atac bàsic. Aquesta és una interacció que fa que el personatge dins d'un rang cap a una entitat enemiga. Aquesta mecànica no gasta mana al accionar-la i sempre es pot fer servir.

Figura 29: Garr atacant amb un atac bàsic a un minion

Aquest atac funcionarà de la següent manera:

- Si el jugador clica amb el ratolí una entitat enemiga en rang, el personatge començarà a atacar-la.
- S'activarà un indicador del rang del personatge mentre aquest l'estigui atacant. Aquest canviarà de lloc si es canvia d'objectiu a mig atac.
- Si es torna a clicar a la mateixa entitat mentre s'està fent la animació, el personatge recordarà que ha de tornar a atacar. Només recordarà un atac, per tant, si li donem molts cops seguits mentre s'està fent el primer atac i després deixem de clicar, el personatge acabarà l'atac actual, en farà un més (el que recorda) i deixarà d'atacar quedant-se quiet.
- Mentrestant, el personatge es pot moure lliurement dins del rang d'atac.
- Si el personatge corre o surt del rang la acció es cancel·larà (i també oblidarà si ha de tornar a atacar).

Aquesta nova mecànica introdueix les següents estadístiques pel personatge:

- **Mal d'atac**: Mal que fa l'atac bàsic a l'entitat enemiga.

- **Rang d'atac**: Radi o distància màxima a la que cal que l'enemic estigui per cancel·lar l'atac bàsic.

- **Velocitat d'atac**: Aquesta determinarà la velocitat de l'animació d'atac. Per tant, quanta més velocitat, més atacs per segon es podran aplicar i més mal es farà a la llarga.

- **Armadura**: Resistència de qualsevol entitat de resistir *mal d'atac*.

5.2.3.5 Habilitats

Cada personatge tindrà un conjunt d'habilitats. En tots els personatges implementats fins ara, tots tenen quatre habilitats actives. Cadascuna està vinculada a un número del teclat alfanumèric.

Figura 30: Habilitats de Garr

Les habilitats funcionaran de la següent manera:

- Si l'habilitat no es troba disponible, la tecla s'ignorarà. Pot no estar disponible per les següents causes:
  - Falta manà.
  - Està bloquejada.
  - El personatge està mort.
  - El temps de refredament és més gran que zero.

- Quan el botó de l'habilitat es prem, i mentre està premut, aquest mostra un indicador, si l'habilitat ho requereix. Hi ha tres tipus d'habilitats en aquest aspecte:

  - **Habilitats sense indicador**: Que s'activaran quan es premi el botó i no mostraran cap indicador.

  - **Habilitats amb indicador direccional**: Que mostraran indicador i aquest seguirà la trajectòria del ratolí per mostrar cap on es llençarà l'habilitat.

  - **Habilitats amb indicador estàtic**: Que mostraran indicador però aquest romandrà estàtic ja que l'habilitat no té direccionalitat.

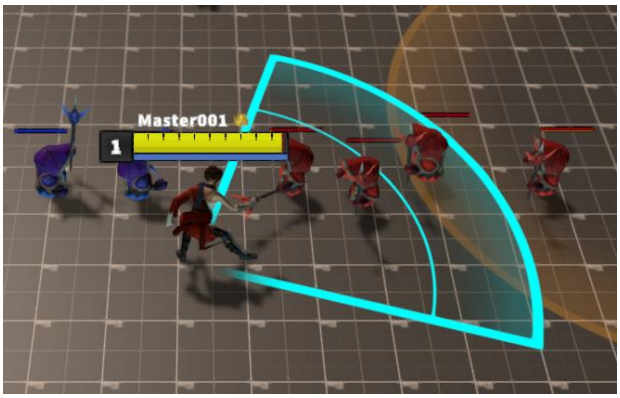

Figura 31: Indicador de l'habilitat

- Si mentre el botó està premut, es clica el botó dret del ratolí, aquesta habilitat es cancel·la i l'indicador desapareix.
- Si es deix anar el botó i l'habilitat no s'ha cancel·lat, aquest s'activarà i farà la següent seqüència d'accions.
    - L'habilitat gastarà el mana que costa.
    - S'activarà l'habilitat i s'executarà la funcionalitat que té implementada.
    - L'habilitat bloquejarà a totes les altres habilitats[25] mentre es tira, per evitar interaccions que podrien causar errors i interaccions no desitjades.
    - Començarà un comptador[26] que evita que es pugui tornar a tirar l'habilitat en un temps.

Aquesta nova mecànica introdueix següents estadístiques pel personatge:

- **Poder d'habilitat**: Mal que fa l'habilitat a l'entitat enemiga.

- **Resistència màgica**: El mateix que l'armadura però amb mal d'habilitat.

Aquesta nova mecànica introdueix següents estadístiques per cada habilitat:

- **Mal de l'habilitat**: Mal base que fa l'habilitat a l'entitat enemiga[27].

- **Nom**: Nom que mostrarà l'habilitat quan es necessiti la informació.

- **Descripció**: Petita descripció a mostrar de l'habilitat.

- **Cost de mana**: Quantitat de mana que costa l'habilitat.

- **Duració**: Temps que dura l'habilitat en tirar-se. Aquest serà el temps que les altres habilitats romandran bloquejades.

Cada personatge tindrà un kit d'habilitats diferents. Al joc hi haurà dos personatges que es podran jugar dels quatre dissenyats. El disseny inicial dels quatre personatges es descriurà a l'Apartat següent. Cal aclarir que aquest és el disseny inicial que es va plantejar pels personatges. Al projecte final aquests han patit una sèrie de canvis que fan que la seva implementació sigui més escalable.

[25] Algunes habilitats no cal bloquejar-les ja que no resulten problemàtiques. Les habilitats que es bloquegen seran les descrites per una llista que guardarà la pròpia habilitat.

[26] Aquest comptador comença a comptar a partir de que el botó es deix de prémer i l'habilitat s'activa. El comptador continua comptant mentre el personatge està mort. No comença a comptar ni quan es comença a mostrar l'indicador, ni quan l'habilitat acaba.

[27] El mal d'habilitat i el poder d'habilitat es sumen i formen el mal màgic total. Aquest mal després es redueix amb la resistència màgica de l'entitat que rep el mal.

5.2.3.5.1 Lis – La ninja

Com es pot veure a la Figura 32, Lis és una *ninja* que ataca cos a cos i regenera vida al atacar. Guanya velocitat de moviment amb el *combo* ben executat. Les seves habilitats són les següents:

- **Hook kick**: Lis es prepara i tira una puntada alta que fa mal a l'àrea davant d'ella. Fa mal a totes les entitats dins la zona i aplica més mal a les entitats més al centre (quant més al centre més mal, progressiu entre un mínim i un màxim).

- **Focus**: Lis es concentra. Es regenera vida que escala amb la vida que li falta i l'atac amb qualsevol de les seves habilitats augmenta.

- **Dash**: Lis es mou una distancia ràpidament. Si ha passat per damunt d'un personatge enemic, el refredament de *Hook kick* es redueix a 0.

- **Hurricane Kick**: Lis comença a rodar sobre si mateixa amb la cama estirada fent mal en àrea. La seva velocitat de moviment es veu reduïda mentre es tira l'habilitat.

Figura 32: Model de Lis

5.2.3.5.2 Garr – El guerrer

Com es pot veure a la Figura 33, Garr és un guerrer amb espasa i escut que es basa en el CC[28] i té molta vida. Guanya resistència amb l'escut i pega amb l'espasa cos a cos. Les seves habilitats són les següents:

- **Long sword**: Garr salta amb l'espasa i pega en una àrea estreta i llarga davant seu. Mentre està saltant, Garr rep un 50% més de mal, però si encerta l'habilitat, es regenera la vida extra treta i li aplica de mal a l'enemic.

- **Armor, UP!**: Garr es posa l'escut davant. Mentre està amb l'escut no pot atacar ni tirar habilitats i es mou un 50% més lent. Aquesta habilitat dura 2s encara que es pot cancel·lar tornant a clicar el botó. Un percentatge del mal que hagi rebut, l'aplicarà a la següent habilitat. Aquest percentatge escala amb el nivell (de l'habilitat o del personatge).

- **Kick**: Garr pega una puntada frontal que fa mal i immobilitza a l'enemic.

- **Power-up**: Garr guanya la passiva de "La Llum de l'Esperança". La Llum fa que Garr recordi tots el mal aplicat en forma *Armor, Up!* i aplica un percentatge d'aquest en forma de mal en les habilitats.

Figura 33: Model de Garr

[28] CC: Crowd Control. Traduït al català seria "control de masses". El fet de que un personatge pugui limitar el moviment o la capacitat d'atacar (amb habilitats o atacs bàsics) d'un enemic.

5.2.3.5.3 Reno – El mag

Com es pot veure a la Figura 34, Reno és un mag que ataca a distancia i fa CC. Fa bastant de mal però les seves habilitats tarden molt de temps en poder-se tornar a activar.

- **Magic missile**: Reno tira una bola màgica que fa més mal quan més lluny estigui (o quant més temps hagi passat des que s'ha tirat). Aquesta habilitat fa el doble de mal a enemics immobilitzats.

- **Deadlock**: Reno es protegeix amb un escut màgic que li impedeix moure's i el protegeix d'un 100% del mal rebut durant 0.5s. Aquest efecte es pot cancel·lar tornant a clicar.

- **Mental Control**: Reno genera una zona en forma de mitja lluna davant seu que immobilitza i fa mal a tothom dins.

- **Super Ultra Mega Magic Ray**: Reno acumula energia i la desprèn en forma de raig que afecta a tots els enemics al seu pas. Habilitat amb molt de rang.

Figura 34: Model de Reno

5.2.3.5.4 Holt – L'arquera

Com es pot veure a la Figura 35, Holt és una arquera que ataca a distància, té poca vida i es mou ràpid.

- **Unerring arrow**: Holt tira una fletxa de llarg abast que travessa entitats i que fa mal a tothom qui toqui.

- **Lowering guard**: Holt guarda l'arc i obté un 100% de velocitat de moviment extra. En aquest estat, Holt no pot tirar cap altra habilitat. Al tornar a clicar aquesta, Holt torna a agafar l'arc i torna a l'estat normal.

- **The Fenix Jump**: Holt fa un salt cap a la direcció apuntada amb el cursor.

- **Arrow Burst**: Holt s'acotxa i redueix un 30% la velocitat de moviment. Holt tira 3 ràfegues de fletxes en la direcció escollida. Holt es pot moure mentre es tiren les ràfegues, canviant la posició instanciada de les ràfegues.

Figura 35: Model de Holt

5.2.3.6 Atac crític

Hi ha una certa probabilitat de que un atac bàsic o una habilitat apliqui un mal crític. Aquesta probabilitat serveix per què, cada cop que el personatge aplica qualsevol tipus de mal a una entitat enemiga, aquest apliqui o no un mal crític. Aquesta mecànica introdueix dos nous tipus d'estadístiques al personatge:

- **Probabilitat de crític**: Aquesta serà la probabilitat mitjana a la qual puguin sortir els cops crítics.

- **Poder de crític**: Aquest serà l'escalar el qual multiplicarà el mal inicial per potenciar-lo.

5.2.3.7 Inventari i monedes

Com es pot veure a la Figura 36, Cada jugador tindrà un inventari amb ítems i monedes. Cada cop que el personatge elimini un altre personatge, un *minion* o bé una torreta enemiga, el personatge rebrà una recompensa en forma de monedes d'or. Cada jugador començarà amb 500 monedes d'or al principi de la partida i s'anirà regenerant una certa quantitat cada cinc segons.

Figura 36: Inventari del personatge

Aquesta mecànica afegeix una estadística al personatge:

- **Regeneració d'or**: Quantitat de monedes es guanyaran cada cinc segons.

A la botiga es poden comprar ítems amb aquestes monedes d'or com es descriu a l'Apartat següent. Aquests ítems potencien les estadístiques del personatge mentre es tenen a l'inventari. Aquests es poden vendre amb el botó dret del ratolí per una fracció del cost original.

5.2.4 Botiga

La botiga és un lloc on es poden comprar ítems. Com es pot veure a la Figura 37, Aquests costaran una quantitat d'or i aportaran una millora a les estadístiques del personatge. Tots els ítems seran únics. Això vol dir que no es podran comprar dues vegades per rebre la millora múltiples cops. Es pot obrir la botiga i comprar ítems en qualsevol lloc del mapa, inclús mentre s'està en combat o mort.

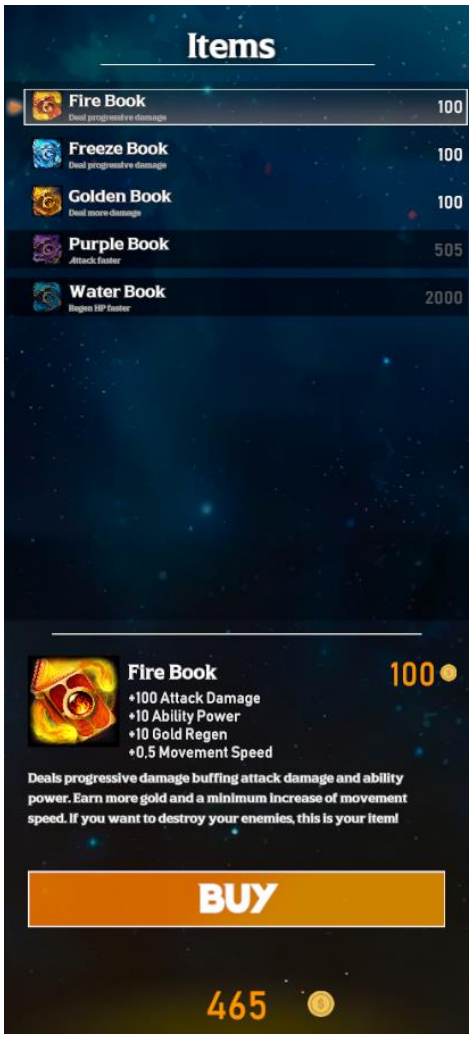

Figura 37: Botiga

Els ítems que no es puguin comprar es marcaran de forma més fosca. Es podran clicar per veure més informació però el botó de comprar estarà desactivat. També es podrà comprar sense veure la informació amb el botó dret del ratolí sobre l'ítem a la llista. Els ítems poden no estar disponible per les següents causes:

- Per què no es té suficients monedes per comprar-lo
- Per què l'ítem ja s'ha comprat.

5.2.5 Minions

Els *minions* son entitats d'un dels equips que surten de cada base i el seu objectiu és la base enemiga. Aniran atacant a tots els enemics que es trobin pel camí. Els *minions* tindran l'aspecte que es pot veure a la Figura 38.

Figura 38: Model d'un Minion

A l'hora d'escollir objectiu, els *minions* escolliran al primer enemic que hagi entrat al seu rang. Quan aquest surti del rang o es mori, el *minion* atacarà al següent. Si no hi ha enemics en rang, el *minion* seguirà el seu rumb cap al a base enemiga.

5.2.6 Estructures

Les estructures són un tipus d'entitat immòbil que pot ser d'algun dels dos equips. A les estructures només se li pot fer mal amb els atacs bàsics, pel què les habilitats són inútils contra aquestes entitats. Els dos tipus d'estructures es descriuran als Apartats següents.

5.2.6.1 Torretes

Les torretes (veure a la Figura 39) seran un tipus d'estructures que atacaran a tots els enemics que tinguin en rang. Aquestes estructures faran mal verdader. Aquest mal no té cap estadística que el resisteixi, per definició. Aquestes atacaran als enemics en rang amb la prioritat següent (de més a menys prioritari):

- Primer atacaran a tots els *minions* enemics dins del rang.
- Quan no quedin més *minions*, serà llavors quan atacaran als jugadors enemics que hi hagi en rang.
- Si no hi ha cap enemic, les torretes esperen.

Figura 39: Model d'una Torreta

5.2.6.2 Nexe

Com es pot veure a la Figura 40, aquesta estructura només té vida i no fa res. Encara així, el nexe és la estructura més important de la partida, ja que d'aquesta depèn la victòria. El primer equip que destrueixi el nexe enemic guanya la partida.

Figura 40: Model del Nexe

Per evitar que els jugadors més astuts ignorin les torretes i ataquin directament al nexe hi haurà un sistema d'inhabilitació d'estructures on cadascuna depèn de l'anterior. Per tant, s'haurà de destruir les torretes en ordre fins a arribar al nexe, en cas contrari és completament impossible interactuar amb l'estructura.

5.2.7 Pantalla de mort

Quan el personatge es quedi sense vida, aquest morirà i romandrà al terra amb l'animació de mort. En aquest estat no es podrà fer res més que esperar que el comptador arribi a zero, o bé comprar una reanimació (veure a la Figura 41).

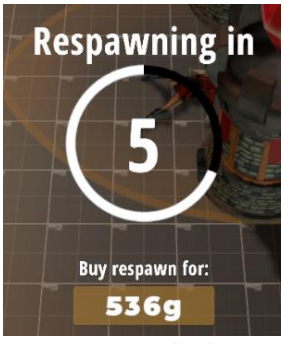

Figura 41: Pantalla de mort

El temps del comptador de mort és variable i escala amb el nivell del personatge. Això vol dir que si el personatge és nivell 15, caldrà esperar més temps que si és nivell 4. El que costa la reanimació instantània també és variable i depèn del temps que falta per reanimar-se de forma automàtica.

5.3 Funcionament de la partida

L'objectiu principal de la partida és acabar amb el nexe enemic. Per tant, s'ha de competir amb el contrincant per tal de ser el més ràpid a l'hora d'atacar torretes. És important eliminar *minions* i, si es pot, jugadors enemics ja que aquests donen or i és important per sumar estadístiques i ser més fort a l'hora del combat jugador contra jugador. A continuació es pot veure a la Figura 42 la representació del flux del comportament del jugador a una partida en trets generals.

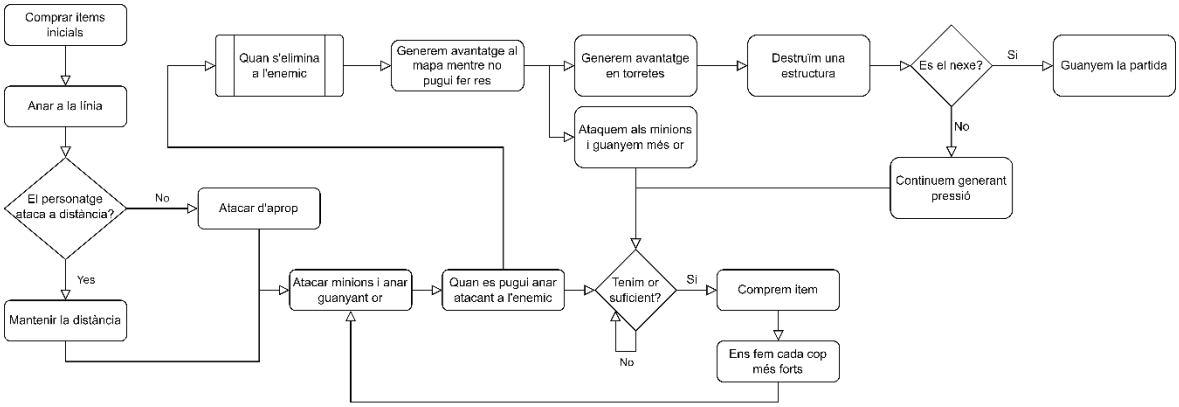

Figura 42: Diagrama de flux d'una partida

5.4 Economia de la partida

Per entendre l'economia de la partida és molt recomanable seguir una sèrie de principis de disseny. En aquest cas, el terme més essencial és "l'efecte bola de neu" o "*snowball effect*", en anglès. Aquest sorgeix de la metàfora que, quan una bola de neu cau muntanya avall, guanya cada cop més i més massa, fent un efecte imparable al llarg del temps. A l'economia dels videojocs MOBAs també se li pot aplicar aquesta metàfora. Aquesta apareix pel fet que els ítems que donen estadístiques al jugador costen or, i aquest or bé donat per les baixes als enemics i destruccions d'estructures principalment. Això fa que, sempre el jugador que vagi per davant a la partida serà qui pugui comprar més i més ítems i fer-se més fort; mentre que el que no ha aconseguit posar-se per davant, cau en fallida.

Això pot arribar a frustrar al jugador que va perdent. Per això s'ha implementat una contra-mecànica per pal·liar aquest efecte el màxim que es pugui i que la partida sigui interessant de principi a final. Aquesta mecànica fa que l'or que dóna l'enemic depengui de la diferència de nivells entre els jugadors. Per tant si el jugador que va per darrere aconsegueix remuntar i acabar eliminant a l'enemic aquest rebrà una gran recompensa i podrà tornar a comprar els ítems necessaris per remuntar la partida i posar-se al mateix nivell que l'enemic. Si la diferència és positiva, vol dir que s'ha de recompensar a l'usuari que va perdent el màxim possible, per tant, la recompensa es multiplica per un factor alt. En canvi, si aquesta és negativa se li aplica una reducció de la recompensa, fins al punt que eliminar a un personatge valgui el mateix que eliminar un *minion*.

5.5 Interfícies del joc

Al videojoc hi ha una sèrie d'interfícies dins i fora de la partida que fan possible la interacció amb l'usuari. Als següents Apartats es mostrarà la totalitat d'aquestes i es descriurà el seu sentit dins del joc.

5.5.1 Interfícies fora de la partida

En aquest Apartat descriurem la llista d'interfícies que resideixen fora de la pròpia partida. Aquestes interfícies són importants perquè són la primera impressió que té el jugador amb el joc.

5.5.1.1 Menú d'entrada al joc

Aquest és el primer menú que el jugador veu a l'entrar al joc (veure Figura 43). Aquest serveix com a impàs per connectar-se al servidor per tal de comprovar connectivitat amb el servidor, entre altres coses. Aquest ens demanarà que cliquem alguna tecla. Quan ho haguem fet, el joc intentarà connectar-se amb el servidor. Si hi ha hagut algun error, es mostrarà en pantalla. Si s'ha aconseguit amb èxit, es passarà al següent menú, explicat a continuació.

Figura 43: Menú d'entrada a Masters Arena

5.5.1.2 Menú principal

Aquesta serà la pantalla on hi haurà el contingut principal del joc. Com es pot veure a la Figura 44, aquesta consistirà de quatre parts principals. A la part superior s'hi podrà veure el nom d'usuari, el nivell del perfil i les monedes de fora del joc. Aquestes monedes serviran per comprar altres personatges. A la part esquerra tenim un fons decoratiu que mostra un dels personatges jugables del joc i que anirà canviant de personatge en futures actualitzacions del videojoc, com podem veure a l'Apartat 10 de treball futur. A la part dreta hi haurà dos

botons: un per unir-se al servidor de *Discord* per seguir el desenvolupament del projecte més de prop i un altre per entrar al tutorial del joc per aprendre les mecàniques bàsiques. Finalment a la part inferior de la pantalla podem veure el botó de Jugar de color daurat. Just a sota hi ha la barra de menús secundaris: El botó per veure la col·lecció de *Masters* adquirits, els ajustos del joc i un botó per poder sortir.

Figura 44: Menu principal de Masters Arena

5.5.1.3 Lobby

Aquesta serà la pantalla que es veurà quan s'estigui dins d'una sala. Aquesta mostrarà un text d'espera mentre la sala no estigui plena. Després es podrà escollir un *Master*, com es pot veure a la Figura 45, i finalment es podrà fixar-lo i començar la partida.

Figura 45: Menu de Lobby de Masters Arena

5.5.1.4 Menú després del joc

Com es pot veure a la Figura 46, en aquesta pantalla es mostrarà tota la informació relativa a la partida que s'acaba de jugar. Per una part es mostrarà si el jugador ha guanyat o ha perdut la partida i el temps que ha durat. Seguidament es mostraran els comptadors de baixes a enemics, *minions* i torretes derruïdes durant la partida. Finalment, es mostraran els personatges que ha jugat cada jugador amb els respectius noms d'usuari. També contindrà un botó per tornar al menú principal.

Figura 46: Pantalla d'estadístiques del final de la partida

5.5.1.5 Pantalla de càrrega

Com es pot veure a la Figura 47, aquesta pantalla es mostrarà al començar la partida mentre el mapa s'està carregant en memòria. En aquesta pantalla mostrarà els noms d'usuari de cada jugador i una imatge i el nom del *Master* que hagin escollit.

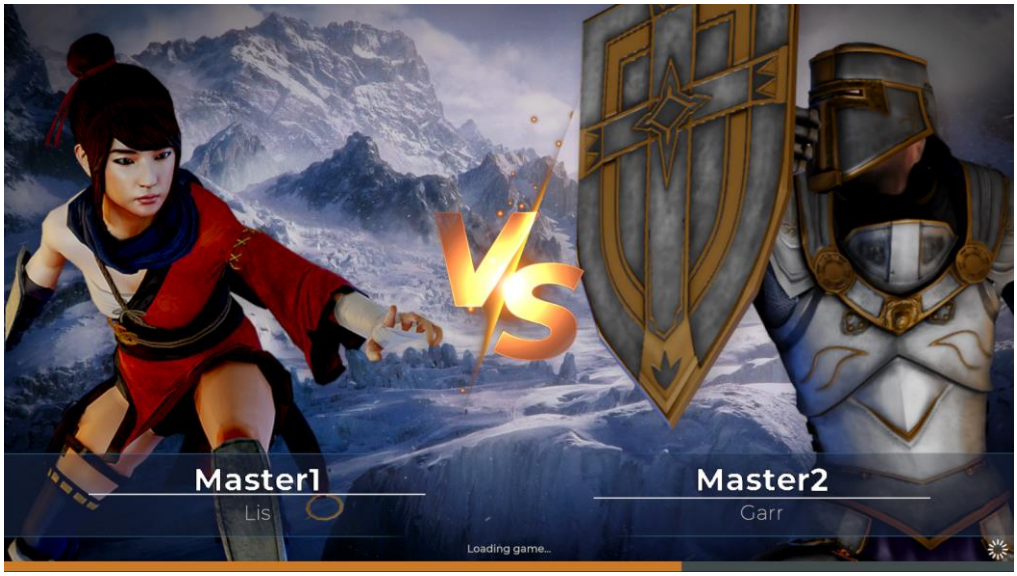

Figura 47: Pantalla de càrrega de Masters Arena

5.5.2 Interfícies dins de la partida

També existeixen diferents interfícies dins de la partida que mostren informació rellevant al jugador. Aquestes es poden dividir en les següents categories:

5.5.2.1 Interfícies diegètiques

Les interfícies diegètiques són aquelles que formen part de l'entorn del joc i que mostren informació útil per al jugador. Dins de la partida hi ha les següents interfícies diegètiques:

- Els personatges reprodueixen diferents animacions respecte l'habilitat que estan fent servir en aquell moment (veure Figura 48). Això informa al jugador local i al jugador enemic de quina habilitat és i poder actuar al respecte.

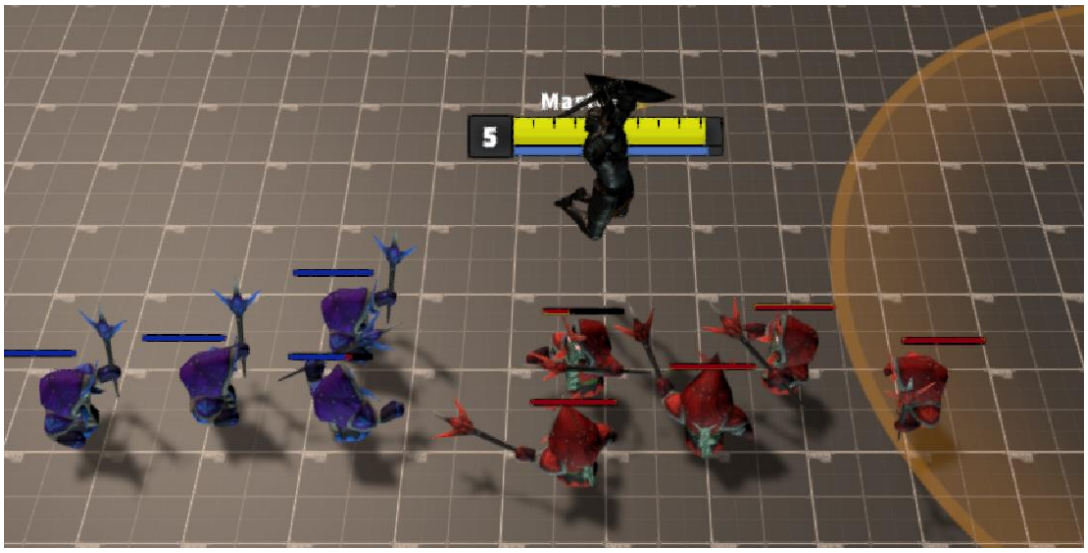

Figura 48: Garr atacant amb Long Sword als minions enemics

- Cada habilitat té una forma diferent, fent més fàcil de distingir entre una habilitat i una altra (veure Figura 49).

Figura 49: Lis activant l'habilitat Hook Kick

- Els colors de les torretes, nexe i *minions* canvia respecte l'equip en el que estiguis jugant. Això vol dir que, si s'està jugant al costat esquerre, les torretes aliades seran blaves i les enemigues seran vermelles, però si som del costat esquerra, els colors es mostraran a l'inrevés (veure Figures 50 i 51).

Figura 50: Demostració del color (equip esquerre)

Figura 51: Demostració del color (equip dret)

- Quan les torretes ataquen, existeix un làser que apunta a l'objectiu actual de la torreta. Aquest làser només indica i no fa mal real (veure Figura 52).

Figura 52: Làser de la torreta

5.5.2.2 Interfícies no diegètiques

Les interfícies no diegètiques són aquelles que no formen part del món però que el jugador les pot veure per damunt de tot. Aquestes interfícies són molt informatives i serveixen per veure un estat general de la partida molt ràpid, però resten immersió al joc.

5.5.2.2.1 HUD del personatge

La HUD del personatge és un gran exemple d'interfície no diegètica ja que mostra la gran majoria de les estadístiques actuals del personatge. Mostra primerament les dues estadístiques principals de vida i manà (actual, màxim i la regeneració). També mostra la icona del personatge actual i el nivell en el que es troba amb la progressió de XP del nivell.

A sota, les habilitats que informen l'estat actual en cada moment. Per cadascuna de les habilitats es mostra la icona i el cost de manà. Si aquesta està en refredament, es veu el text del comptador en segons i una imatge que va desapareixent de forma radial. Si no es té manà o està bloquejada, també surt un indicador.

Al damunt de les barres s'hi mostren els efectes actuals aplicats al jugador. Finalment, a la dreta hi ha l'inventari amb tots els ítems que tenim comprats i l'or actual.

A part de tota la informació que ens dona el panell, podem passar el ratolí per damunt de totes les habilitats, efectes o ítems que tinguem i ens sortirà un panell amb informació addicional formatat amb els colors pertinents per cada tipus d'estadística (veure Figura 53).

Figura 53: HUD del personatge

5.5.2.2.2 HUD de la partida

Figura 54: HUD de la partida

En aquest panell, al centre, es mostra el temps transcorregut en minuts i segons de la partida. També, per cada equip tenim un comptador d'eliminacions de jugadors, de torretes i de *minions* (veure Figura 54).

5.5.2.2.3 Botiga

La botiga ens permet comprar ítems que milloren les estadístiques. Aquesta és una interfície no diegètica ja que no forma part del món i només la pot veure el jugador quan juga i no el personatge.

La botiga es pot visitar en qualsevol moment de la partida i, si es clica amb el botó esquerre del ratolí a qualsevol dels ítems, es mostra més informació a la part inferior d'aquesta.

La botiga també informa de la quantitat d'or que té el jugador en aquell moment. Si un ítem no es pot comprar apareixerà amb un color més fosc per indicar-ho (veure Figura 55).

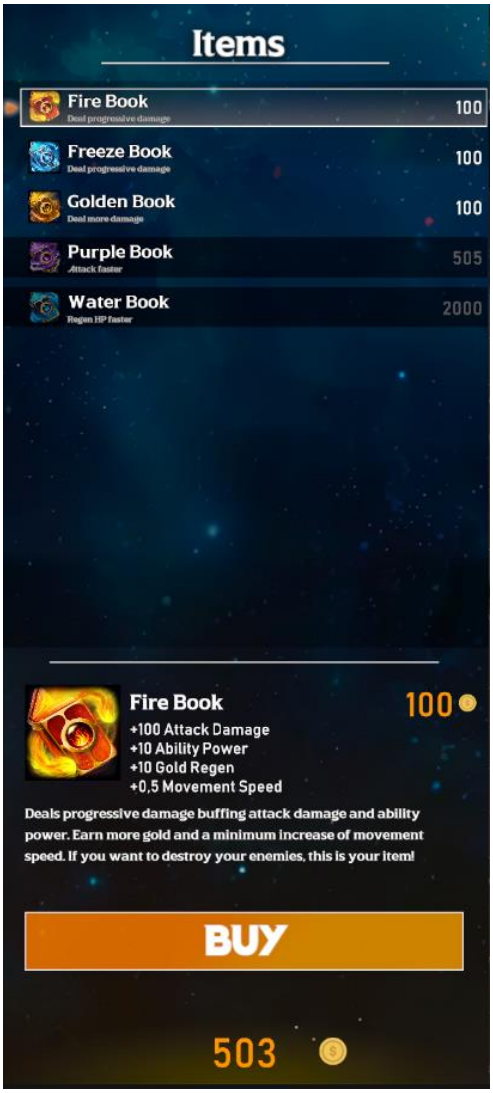

Figura 55: Botiga

5.5.2.3 Interfícies espacials

Les interfícies espacials són aquelles que viuen a l'espai del món del personatge però que aquest no les percep com a informació del món.

5.5.2.3.1 Indicadors de les habilitats

Quan el jugador prem una tecla d'una habilitat que té indicador, aquest es mostra al mapa en la direcció del ratolí. Aquesta és una interfície espacial ja que no existeix cap indicador físic al món que el personatge pugui veure realment, aquest només el veu el jugador però ha d'existir en un context espacial per tal de tenir algun sentit (veure Figura 56).

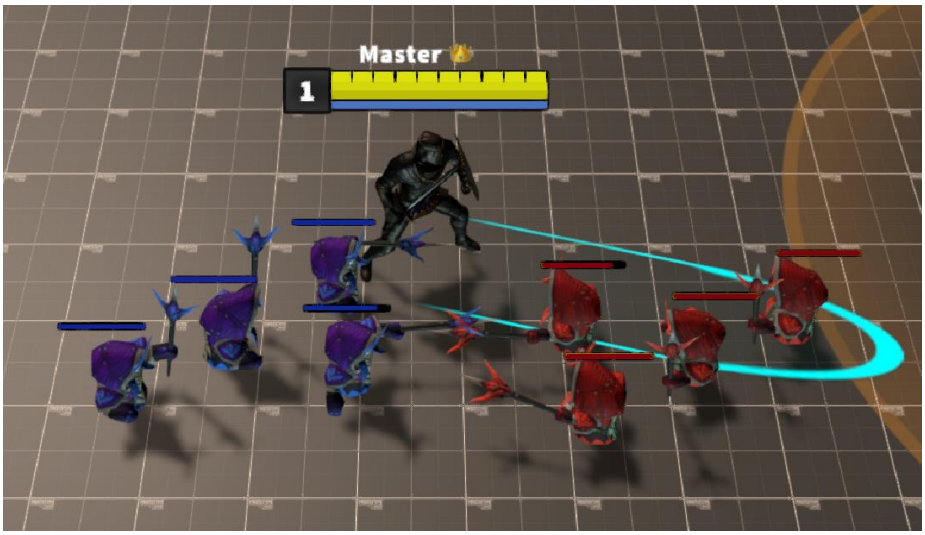

Figura 56: Indicador de Long Sword de Garr

5.5.2.3.2 Barres de vida de les entitats

Cadascuna de les entitats dins del joc tenen una barra de vida just al seu cap. Aquesta barra no és visible en el món virtual del personatge, però si que el jugador la pot veure per tal d'informar-se de quanta vida té aquella entitat en aquell precís moment. Cadascuna de les barres canvia de color respecte si és una entitat enemiga (vermell) o aliada (blau). En el cas del propi personatge, la barra de vida es posa de color groc (veure Figura 57).

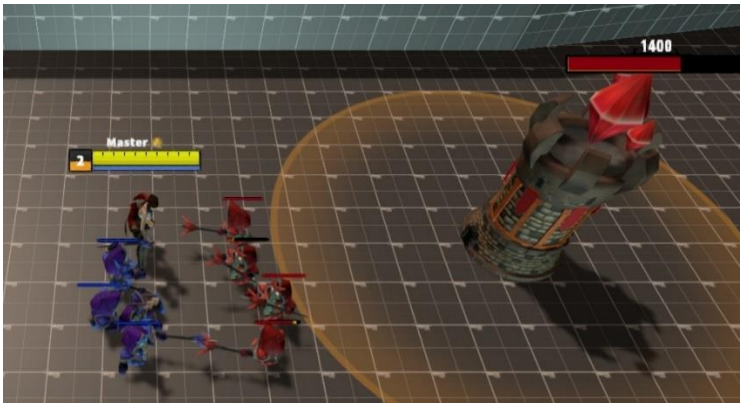

Figura 57: Barres de vida de les entitats

La barra de vida de cada personatge, a diferència de la resta d'entitats, conté més informació a part de la quantitat de vida (veure Figura 58). Primerament, el nom d'usuari apareix a damunt de tot. També es mostra una corona al costat del nom si es tracta del creador d'aquesta. Es mostra el número del nivell i la progressió d'aquest en XP de baix a dalt. Apareix la vida màxima indicada amb una sèrie de línies repetides on cada línia representa 100 de vida. Finalment a sota de la vida apareix el mana.

Figura 58: Barra d'informació del jugador

5.5.2.3.3 Indicadors de la torreta

Quan el jugador s'apropa a alguna de les torretes enemigues aquestes comencen a mostrar un indicador (veure Figura 59). La opacitat d'aquest depèn de la distància al jugador dins d'uns marges. L'indicador de la torreta funciona de la següent manera:

- Si la torreta no està atacant a cap entitat, el cercle es mostrarà taronja indicant que no hi ha perill imminent, però si el personatge entra en rang, la torreta l'atacarà.
- Si la torreta està atacant a alguna entitat:
    - Si aquesta entitat és el jugador, l'indicador es posa de color vermell indicant que és urgent que aquest surti del rang si no vol morir.
    - Si aquesta entitat no és el jugador, l'indicador es posa de color verd indicant que és segur entrar dins del rang de la torreta ja que aquesta està atacant a una altra entitat.

Figura 59: Tipus d'indicadors de la torreta

5.5.2.4 Interfícies meta

A les barres de vida i manà de totes les entitats hi ha una barra secundaria que fa la transició de forma suau i amb un color accentuat. Aquesta barra serveix per donar la impressió de que s'ha pegat un cop molt fort mantenint la informació de la barra de vida actualitzada instantàniament per damunt.

Totes les interfícies meta que no s'han acabat implementant es troben a l'Apartat de *Treball Futur*. Aquestes interfícies generen molt de feedback al jugador i, per tant, fan que el joc sigui més agradable de jugar i cada cop tingui més impacte, fent que el jugador es senti més còmode i recompensat cada cop que interacciona amb el joc.

5.6 Narrativa del joc

El joc es basa en una arena virtual de proves. Aquesta està simulada per un súper-ordinador quàntic. Aquest ordinador està controlat per un equip de científics (la ALU[29], sigles per AI Lead Union) que pretenen dominar el món creant la millor criatura a partir d'intel·ligència artificial. Aquestes criatures són els personatges jugables del videojoc i es denominen *Masters*. El jugador representa la intel·ligència artificial que controla a cadascun dels personatges i aquest s'ha de fer amb la victòria per poder sobreviure al bucle infinit de partides que aquest pateix dins de l'execució d'aquesta màquina. La màquina anirà generant procedimentalment personatges de la seva imaginació. Els dos que ha anat generant fins ara són Lis i Garr, els quals protagonitzen el videojoc. Tots els personatges es creuen que són algú i recorden que han tingut una vida, però realment són ratolins de laboratori que només es componen de dígits binaris.

Lis recorda que és una *ninja* que ha passat tota la seva vida entrenant per aquest moment. La seva mare, *Mochizuki Hanzo*[30] era una mercenària *shinobi* que treballava per la elit criminal japonesa. *Mochizuki* va morir a la guerra contra els *samurais* poc després de que ella nasqués. Des de petita, ella s'ha anat desenvolupant individualment i no ha volgut tenir amics, diu que no fan falta per sobreviure en aquest món de covards. Sempre li ha interessat participar activament en la resolució agressiva dels conflictes. Encara així, a Lis li agrada la pau; sempre que la beneficiï d'alguna manera. Fa servir els seus enemics i els saboteja per treure la màxima informació possible d'ells, finalment els assassina sense deixar rastre algun. Un cor cruel i sense pietat és el que domina a Lis i que la mou amb instints més primitius. Més val no buscar problemes amb ella si no es vol estar entre la seva llista de pròxims desapareguts.

[29] Les sigles ALU són una referència a la unitat aritmeticològica que qualsevol CPU té al seu interior.

[30] El nom de Mochizuki Hanzo és una referència a dos noms molt famosos de la cultura ninja. Per una part, Mochizuki Chiyome va ser una líder d'un clan de ninges completament femení conegudes com a *kunoichi* al voltant del 1570. Per altra banda tenim el cognom Hanzo que ve de Hattori Hanzo. La família de Hattori era de la classe samurai, però ell, per contrapart dirigia un grup de ninges local al voltant del 1580.

Garr és un guerrer armat fins les dents. Ve d'una família humil i treballadora de ferrers. Quan era petit, Garr es va cremar forjant una armadura i va anar a veure la seva tieta maga, Glinda[31]. Ella va veure que el foc no era normal, sinó que era màgic. En aquelles terres només ella coneixia el què era la màgia. Com podia ser que algú que no fos ella controlés la màgia? Vista la situació, Glinda va anar a buscar dins del seu llibre de conjurs per reparar l'armadura de Garr. Just en el moment que Glinda va a llençar un conjur per reparar la ferida, la seva ment es va emplenar de pensaments negatius i tòxics. Va pensar en qui pot ser la maleïda persona que li pugues destronar el poder de dominar la màgia. En aquest precís moment Glinda va canviar de conjur. La por va recórrer el seu cos mentre el nebot no entenia res del què estava passant. Per protegir-lo, el conjur que va escollir la maga va fer que Garr quedés vinculat per sempre a l'armadura impenetrable. Garr es va intentar treure l'armadura instantàniament. Desconcertat, va mirar a la seva tieta. En aquell moment, Glinda es va apropar al seu nebot i li va dir amb un somriure: "Ara ja ningú et podrà fer mai més cap mal".

5.7 Nom del joc

El nom del joc és "Masters Arena", traduït al català seria "La Arena dels Mestres". A l'Apartat anterior hem vist que els Masters són els personatges generats per la imaginació de la intel·ligència artificial del super-ordinador. Finalment, "Arena" descriu la Arena o camp de batalla on combaten entre ells per proclamar-se guanyadors. En conjunt, el nom del joc fa referència a la simulació de la partida al super-ordinador.

Figura 60: Logotip (fosc) del videojoc

El logotip manté una estètica minimalista i futurista. Té una font amb vores punxegudes per donar la sensació de combat i acció. La iconografia de la forma es basa en la idea de tractar la "M" de "Masters" i transformar-la en una corona, expressant així que només un dels dos Masters que competeixin serà finalment coronat guanyador.

5.8 Controls del jugador

El jugador podrà gaudir del videojoc amb teclat i ratolí. Aquests es combinaran per generar una experiència el més immersiva possible.

5.8.1 Moviment del personatge

El moviment del personatge serà amb els botons WASD del teclat per moure's en la direcció desitjada respecte a la càmera.

[31] El nom de Glinda és una referència a la maga bona de "The Wizard of Oz".

5.8.2 Esprint

El personatge té una capacitat d'esprintar si es manté clicat el Shift esquerre del teclat. Si s'està tirant una habilitat, aquest no funciona. Si s'està atacant amb l'atac bàsic, aquest cancel·la l'animació.

5.8.3 Habilitats

Les habilitats es llençaran amb els números del teclat alfanumèric. Els quatre primers números per les quatre habilitats que té cada personatge. També es pot cancel·lar qualsevol habilitat amb indicador si, mentre s'està llençant, es clica el botó dret del ratolí.

5.8.4 Indicadors

Els indicadors surten quan es manté el botó de l'habilitat just abans de deixar-lo i activar-la. Els controls en aquest aspecte són els mateixos que els de l'apartat anterior.

5.8.5 Mostrar rang d'atac del personatge

Existeix un indicador manual per veure el rang d'atac del personatge. Aquest es pot activar mantenint la tecla Q del teclat.

5.8.6 Obrir la botiga

Per obrir la botiga cal clicar la tecla B del teclat. Aquesta tecla també funciona per tancar-la.

5.8.7 Atac bàsic

Si es clica amb el botó dret del ratolí apuntant a una entitat enemiga en rang, el personatge podrà atacar-la fent servir la mecànica de l'atac bàsic. Si es torna a clicar amb el mateix botó l'entitat mentre s'ataca, aquest recordarà que ha de tornar a atacar.

5.8.8 Tooltips de la UI

Als elements dinàmics de la UI dins del joc, si es passa el ratolí per damunt, aquest mostrarà certa informació rellevant que roman oculta per no ocupar espai en pantalla.

5.8.9 Botons de la UI

Tots els botons de la UI es poden clicar amb el botó esquerre del ratolí.

5.8.10 Sortir del joc

Per sortir del joc l'usuari cal que cliqui la tecla Escape del teclat. Allà sortirà un menú contextual d'opcions. A sota d'aquest hi ha un botó per desconnectar-se de la partida.

5.9 Flux del jugador al videojoc

Al diagrama de la dreta (veure Figura 61) es pot veure quin és el flux de les escenes a l'hora de jugar al videojoc. Cadascuna de les interfícies d'aquestes escenes s'ha explicat a l'Apartat 5.5.1.

Primerament es comença el joc a la pantalla d'inici. Aquesta comprova la connectivitat amb el servidor. Si és correcta es passa al menú principal.

Al menú principal es pot veure diversa informació del compte de l'usuari. Aquesta informació serà diferent per cada usuari que jugui al joc. Del menú principal es podrà crear una partida nova o unir-se a una partida ja existent. Quan aquesta connexió amb la sala es compleixi i la sala estigui plena, es començarà la partida a l'escena del joc.

Finalment, quan el joc acabi, es mostraran les estadístiques de la partida a la escena de Post Game, la qual tindrà un botó per tornar al menú principal i repetir el bucle.

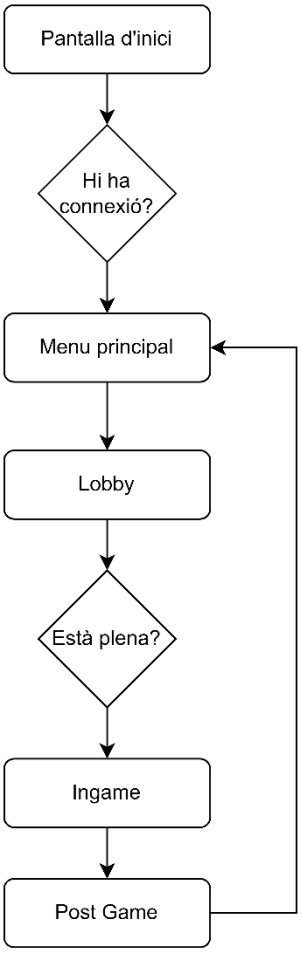

Figura 61: Diagrama de flux de les escenes

6 Implementació

En aquest Apartat s'explicarà com s'han implementat les principals mecàniques descrites a l'apartat anterior.

6.1 Com funciona Photon PUN 2?

Photon és un sistema compatible amb l'SDK[32] de Unity per fer videojocs multijugador. Aquest sistema té diferents variants que depenen del tipus de joc i els requeriments. En aquest projecte es farà servir el sistema PUN (la seva segona versió). Aquest sistema és una variant generalista que suporta de forma estable qualsevol tipus de joc fent servir un sistema client-servidor. El servidor és una estació física a Frankfurt propietat de Photon. Cada aplicació, dins del rang gratuït, suporta fins a 20 jugadors concurrents.

6.2 Funcionalitats bàsiques d'un videojoc en línia

6.2.1 Connexió al servidor

Cada usuari, per connectar-se, cal que comprovi si té connexió a internet. Si la té s'intenta connectar al servidor utilitzant la configuració per defecte (veure Figura 62).

```
public void RetryConnection()
{
    errorPopUp.SetActive(false);
    if (Application.internetReachability == NetworkReachability.NotReachable)
    {
        ShowError("It seems you don't have connection to the Internet.");
    }
    else
    {
        PhotonNetwork.ConnectUsingSettings();
    }
}
```

Figura 62: Funció de connexió amb el servidor de Photon

Quan s'executa la funció, Photon comença a intentar posar-se en contacte amb el servidor. Si la connexió és correcta s'executa un *callback*[33] de Photon (veure Figures 63 i 64). Per tal de que funcionin aquestes funcions, cal que la classe hereti de *MonoBehaviourPunCallbacks*.

```
public override void OnConnectedToMaster()
{
    SceneController.LoadScene(Scenes.MAIN_MENU);
}
```

Figura 63: Callback de connexió correcta

[32] SDK: Equip de desenvolupament de programari. És un conjunt d'eines que ajuda al desenvolupador a crear aplicacions per un programa concret.

[33] Funció que es crida automàticament quan una acció concreta passa.

Si hi ha hagut algun error, es mostra una pantalla d'error amb la informació retornada pel servidor.

```
public override void OnErrorInfo(ErrorInfo errorInfo)
{
    ShowError(errorInfo.Info);
}
```

Figura 64: Callback d'error

6.2.2 Creació de sales

El joc funcionarà amb un sistema de sales que s'aniran creant i destruint mentre els jugadors van entrant i sortint de les partides. Cadascuna de les sales mantindrà una partida i com a màxim hi podrà mantenir dos jugadors. Un jugador haurà de crear la sala i l'altre haurà d'unir-s'hi.

```
public void CreateRoom()
{
    string nick = PlayerPrefs.GetString("NickName") ?? "Master001";
    if (roomNameInput.text == "")
    {
        errorTexts[0].text = "Error";
        errorTexts[1].text = "You need to enter a room name!";
        errorPopup.SetActive(true);
    }
    else if (string.IsNullOrEmpty(nick))···
    else
    {
        RoomOptions roomOptions = new RoomOptions
        {
            MaxPlayers = maxPlayers,
            PublishUserId = true
        };
        gameSettingsPanel.SetActive(false);
        loadingPanel.SetActive(true);
        PhotonNetwork.NickName = nick;
        PhotonNetwork.CreateRoom(roomNameInput.text, roomOptions);
    }
}
```

Figura 65: Creació d'una sala

Com es pot veure a la Figura 65, per crear la sala cal posar-li un nom. Aquest l'escollirà el creador de la partida. Si aquest nom ja està agafat per una altra partida, es mostrarà un error.

6.2.3 Entrada a les sales

Al crear la partida, l'altre jugador haurà de connectar-se a aquesta. Per unir-se a una sala cal posar el nom de la sala que ha escollit el creador (veure Figura 66).

```
public void JoinRoom()
{
    string nick = PlayerPrefs.GetString("NickName") ?? "Master001";
    if (roomNameJoinInput.text == "")
    {
        errorTexts[0].text = "Error";
        errorTexts[1].text = "You need to enter a room name!";
        errorPopup.SetActive(true);
    }
    else if (string.IsNullOrEmpty(nick))···
    else
    {
        gameSettingsPanel.SetActive(false);
        loadingPanel.SetActive(true);
        PhotonNetwork.NickName = nick;
        PhotonNetwork.JoinRoom(roomNameJoinInput.text);
    }
}
```

Figura 66: Entrada a sales

6.2.4 Obtenció d'informació de cada jugador

Quan els dos jugadors estan dins de la partida, es pot accedir a les propietats de cadascun d'ells per índex. Aquesta llista està ordenada per ordre d'entrada a la sala. Així és molt fàcil saber qui és el creador de la partida i qui és el visitant.

```
masterNickTxt.text = PhotonNetwork.PlayerList[0].NickName;
```

Figura 67: Trobar el nom d'usuari del creador

La propietat PlayerList retorna una llista de jugadors, només si el jugador està dins d'una sala (veure Figura 67).

6.2.5 Sincronització els transforms entre els clients

Per sincronitzar la orientació i posició de cada objecte que es mogui a la partida, cal utilitzar el component *Photon Transform View Classic* (veure Figura 68). Aquest component es pot configurar per sincronitzar la posició, rotació i/o escala.

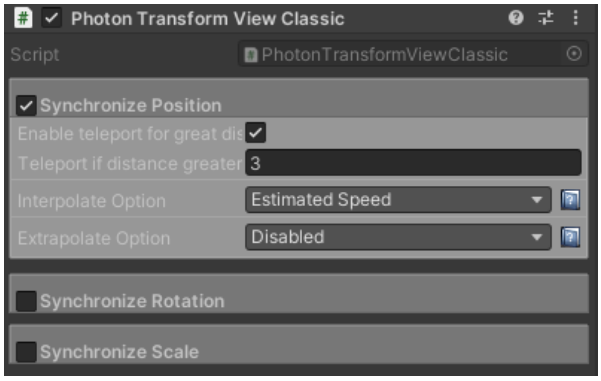

Figura 68: Component que sincronitza el transform

Aquest script ja suavitza el moviment de les entitats encara que internament estigui funcionant amb un *tickrate*[34] baix. El sistema de Photon funciona amb una freqüència de 50 missatges per segon encara que es pot augmentar amb un nivell més alt de pagament del sistema si el joc ho requereix. Quanta menys informació passi, menys ocuparà el paquet i més ràpid podrà anar el joc. Per tant, és important només enviar la informació necessària pels clients. En aquest cas hem desactivat l'enviament de la rotació i la escala ja que no es fan servir a l'altre client.

6.2.6 Crida de funcions a tots els usuaris

Imaginem que tenim dos jugadors a la partida i volem fer que l'usuari 1 ataqui amb una habilitat, i per tant que es vegi a l'execució local i a la respectiva rèplica a l'altre usuari. En aquest cas es pot fer servir un tipus de funció especial que es diu RPC[35].

Aquesta funció serà executada a totes les rèpliques de la instància a la sala. Aquest tipus de funció va molt bé a l'hora de sincronitzar comportaments ja que és la pròpia funció que s'executa a l'altra màquina i no només se li poden passar valors serialitzables[36]. Aquests s'enviaran amb la finalitat de sincronitzar el comportament de tots els clients dins de la partida.

A la Figura 69 podem veure com es crida la funció RPC de *DealDamage*. Aquesta funció es crida cada cop que qualsevol entitat rep mal.

```
public void DealDamage(int physicalDamage, int magicalDamage, int trueDamage, Inventory who = null) {
    view.RPC("DealDamageRPC", RpcTarget.AllBuffered, physicalDamage, magicalDamage, trueDamage, who?.gameObject.name);
}
```

Figura 69: Execució d'una funció RPC

Es pot observar com la funció rep dos paràmetres obligatoris i després la resta d'opcionals. Els dos primers paràmetres són el nom de la funció RPC que s'ha de cridar i el *Target* o objectiu. Aquest últim defineix a qui se li ha d'enviar aquesta crida de funció. En aquest cas ens interessa enviar-la a tots els jugadors de la sala, inclòs l'autor de la crida que se l'envia a si mateix. Els paràmetres que es posin a partir d'ara seran els paràmetres que hauran de coincidir amb els d'entrada de la funció RPC i hauran de ser serialitzables.

[34] *Tickrate*: Tassa de missatges per servidor enviats al servidor per tal d'actualitzar l'estat de la partida.

[35] Les funciona RPC (de l'anglès *Remote Procedure Call,* crida a procediment remot) són un tipus de funcions especials que, en ves d'executar-se a la instància del dispositiu local, també s'executen a totes les repliques dels altres dispositius de la sala.

[36] Un valor serialitzable és un tipus de valor que es pot passar a binari i serà independent de la màquina a la que es serialitzi. Per exemple, un numero enter o una llista de *strings* són serialitzables, però una referència a un *GameObject* no, ja que aquesta depèn de la màquina a la qual s'està executant.

```
[PunRPC]
1 reference
protected virtual void DealDamageRPC(int physicalDamage, int magicalDamage, int trueDamage, string who) {
    if (isDead) return;

    if (who ≠ null) {···

    HP = Mathf.Max(0, HP - DamageReduction(physicalDamage, magicalDamage, trueDamage));
    if (HP == 0) {
        Die();
    } else {
        GetUI().UpdateCurrentHP(HP, maxHP);
    }
}
```

Figura 70: Funció RPC de DealDamage

En aquest cas, aquesta funció (veure Figura 70) calcula el mal que s'ha de restar aplicant les reduccions per resistència a cada tipus de mal i el resta. Si aquesta quantitat arriba a zero, l'entitat mor. Aquesta funció es crida a cada client que està dins de la sala, per tant es manté la coherència entre clients. Per marcar que una funció pot ser cridada des de Photon fent servir el sistema de RPCs, cal marcar-la amb l'atribut "PunRPC". Això afegeix la funció a la llista de RPCs que manté Photon amb una diccionari d'accés constant, i fa que les crides siguin el més eficients i ràpides possible.

6.2.7 Sincronització d'escenes entre usuaris de la mateixa sala

Per a què els usuaris de la mateixa sala es mantinguin a les mateixes escenes de Unity, cal fer servir el SceneManager de forma diferent. En ves de dir-li a Unity directament, cal avisar a Photon per a què notifiqui als altres clients que han de canviar d'escena (veure Figura 71).

```
0 references
public void StartGame()
{
    startGameBtn.interactable = false;
    view.RPC("ShowLoadingScreenRPC", RpcTarget.AllBuffered);
}

[PunRPC]
0 references
private void ShowLoadingScreenRPC()
{
    SceneController.Instance.LoadGame();
    Debug.Log("Carregant la partida ... ");
}
```

Figura 71: Entrar a la partida

En aquest cas, avisem de forma manual a tots els clients de la sala amb una funció RPC per tal de que canviïn d'escena (veure Figura 72).

```
/// <summary>
/// This will load the game scene in the background showing the loading screen.
/// </summary>
1 reference
public void LoadGame()
{
    loadingScreen.gameObject.SetActive(true);
    SceneManager.UnloadSceneAsync((int)current);
    current = Scenes.INGAME;
    loadingProcess = SceneManager.LoadSceneAsync((int)Scenes.INGAME, LoadSceneMode.Additive);
    StartCoroutine(SceneProgress()); // Show the current progress in the loading bar
}
```

Figura 72: Carregar escena asíncronament

La classe *SceneController* és un *manager* que s'encarregarà de carregar totes les escenes de forma asíncrona per mantenir la fluïdesa en aquest tipus de transicions. En el cas especial d'entrar a la partida, es mostra una pantalla de càrrega que s'afegeix de forma additiva, fent que hi hagi més d'una escena al mateix temps. Una escena mantindrà la pantalla de càrrega i el joc romandrà ocult mentre s'està carregant tots els objectes i tots els usuaris estan sincronitzant la partida.

6.3 Mecàniques de la partida

En aquest Apartat s'explicaran en profunditat les mecàniques més importants de la partida.

6.3.1 Sistema de personatges dinàmic

Un dels primers sistemes que s'ha implementat al joc és el dels personatges dinàmics. Aquest permet jugar al joc podent escollir el personatge i que aquest variï la seva aparença, les seves estadístiques bàsiques, a més de tot el kit d'habilitats.

6.3.1.1 Instanciar els personatges a la escena

Al principi de la partida, mentre els clients van entrant a la partida, aquests van instanciant el seu propi personatge. Quan estan tots instanciats, s'inicialitzen i, al acabar la configuració de tota la partida, aquesta comença.

```
PhotonNetwork.Instantiate("Player/Player", Vector3.zero, Quaternion.identity);
```

Figura 73: Instancia del prefab[37] del jugador

Com es pot veure a la Figura 73, l'script *GameController* (que cada client té a la seva escena) instancia de forma compartida el seu propi *GameObject* del jugador. Aquest *prefab* (veure Figura 74) és genèric i no conté cap model de personatge. El model s'instancia com a fill d'aquest *prefab* genèric a la següent etapa d'inicialització.

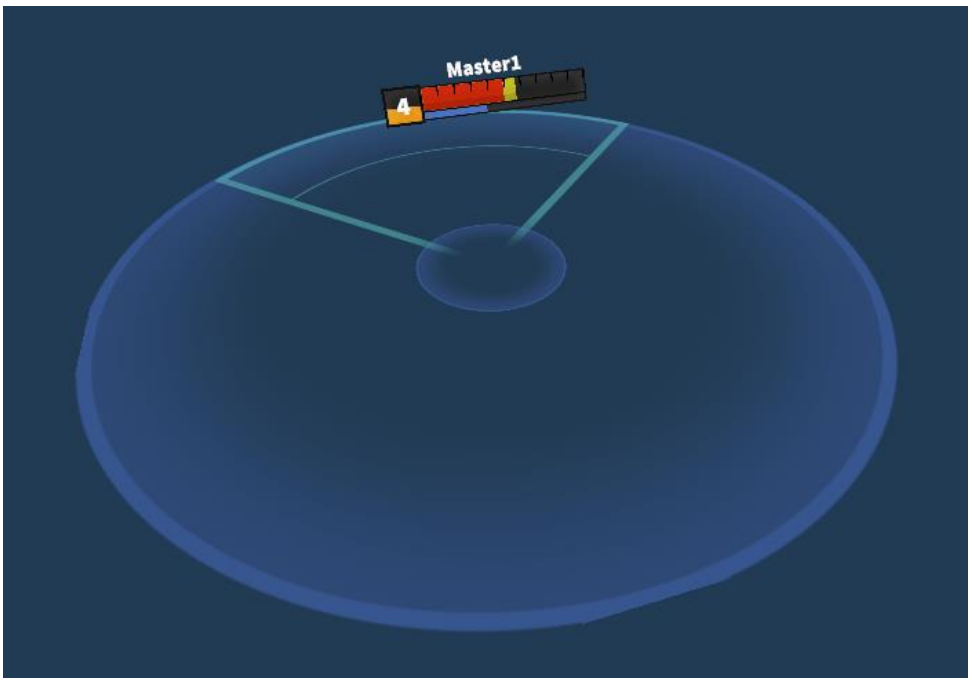

Figura 74: Prefab genèric del personatge

[37] Un *prefab* és un objecte prefabricat. Aquest determina el comportament d'un objecte de forma encapsulada.

6.3.1.2 Inicialització de cada personatge

Per cadascun dels personatges s'executa una etapa d'inicialització que munta tota la configuració inicial. Existeix l'script *ModelSync* que s'encarrega de tota la cadena d'inicialització de cada personatge. Aquest script viu al *prefab* genèric de Player. Quan el *prefab* s'instancia, aquest script s'encarrega d'emparentar el model del personatge amb el *prefab* del jugador i inicialitzar el *PlayerController* entre altres scripts, com es pot veure a la Figura 75.

```
/// <summary>
/// This starts all the initialization chain for the player.
/// </summary>
1 reference
private void InitializeGame()
{
    PlayerController playerController = playerGO.GetComponent<PlayerController>();
    BasicAttack basicAttack = playerGO.GetComponent<BasicAttack>();

    playerController.SetParent(gameObject);
    playerController.Initialize(anim, rangedInstancePivot);
    mediator.Initialize(basicAttack);

    Debug.Log("==== ALL SETINGS DONE ====");
    enabled = false;
}
```

Figura 75: Funció d'inicialització del personatge

Com es pot veure a la Figura 76, el *PlayerController* és l'encarregat (a la etapa d'inicialització) d'inicialitzar l'atac bàsic, les estadístiques base del jugador i d'obtenir les seves habilitats. Aquestes habilitats passen a l'script *PlayerStats* que descriu totes les estadístiques del personatge. Aquest inicialitza les interfícies, tant les barres dalt del cap del personatge com l'HUD. També inicialitza el sistema d'efectes, inventari, totes les habilitats i carrega tots els valors inicials pel personatge escollit.

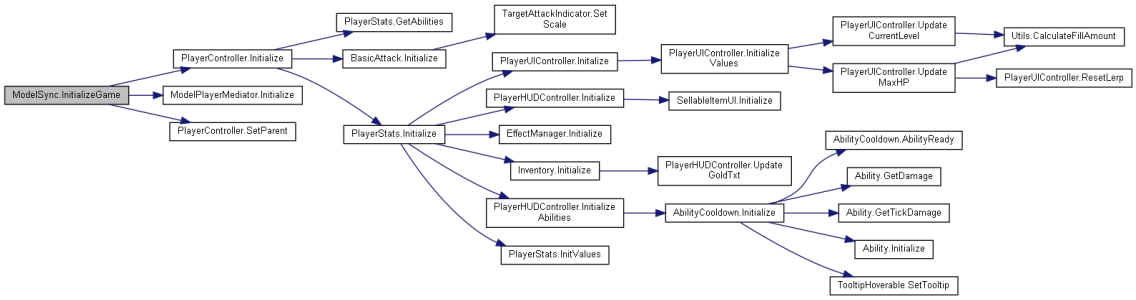

Figura 76: Gràfic de crida de funcions d'inicialització

Després de fer aquesta inicialització, l'script *ModelSync* es desactiva per no fer servir recursos del joc, ja que aquest només es fa servir a la etapa d'inicialització.

6.3.1.3 Sincronitzar els estats i animacions

Per tal de sincronitzar les animacions entre clients, cal simular que la còpia del client remot al dispositiu local. Posem d'exemple l'*animator*[38] del personatge Lis, com es pot veure a la Figura 77.

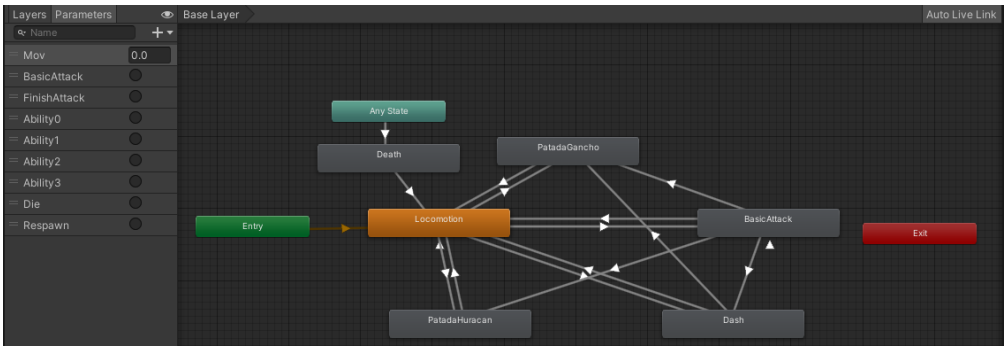

Figura 77: Animator de Lis

Pels paràmetres numèrics com la quantitat de moviment, aquests es poden enviar als altres clients amb el component PhotonAnimatorView (l'homònim al PhotonTransformViewClassic vist a la Figura 68), com es pot veure a la Figura 78.

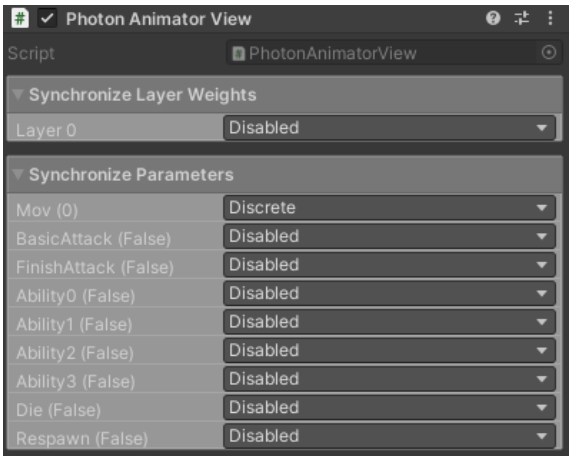

Figura 78: Component Photon Animator View

Lamentablement els *triggers*[39] no es poden enviar per aquest component, però sí que es poden fer servir funcions RPC que essencialment són el mateix, com es pot veure a la Figura 79.

```
[PunRPC]
0 references
private void StartBasicAttackRPC()
{
    anim.speed = stats.attackSpeed;
    targetIndicator.SetTarget(otherStats);
    anim.SetTrigger("BasicAttack");
}
```

Figura 79: RPC sincronitzant l'animació d'atac bàsic

[38] L'*animator* és un component de Unity que s'encarrega de mantenir una màquina d'estats per reproduir una animació què depèn de l'estat del jugador.

[39] Un *trigger* és un disparador o activador d'algun event.

6.3.2 Tipus d'entitats

Qualsevol tipus d'entitat al joc tindrà algun tipus derivat a la classe *Stats*. Aquesta classe implementa el comportament de totes les estadístiques per cada tipus d'entitat. Com podem veure a la Figura 80, hi ha dos tipus d'entitats: les entitats que es poden moure i les estructures. Dins del tipus mòbils existeixen dos tipus: els *minions* i els jugadors. La classe *MannequinStats* és una derivació del jugador que serveix només per fer proves del joc de manera més ràpida (veure Apartat 7.2). A part, hi ha dos tipus d'estructures: el nexe i les torretes.

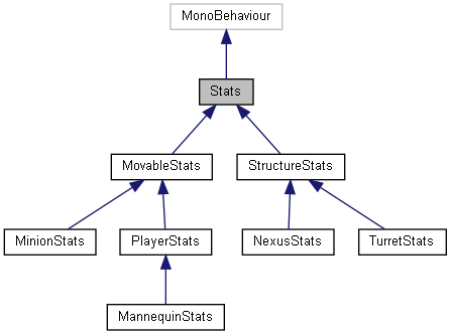

Figura 80: Arbre de classes per tots els tipus d'entitats

Cada entitat haurà d'implementar, com a mínim, les següents funcions:

- *DealDamage*: com rep mal l'entitat?
- *Die*: què passa quan l'entitat mor?
- *GiveRewards*: com es recompensa a l'autor de la eliminació?

6.3.3 Fer mal a altres entitats

Cada cop que s'hagi de danyar a una entitat cal distingir entre tres tipus de mal:

- **Mal físic**: Mal que fa l'atac bàsic dels jugadors i *minions*
- **Mal màgic**: Mal que fan les habilitats
- **Mal verdader**: Mal que fan les torretes (no s'aplica cap reducció de mal)

Aquests tres mals es poden enviar en qualsevol combinació de quantitats. Cada tipus de mal té un tipus de resistència, que són els següents:

- Mal físic → **Armadura**
- Mal màgic → **Resistència màgica**
- Mal verdader → (No en té, per definició)

Per tal d'implementar aquest comportament, s'ha de filtrar el mal arribat des de l'enemic per parts i aplicar-li una reducció respecte a la quantitat de resistència de cada mal.

Tal i com es veu a la Figura 81, es resta a la vida actual un filtre del mal original.

```
/// <summary>
/// Specify the amount of damage and the Inventory that the amount of Gold will go in if it is hit for the last time
/// </summary>
/// <param name="physicalDamage">The amount of Physical Damage dealt</param>
/// <param name="magicalDamage">The amount of Magical Damage dealt</param>
/// <param name="trueDamage">The amount of True Damage dealt</param>
/// <param name="who">Inventory from who is attacking. Optional for the non-player entities.</param>
9 references
public void DealDamage(int physicalDamage, int magicalDamage, int trueDamage, Inventory who = null) {
    view.RPC("DealDamageRPC", RpcTarget.AllBuffered, physicalDamage, magicalDamage, trueDamage, who?.gameObject.name);
}

[PunRPC]
1 reference
protected virtual void DealDamageRPC(int physicalDamage, int magicalDamage, int trueDamage, string who) {
    if (isDead) return;

    if (who != null) {...

    HP = Mathf.Max(0, HP - DamageReduction(physicalDamage, magicalDamage, trueDamage));
    if (HP == 0) {
        Die();
    } else {
        GetUI().UpdateCurrentHP(HP, maxHP);
    }
}
```

Figura 81: Funció de fer mal

Aquest filtre s'ha implementat fent us d'una funció asimptòtica per no provocar divisions per zero ni reduccions tan grans que regeneressin vida al jugador. Aquesta funció és la següent, de la qual es pot veure la seva representació gràfica a la Figura 82:

$$F_p = d_p \frac{100}{100 + r_f}$$

$$F_m = d_m \frac{100}{100 + r_a}$$

$$F_t = F_p + F_m + F_t$$

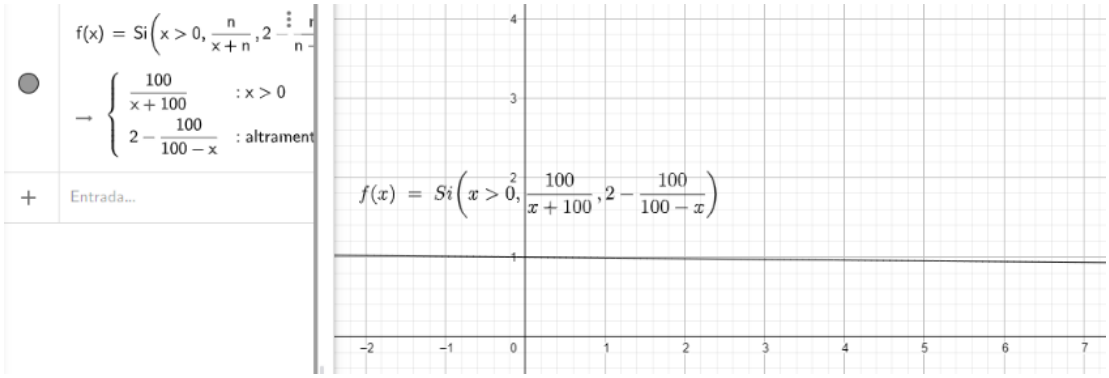

Figura 82: Gràfic de l'asímptota

A les anteriors fórmules, d_p és el mal físic que rep l'usuari abans de ser filtrat i r_f la resistència física. A continuació, F_p és el mal físic filtrat. El mateix passa amb el mal màgic. Finalment, es sumen les tres magnituds de mal (el mal filtrat físic F_p, el mal filtrat màgic F_mi el mal verdader F_t) i tenim el mal final que l'entitat rebrà. A la Figura 83 es pot veure el tros de codi que ho implementa.

```
/// <summary>
/// Apply the corresponding damage to this entity according to the armor and magicResist it have
/// </summary>
/// <param name="physicalDamage">The amount of Physical Damage dealt</param>
/// <param name="magicalDamage">The amount of Magical Damage dealt</param>
/// <param name="trueDamage">The amount of True Damage dealt</param>
/// <returns>The reduced damage amount</returns>
1 reference
protected int DamageReduction(int physicalDamage, int magicalDamage, int trueDamage) {
    float total = trueDamage;
    total += (float) physicalDamage * (100 / 100 + (float) armor);
    total += (float) magicalDamage * (100 / 100 + (float) magicResist);
    return (int) total;
}
```

Figura 83: Funció de reduir el mal d'entrada

6.3.4 Atac bàsic

L'atac bàsic és un tipus d'atac que tots els personatges tenen. Com es pot veure a la Figura 84, aquest funcionarà de la següent manera:

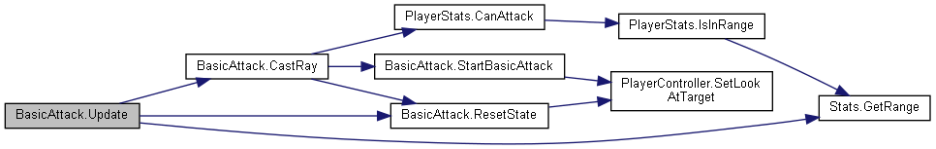

Figura 84: Gràfic de crides de funcions de l'atac bàsic

- Si el jugador clica el botó d'atacar
    - Es llençarà un raig des de la càmera fins a la posició física del ratolí al món 3D.
    - Si aquest col·lideix amb una entitat, és enemiga i està dins del rang d'atac
        - Llavors, s'executa l'animació d'atac.
- Mentre s'està atacant, si el personatge surt del rang, la animació es cancel·la.

Com es pot veure a la Figura 85, l'animació d'atac de cada personatge té un event que indica quan s'ha d'aplicar el mal.

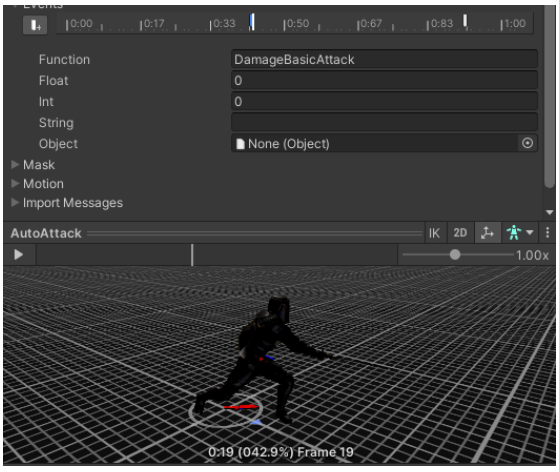

Figura 85: Event d'atac de l'animació d'atac bàsic de Garr

Hi ha dos tipus d'entitats que ataquen de forma diferent:

- **Els personatges cos a cos**: Aquest event aplica el mal directament.
- **Els personatges a distància**: Aquest event instancia una bala (amb un aspecte diferent per cada personatge) que viatja cap a l'enemic i aplica mal l'arribar.

També hi ha un segon event que reinicia l'atac, per poder executar l'animació si així ho desitja l'usuari. La funció lligada a aquest event també es crida quan es cancel·la l'animació d'atac corrents.

6.3.5 Sistema d'habilitats dinàmiques

Cada personatge tindrà un total de quatre habilitats. Aquestes habilitats funcionen amb una combinació de tres scripts funcionant al mateix temps.

6.3.5.1 Ability

Aquest script s'encarrega d'identificar cada habilitat i guardar-la de forma persistent al disc. Per tal de que el joc tingui un accés ràpid a tota la informació, es fan servir els *ScriptableObjects*. Aquests són un tipus de classe molt útil que permet crear instàncies en fitxers per poder configurar el videojoc sense perdre la consistència que aporten les classes. Si es guardessin en un fitxer de text, s'hauria de crear un intèrpret personalitzat i no hi ha cap forma d'assegurar a priori que el contingut del text coincideix amb el contingut de la classe. Com podem veure a la Figura 86, existeix un arbre de classes que determina el tipus de cada habilitat. Aquesta com a mínim tindrà un índex, un nom, una descripció, un cost de mana, un temps de refredament, una duració (que serà zero si l'habilitat és instantània) i una icona. Cada tipus concret d'habilitat extendrà aquesta llista d'atributs i mètodes per tal d'implementar cada funcionalitat concreta.

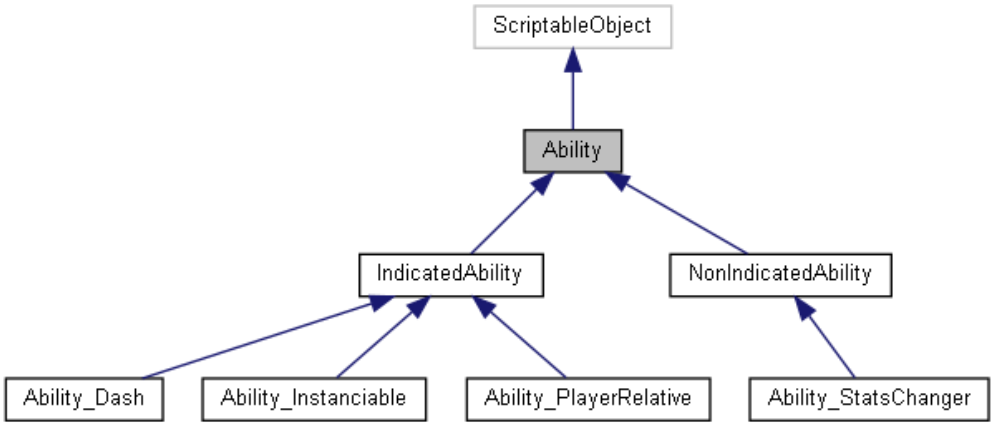

Figura 86: Arbre de classes dels tipus d'habilitats

Com podem veure a les Figures 87 i 88, per tal de poder crear un *asset*[40] del tipus Ability, cal posar l'atribut *CreateAssetMenu*. Aquest crearà una nova opció dins de tots els tipus de recursos que es poden crear amb el nom que li indiquem com a paràmetre.

```
[CreateAssetMenu(menuName = "Ability/Indicated/Dash")]
0 references
public class Ability_Dash : IndicatedAbility
```

Figura 87: Atribut CreateAssetMenu

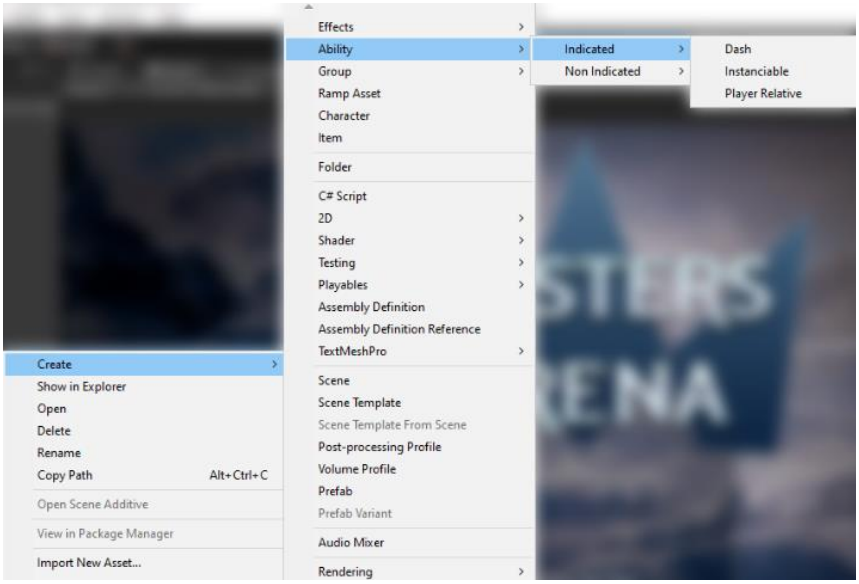

Figura 88: Menú de creació d'assets

6.3.5.2 AbilityBase

Aquest script s'encarrega de mantenir el comportament de cada habilitat instanciable. Cadascun dels *prefab* de cada habilitat tindrà un script diferent per separar el comportament respecte a algun altre. A la Figura 89 podem veure els diferents scripts de les habilitats intanciables.

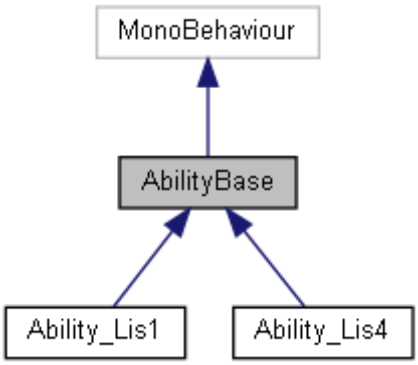

Figura 89: Arbre de classes d'AbilityBase

[40] Un *asset* es qualsevol recurs (o fitxer) que viu a les carpetes del videojoc.

Aquests scripts mantenen el comptador de temps que tarda l'habilitat en fer mal des que apareix per primera vegada, el comptador de quan s'ha de destruir i el què passa quan una entitat entra dins (si se li ha d'aplicar mal i/o un efecte determinat per la configuració de l'habilitat).

6.3.5.3 AbilityCooldown

Aquest script s'encarrega de mantenir cadascuna de les habilitats en termes generals. Com es pot veure a la Figura 90, s'encarrega de mostrar i ocultar l'indicador, cancel·lar la habilitat, manté el comptador del refredament de la habilitat, i actualitza la UI respecte a l'estat actual d'aquesta. Hi ha un component d'aquest script per cada habilitat del personatge.

void	**ShowIndicator** () Sets the current indicator to this ability. **Més...**
void	**HideIndicator** () Hides the current indicator and triggers the current ability. **Més...**
void	**CancelIndicator** () Cancels the current indicator and DOESN'T trigger the current ability. **Més...**
void	**ShowAbilityBlocked** () This function is called whenever this ability is blocked. **Més...**
void	**AbilityReady** () This function is called whenever this ability is ready and you have enough mana. **Més...**
void	**CoolDown** () This function is called whenever this ability is in cooldown. **Més...**
void	**NoMana** () This function is called whenever you have no mana for this ability. **Més...**
bool	**HaveEnoughMana** () Returns true if you have enough mana to cast this ability **Més...**

Figura 90: Llista de mètodes privats de l'script AbilityCooldown

6.3.6 Sistema d'efectes

Els efectes seran tots els canvis d'estadístiques de qualsevol tipus que se li apliquin a les entitats.

6.3.6.1 Effect

Aquests canvis han de ser iguals per tots els jugadors de la partida, per tant s'ha optat per crear un *ScriptableObject* que els mantingui. Aquest objecte ens aporta els beneficis comentats a l'Apartat 6.3.5.1, però en aquest cas per als efectes. Com es pot veure a la Figura 91, hi ha diferents tipus d'efectes.

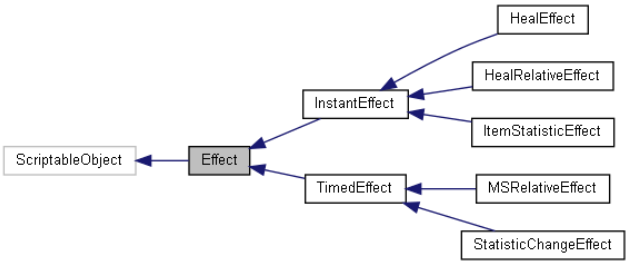

Figura 91: Arbre de classes d'Effect

Principalment podem distingir entre dos tipus d'efectes:

- **Efectes instantanis**: Aquests efectes s'executen un cop i no fan res més.
- **Efectes en el temps**: Aquests efectes s'executen i en un temps determinat es reverteixen.

Per a crear nous efectes és molt fàcil ampliar sobre el sistema creat. Només cal crear una nova classe que hereti d'algun d'aquests dos tipus i implementar la funcionalitat desitjada.

6.3.6.2 EffectManager

Aquest script és l'encarregat de canviar les estadístiques a l'entitat. Hi haurà un component d'aquest tipus en totes les entitats que puguin ser atacades per les habilitats (entitats mòbils: personatges i *minions*). En el cas que la habilitat sigui en el temps, aquest script mantindrà la corutina que espera aquest temps abans de revertir el canvi d'estadístiques, com es pot veure a la Figura 92.

```
1 reference
public override IEnumerator GetCorrutine(MovableStats stats, EffectManager manager) => ApplyTimedEffect(stats, manager);

1 reference
private IEnumerator ApplyTimedEffect(MovableStats stats, EffectManager manager) {
    if (usesDecimal) {
        stats.ApplyStatistic(eStatistic, eAmountFloat);
        yield return new WaitForSeconds(eDuration);
        stats.RevertStatistic(eStatistic, eAmountFloat);
    } else {
        stats.ApplyStatistic(eStatistic, eAmountInt);
        yield return new WaitForSeconds(eDuration);
        stats.RevertStatistic(eStatistic, eAmountInt);
    }
    manager.OnFinishEffect(this);
}
```

```
5 references
public void TriggerEffect(Effect effect)
{
    if (effect is TimedEffect)
    {
        Debug.Log($"Activant l'efecte {(effect as TimedEffect).eName}");
        StartCoroutine(effect.GetCorrutine(stats, this));
        currentEffects.Add(effect as TimedEffect);
        hUDController?.UpdateEffects(currentEffects);
    }
    else
    {
        Debug.Log($"Activant l'efecte {effect.name}");
        effect.Activate(stats);
    }
}
```

Figura 92: Corutina dels efectes en el temps

6.3.6.3 EffectHUDController

Aquest script s'encarrega de mostrar tota la llista d'efectes en el temps, actius actualment en pantalla. Aquest funciona tenint una llista dels efectes actuals i mostrant-los en pantalla. No es pot tenir més de deu efectes al mateix temps, ja que només hi ha deu imatges que s'activen i modifiquen un *sprite* per representar-ho. Com es pot veure a la Figura 93, es comprova el màxim d'efectes i després es mostren en ordre les imatges de cada efecte. Totes les imatges restants que no tenen *sprite*, es desactiven.

```
public void UpdateEffects(List<TimedEffect> effects)
{
    if (effects.Count > images.Length)
    {
        Debug.LogWarning($"No es poden tenir més de {images.Length} efectes alhora!");
        return;
    }

    int i = 0;
    foreach (TimedEffect effect in effects)
    {
        // Show the effect in the image
        images[i].gameObject.SetActive(true);
        images[i].sprite = effect.eSprite;
        tooltips[i].SetTooltip(effect.eName, effect.eDescription, effect.eSprite, $"{effect.eDuration} s");
        i++;
    }

    while (i < images.Length)
    {
        // Disable the image
        images[i].gameObject.SetActive(false);
        i++;
    }
}
```

Figura 93: Funció de mostra dels efectes

6.3.7 Personatges

Cada personatge tindrà unes característiques inicials úniques, a part de les quatre habilitats pròpies.

6.3.7.1 Character

Aquest recurs serà el que guardarà cadascun dels personatges fent ús dels *ScriptableObjects*. A la Figura 94 es pot veure una llista amb els atributs principals d'aquesta classe.

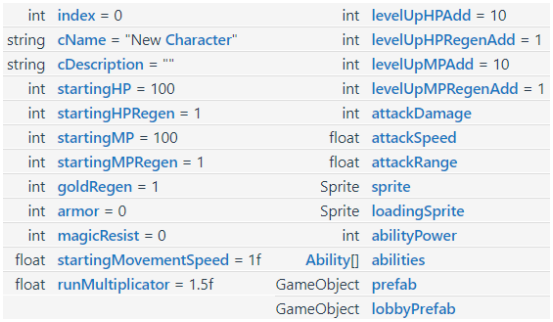

Figura 94: Atributs de Character

6.3.7.2 CharacterGroup

Per tal de passar sempre la mateixa col·lecció de personatges per generar una versió coherent del joc, s'ha creat un *asset* que els agrupa. Així, al només haver-hi una sola instància d'aquest tipus, s'assegura que sempre hi haurà la mateixa informació per a tots els clients i per a totes les pantalles del joc (veure Figura 95).

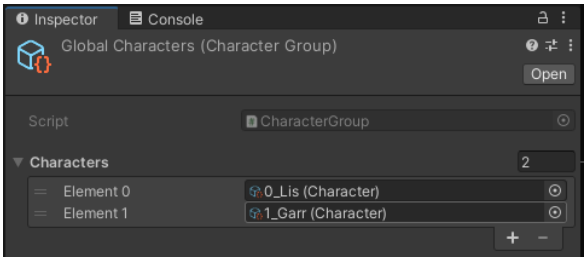

Figura 95: Llista de personatges

Aquesta llista es farà servir a qualsevol part del joc que calgui obtenir informació de qualsevol personatge, com per exemple:

- A la pantalla de càrrega per obtenir la imatge i el nom a partir de l'índex del personatge escollit.
- Al principi de la partida per inicialitzar el personatge amb els valors inicials a partir de l'índex.
- A la lobby per mostrar els models i noms personatges de forma dinàmica quan es clica als botons.

6.3.8 Sistema d'ítems, botiga i inventari

Per tal de poder tenir ítems que es puguin comprar i que apliquin efectes de forma dinàmica, existeixen els següents *scripts*.

6.3.8.1 Item

Aquest script serà un tipus d'*ScriptableObject* que mantindrà tota la informació d'un ítem en concret. A la Figura 96 podem veure l'exemple de l'ítem "*Fire Book*".

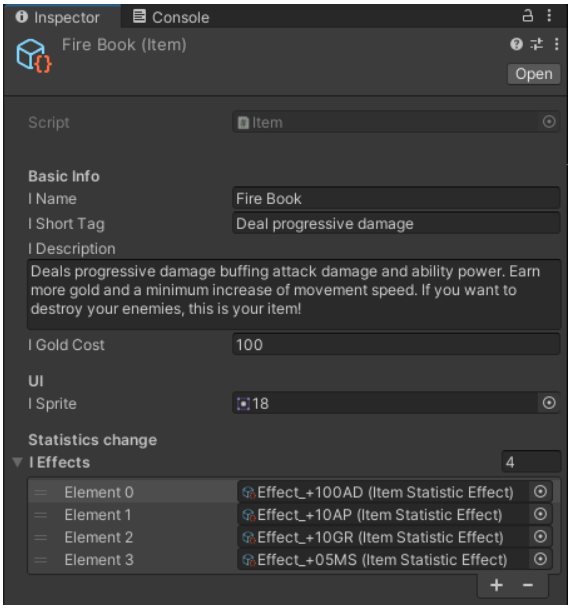

Figura 96: Estadístiques de l'ítem Fire Book

6.3.8.2 ItemGroup

Per mantenir la consistència a la botiga, la qual s'explicarà a l'apartat següent, cal tenir una llista constant que no canviï durant l'execució i que sigui la mateixa per tots els clients del joc. Aquest script permet crear un *asset* que es guarda en disc. Aquest guarda tots els ítems que formaran la llista de la botiga. A la Figura 97 podem veure la instància d'aquest *ScriptableObject* que ens ho permet.

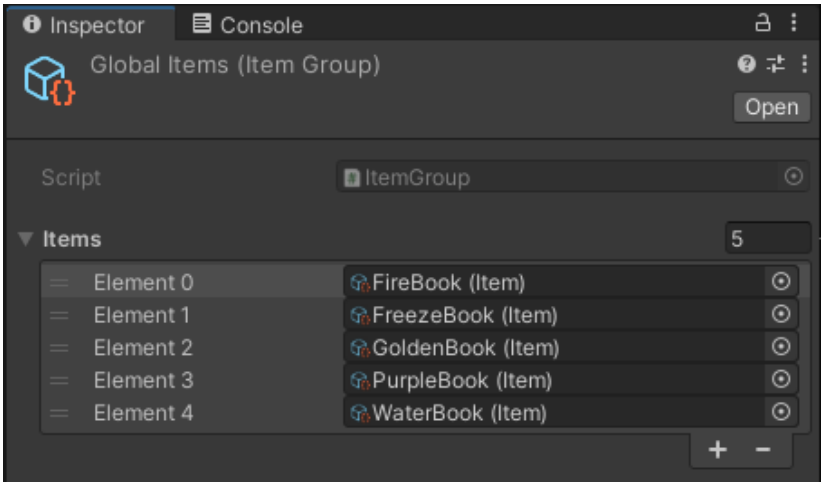

Figura 97: Llista de tots els ítems

6.3.8.3 Shop

La botiga serà la encarregada de mostrar la llista d'ítems de forma dinàmica (com podem veure a la Figura 98) a part d'avisar a l'inventari cada cop que es compri un ítem. La botiga també es pot obrir i tancar, aquesta té una animació cada cop que ho fa. Aquest script serà l'encarregat de moure-la allà on toqui.

```
private void MountItems()
{
    Array.Sort(itemGroup.items, (Item a, Item b) => a.iGoldCost - b.iGoldCost);

    foreach (Item item in itemGroup.items)
    {
        GameObject newItemUI = GameObject.Instantiate(itemUIPrefab);
        ItemUI ui = newItemUI.GetComponent<ItemUI>();
        ui.Initialize(item, currentItem);
        items.Add(ui);
        newItemUI.transform.SetParent(itemList.transform);
        newItemUI.transform.localScale = Vector3.one;
    }
}
```

Figura 98: Funció muntadora dels ítems

6.3.8.4 ItemUI

Cadascun dels ítems a la llista d'ítems de la botiga cal que s'inicialitzi mostrant el nom de l'ítem, i l'*sprite* corresponent, entre altres propietats. A part d'inicialitzar el botó, també s'encarregarà de mostrar si l'ítem és comprable i comprar-lo si es clica amb el botó dret o es fa doble clic, com es mostra a la Figura 99. En el cas de que es faci un sol clic esquerre, es mostrarà més informació en el panell inferior, com s'explica a l'apartat següent.

```
public void OnPointerClick(PointerEventData eventData)
{
    if (eventData.button == PointerEventData.InputButton.Left)
    {
        if (eventData.clickCount == 1)
        {
            currentItem.SetCurrent(this);
            activeImg.enabled = true;
        }
        else if (eventData.clickCount == 2)
        {
            // Right click or double click, buy directly
            BuyItem();
        }
    }
    else if (eventData.button == PointerEventData.InputButton.Right)
    {
        BuyItem();
    }
}
```

Figura 99: Funcionalitat dels clics de l'ítem de la botiga

6.3.8.5 CurrentItemShop

La part inferior de la botiga està controlada per aquest *script*. Com es pot veure a la Figura 100, cada cop que es selecciona un ítem per veure les característiques, aquest ítem canvia els textos i *sprites* per tal de mostrar tota la informació de l'ítem actual.

```
public void SetCurrent(ItemUI itemUI)
{
    if (current != null)
    {
        current.ResetActive();
    }
    else
    {
        localInventory = GameController.Instance.GetLocalPlayer()?.GetComponent<Inventory>();
        group.alpha = 1f;
    }

    current = itemUI;
    Item item = itemUI.GetItem();
    itemImg.sprite = item.iSprite;
    itemNameTxt.text = item.iName;
    itemDescriptionTxt.text = item.iDescription;
    goldValueTxt.text = item.iGoldCost.ToString();

    string effects = "";
    foreach (ItemStatisticEffect effect in item.iEffects)
    {
        effects += effect.ToString() + "\n";
    }
    effectsTxt.text = effects.TrimEnd();
}
```

Figura 100: Mostrar ítem actual a la botiga

6.3.8.6 Inventory

Quan es compra un ítem, aquest *script* el guardarà dins d'una llista. També mantindrà la quantitat d'or que té el jugador. Com es pot veure a la Figura 101, Aquest implementarà tota la funcionalitat de com comprar i vendre ítems. Quan s'intenten comprar, primer es comprova que realment es pugui, seguidament es resta l'or, s'actualitza el text en pantalla, s'afegeix l'ítem a la llista d'ítems comprats, s'apliquen tots els efectes de l'ítem i s'actualitzen les imatges en pantalla per tal de mostrar la nova imatge de l'ítem comprat. En el cas de vendre, l'acció és exactament a la inversa excepte que es retorna un 40% del cost original.

```
2 references
public void BuyItem(Item item)
{
    if (CanBuy(item))
    {
        gold -= item.iGoldCost;
        hUDController.UpdateGoldTxt(gold);
        items.Add(item);
        foreach (ItemStatisticEffect effect in item.iEffects)
        {
            effect.Activate(stats);
        }
        hUDController.UpdateItems(items);
    }
}

1 reference
public void SellItem(Item item)
{
    gold += Mathf.FloorToInt(item.iGoldCost * 0.4f);
    hUDController.UpdateGoldTxt(gold);
    items.Remove(item);
    foreach (ItemStatisticEffect effect in item.iEffects)
    {
        effect.Revert(stats);
    }
}
```

Figura 101: Comprar i vendre ítems

6.3.9 Torretes

Les torretes son un tipus d'entitat que ataca automàticament a tots els enemics dins d'un rang predefinit.

6.3.9.1 TurretStats

Cada torreta tindrà 2000 de vida inicialment. Com es pot veure a la Figura 102, quan aquesta es destrueixi, donarà 1500 d'or i 100 punts d'experiència a l'últim jugador que hagi participat en la destrucció (que hagi aplicat mal). Aquesta entitat funcionarà de la mateixa forma que totes les estructures: els *minions* li podran fer mal amb l'atac bàsic i el personatge podrà atacar-la amb l'atac bàsic però serà invulnerable a qualsevol habilitat.

```
protected override void GiveRewards()
{
    PlayerStats otherStats = GameController.Instance.GetPlayer(lastHitted).GetComponent<PlayerStats>();
    otherStats.inventory.AddGold(goldReward);
    otherStats.AddXP(XpReward);

    GameController.Instance.AddTurret(otherStats.team);
}
```

Figura 102: Recompensa de la torreta

6.3.9.2 TurretShooter

Aquest *script* serà l'encarregat de disparar a totes les entitats enemigues dins d'un rang. Com podem veure a les Figures 103 i 104, la torreta atacarà cada 1 segon i farà 100 de mal. Si l'entitat és un jugador, el mal anirà incrementant de forma lineal per cada cop consecutiu. La torreta anirà atacant en ordre d'entrada a tots els minions i quan no hi hagi minions, atacarà a tots els jugadors dins del rang de la torreta.

```
0 references
private void OnTriggerEnter(Collider other)
{
    if (other.isTrigger) return;

    Stats otherStats = other.GetComponent<Stats>();
    if (!otherStats) return;

    if (otherStats.team ≠ stats.team)
    { // Es enemic
        AddEnemy(otherStats);
    }

    if (indicator.isIdle())
    { // Si no està atacant a ningu,
        Shoot(); // Atacar ara mateix
    }
}

0 references
private void OnTriggerExit(Collider other)
{
    Stats otherStats = other.GetComponent<Stats>();
    if (!otherStats || otherStats.team == stats.team) return;

    RemoveEnemy(otherStats);
    CheckNewState();
}
```

Figura 103: Events d'entrada i sortida d'entitats dins del rang

```
private void AttackWithPriority()
{
    RangedAttackFollow rangedAttack = null;
    if (!isLaser)···

    if (inRangeMinions.Count > 0)
    {
        // Attack the minion
        timesShot = 0;
        if (isLaser)
            inRangeMinions[0].DealDamage(0, 0, damage, null);
        else
            rangedAttack.SetTarget(inRangeMinions[0], 0, 0, damage, null);
        ray.SetTarget(inRangeMinions[0].transform);
        indicator.OnStateChanged(TurretIndicator.TurretState.AttackingOther);
    }
    else if (inRangePlayers.Count > 0)
    {
        // Attack the player
        if (isLaser) inRangePlayers[0].DealDamage(0, 0, damage, null);
        else
        {
            timesShot++;
            rangedAttack.SetTarget(inRangePlayers[0], 0, 0, damage * timesShot, null);
        }
        if (inRangePlayers[0].name == localPlayerName)
            indicator.OnStateChanged(TurretIndicator.TurretState.AttackingLocal);
        else
            indicator.OnStateChanged(TurretIndicator.TurretState.AttackingOther);
        ray.SetTarget(inRangePlayers[0].transform);
    }
}
```

Figura 104: Atac amb prioritat

6.3.9.3 TurretIndicator

Aquest *script* acompanya al TurretShooter mostrant l'indicador respecte a l'estat actual d'aquest. En termes generals, aquest sistema funciona com una màquina d'estats. Com es pot veure a la Figura 105, l'indicador anirà canviant entre tres colors depenent de l'estat actual. Aquest *script* també mantindrà l'opacitat de l'indicador respecte a la distància amb el jugador local. Per calcular la distància de forma més eficient es fa servir la distància al quadrat per no haver de fer una arrel quadrada a cada iteració del joc, ja que aquestes són relativament costoses.

```
0 references
void Update()
{
    if (isReady && playerTransform)
    {
        float sqrDist = (transform.position - playerTransform.position).sqrMagnitude;
        actualColor.a = Mathf.Clamp(Utils.MapRange(sqrDist, maxDistance, minDistance, 0f, 0.2f), 0f, 0.2f);
        lerpColor = Color.Lerp(lerpColor, actualColor, Time.deltaTime * lerpColorVel);
        _renderer.material.color = lerpColor;
    }
}

/// <summary>
/// This function gets calls whenever the Turret changes the intern state.
/// </summary>
/// <param name="newState">The updated TurretState of the turret</param>
8 references
public void OnStateChanged(TurretState newState)
{
    state = newState;
    actualColor = GetColorFromState();
}
```

Figura 105: Funcionament de l'indicador

6.3.9.4 Inhabilitació de torretes

Per tal d'evitar que l'enemic ataqui directament al nexe enemic, s'ha implementat un sistema d'inhabilitació de torretes. Al mapa, per defecte, només una torreta està activa. Al ser destruïda, activarà les següents. Aquesta acció es reproduirà recursivament fins a arribar al nexe. En quant el nexe es destrueixi, la partida acabarà i no s'haurà d'activar cap altra estructura més. Mentre una estructura està inactiva pot atacar però no pot ser atacada. Com es pot veure a la Figura 106, per indicar-ho al jugador, al damunt de cada torreta no hi apareix cap barra de vida.

Figura 106: Estructures inhabilitades

6.3.10 *Minions*

Els *minions* són entitats d'un dels dos equips que van caminant direcció a la base enemiga i que ataquen a qualsevol entitat que es trobi per davant.

6.3.10.1 MinionAI

Aquest *script* s'encarregarà de mantenir la màquina d'estats que controla la intel·ligència artificial del *minion*. Aquesta màquina té dos estats: Caminant i Atacant. Com es pot veure a la Figura 107, en el cas de que el *minion* es trobi a qualsevol entitat enemiga, aquest canviarà d'estat a Atacant. Quan aquest acabi amb totes les entitats dins del rang, o aquestes surtin, el *minion* tornarà a l'estat Caminant en direcció a la base enemiga.

```
0 references
private void OnTriggerEnter(Collider other)
{
    if (other.isTrigger) return;

    Stats _otherStats = other.GetComponent<Stats>();
    if (_otherStats && _otherStats.team != stats.team)
    {
        inRangeEnemies.Add(_otherStats);
        if (state == AIState.WALKING) // Si el minion està caminant, atacar
        {
            targetStats = _otherStats;
            Attack();
        }
    }
}

0 references
private void OnTriggerExit(Collider other)
{
    Stats _otherStats = other.GetComponent<Stats>();
    if (!_otherStats) return;

    inRangeEnemies.Remove(_otherStats);
    if (_otherStats == targetStats)
    { // Si l'enemic actual se'n va de rang, deixar d'atacar-lo
        if (inRangeEnemies.Count == 0)
        { // Si no tenim cap altra entitat en rang, simplement caminar
            Walk();
        }
        else
        { // Si no, atacar a la primera que ens hem trobat
            targetStats = inRangeEnemies[0];
            Attack();
        }
    }
}
```

Figura 107: Events d'entrada i sortida d'entitats dins del rang del minion

6.3.10.2 MinionStats

Aquest *script* manté les propietats de l'entitat del *minion*. Aquest tindrà inicialment 100 de vida i 50 d'atac físic. Com es pot observar a la Figura 108, aquest recompensarà a l'últim jugador que hi hagi interaccionat (que li hagi aplicat qualsevol tipus de mal) amb 30 d'or i (100xp + 1xp per cada segon de partida al moment de la eliminació) al eliminar-lo.

```
protected override void GiveRewards()
{
    GameController gc = GameController.Instance;
    // Every minion will give a baseXP + 1xp every second
    int xpAmount = baseXpAmount + Mathf.FloorToInt(gc.GetTime());
    PlayerStats otherStats = gc.GetPlayer(lastHitted).GetComponent<PlayerStats>();

    otherStats.AddXP(xpAmount);
    otherStats.inventory.AddGold(goldReward);

    gc.AddMinion(otherStats.team);
}
```

Figura 108: Recompenses del minion

6.3.10.3 MinionSpawner

Cada base tindrà un generador de *minions* que n'anirà generant onades. Com es pot veure a la Figura 109, Aquest està configurat amb cinc paràmetres principals: el temps d'espera inicial, el temps d'espera entre cada onada, el temps d'espera entre cadascun dels *minions* dins d'una onada, la quantitat de *minions* que s'ha de generar en cadascuna de les onades i l'equip del qual han de ser.

```
public float timeBetweenGroupSpawns, timeBetweenSingleSpawn;
1 reference | 1 reference
public int minionsSpawnCount, startTime = 5;
2 references
public Team team;

// Start is called before the first frame update
0 references
void Start()
{
    if (PhotonNetwork.LocalPlayer.IsMasterClient)
    {
        InvokeRepeating("SpawnMinions", startTime, timeBetweenGroupSpawns);
    }
    else
    {
        enabled = false;
    }
}

0 references
private void SpawnMinions()
{
    StartCoroutine(SpawnEach());
}

1 reference
IEnumerator SpawnEach()
{
    for (int i = 0; i < minionsSpawnCount; i++)
    {
        Vector3 lookTowards = (team == Team.Blue) ? Vector3.right : Vector3.left;
        GameObject minionGO = PhotonNetwork.Instantiate("Minion/Minion", transform.position, Quaternion.LookRotation(lookTowards));
        minionGO.GetComponent<MinionStats>().Initialize(team);
        yield return new WaitForSeconds(timeBetweenSingleSpawn);
    }
}
```

Figura 109: Instanciador de minions

6.3.11 UI

Per tal de mostrar les interfícies de cada entitat al joc cal una jerarquia de classes que ho mantinguin.

6.3.11.1 UIController

La llista de classes que es pot veure a la Figura 110 és l'encarregada de mostrar tota la informació necessària per pantalla per cada un dels tipus d'entitat. Cada cop que hi ha un canvi del qual és necessari actualitzar algun text o imatge en pantalla, cada entitat té un UIController específic per fer-ho.

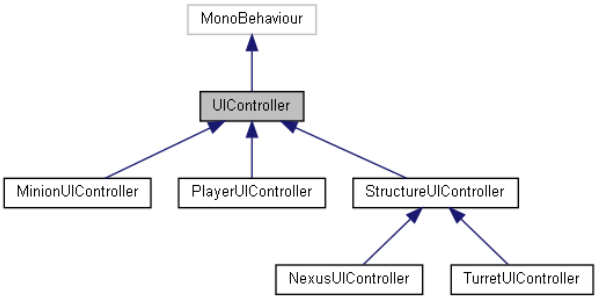

Figura 110: Jerarquia de classes de UIController

6.3.11.2 PlayerHUDController

Aquest script s'encarrega de mostrar tota la informació necessària del propi jugador que no hi cap, o seria molest, a la barra de damunt del cap. Com es pot veure a la Figura 111, aquest controlador manté actualitzats tots els valors de la barra inferior, com per exemple la vida actual, la vida màxima, el manà actual, el manà màxim, el nivell, la experiència, l'or, els ítems comprats o, fins i tot, les habilitats. Aquest també manté l'efecte de la barra secundària que es mostra cada cop que el personatge rep mal, per intensificar el cop.

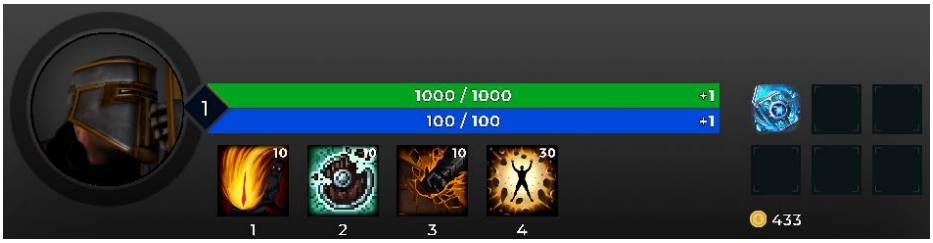

Figura 111: HUD del personatge

6.3.11.3 TooltipManager i TooltipHoverable

Aquests scripts són els encarregats de mostrar i guardar la informació que ha d'aparèixer al *tooltip* cada cop que el ratolí passa per damunt d'un objecte el qual l'ha de mostrar, com es mostra a la Figura 112.

Figura 112: Tooltip de la habilitat Power Up! de Garr

L'*script* TooltipHoverable anirà a qualsevol entitat que es vulgui mostrar el *tooltip* al passar el ratolí per damunt. Aquest haurà de descriure el contingut que hi apareixerà a la etapa d'inicialització. L'*script* TooltipManager rebrà la informació de cada TooltipHoverable i actualitzarà el contingut del *tooltip* al respecte. Aquest li dirà si ha de mostrar-lo perquè ha passat el ratolí per damunt, i quina informació ha de mostrar, o si ha d'ocultar el *tooltip* perquè el ratolí ha sortit de damunt de l'objecte. No es contemplarà el cas de que el ratolí estigui a més d'un lloc alhora, ja no pot passar. Per canviar els colors de cada part del text es fa servir la propietat RichText del TextMeshPro. Com es mostra a la Figura 113, la descripció de cada habilitat té les etiquetes de color corresponents. Totes les variables que viuen entre dobles claus són substituïdes al codi pel valor del respectiu color.

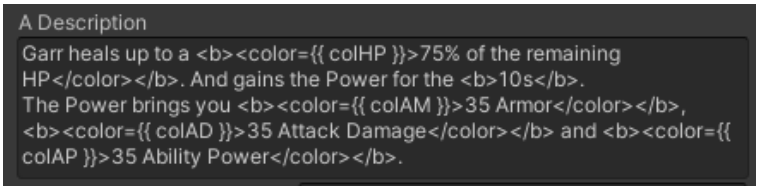

Figura 113: Descripció de l'habilitat

6.3.11.4 Billboard

L'efecte Billboard és un efecte molt utilitzat als videojocs i és molt senzill i útil. Aquest tracta de fer que algun objecte del món sempre miri on està mirant la càmera. Com es pot observar a la Figura 114, Hi ha una subtil diferència entre que "miri cap on està mirant la càmera" i que "miri a la càmera" que es pot observar a la Figura 115.

```
public class Billboard : MonoBehaviour
{

    3 references
    Camera cam;

    0 references
    private void Start()
    {
        cam = Camera.main;
    }

    0 references
    void LateUpdate()
    {
        if (cam) transform.LookAt(transform.position + cam.transform.forward);
    }
}
```

Figura 114: Implementació clàssica del Billboard

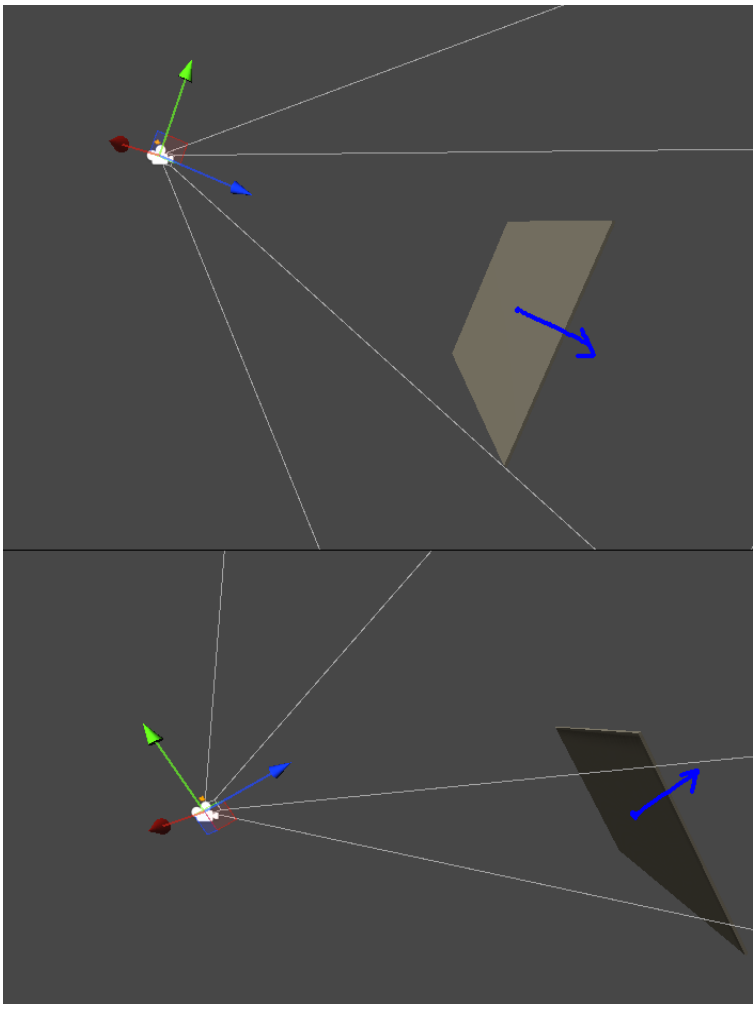

Figura 115: Representació gràfica del Billboard

Aquest script es fa servir a totes les barres de vida de les entitats dins del mapa. Aquestes sempre miren cap on mira la càmera. Aquest *script* s'executa al *LateUpdate*[41] per tal d'haver mogut la càmera i l'entitat sempre abans de rotar-la. Si no s'implementés així, podria portar problemes d'inconsistència entre fotogrames, ja que no s'assegura l'ordre en el qual s'executen els diferents *Updates* dels objectes a l'escena. Inclús es podria optimitzar executant-lo només a l'*Start*, ja que la càmera no rota en tot el joc. S'ha optat aquesta primera opció ja que és la implementació més clàssica i pot portar problemes inesperats si en un futur sí que es vol poder rotar la càmera.

[41] LateUpdate: Event que s'executa a cada iteració del bucle principal del joc, però just després de que tots els events Update s'hagin executat al fotograma actual.

6.3.11.5 Canvi de colors respecte l'equip del jugador local

Com s'ha vist a la Figura 50 i 51, el color de les barres de vida canvien respecte a l'equip del jugador que estigui jugant la partida. A les Figures 116 i 117 es pot veure la implementació concreta d'aquest canvi de color, tant a les barres de vida com als materials de les entitats que ho requereixen.

```
/// <summary>
/// Loads all the necessary colors for all the dynamic UIs in the game
/// </summary>
2 references
private void LoadColors()
{
    myColor = Utils.ColorFromString("#DCE008"); // Yellow // Or green 00A523
    allyColor = Utils.ColorFromString("#11289A"); // Blue
    enemyColor = Utils.ColorFromString("#82030E"); // Red

    smoothAllyColor = Utils.ColorFromString("#82030E"); // Red
    smoothEnemyColor = Utils.ColorFromString("#DCE008"); // Yellow or #C3C026
}

/// <summary>
/// <para>Change the HP color to be green when is ally and red when is enemy.</para>
/// WARNING: The team property NEED TO BE SET BEFORE RUNNING THIS FUNCTION.
/// </summary>
8 references
public virtual void InitializeColors()
{
    if (myColor == Color.clear) LoadColors();
    hpImage.color = (team == PostGameData.myTeam) ? allyColor : enemyColor;
    hpImageSmooth.color = (team == PostGameData.myTeam) ? smoothAllyColor : smoothEnemyColor;
}
```

Figura 116: Implementació del canvi de color a les barres de vida

```
public override void InitializeColors()
{
    base.InitializeColors();
    foreach (MeshRenderer renderer in teamBasedRenderers)
    {
        renderer.material = GetTeamMaterial();
    }
}

1 reference
protected Material GetTeamMaterial() => (team == PostGameData.myTeam) ? blueMat : redMat;
```

Figura 117: Implementació del canvi de material

6.3.12 Regenerador de la base

A dins de la base aliada el personatge regenera vida i mana de forma automàtica. Com es pot veure a la Figura 118, rep una cura de 50 de vida i 50 de mana cada 0.5s. Per mantenir tots els jugadors que hi ha dins de la zona, l'script fa us dels events *OnTriggerEnter* i *OnTriggerExit* mencionats en diversos apartats anteriors.

```
void Update()
{
    // Every 0.5s, regen
    if (Time.time > lastRegenTime + 0.5f)
    {
        HealPlayers();
        lastRegenTime = Time.time;
    }
}

1 reference
private void HealPlayers()
{
    foreach (PlayerStats player in playersInside)
    {
        player.Heal(HPRegen);
        player.RegenMana(MPRegen);
    }
}
```

Figura 118: Implementació del regenerador de base

6.3.13 RespawnCanvas

Cada cop que el jugador mor, pot veure el temps que falta per tornar a reviure amb totes les estadístiques regenerades a la base. Com es pot veure a la Figura 119, per calcular el temps de mort quan el jugador mor, es fa servir la següent formula: $t = 4 + l * 2$, sent t el temps de mort i l el nivell del jugador al moment de morir.

```
/// <summary>
/// Sets the respawn canvas with the proper calculated death time.
/// </summary>
/// <param name="level">My current level that is needed to calculate the respawn time.</param>
/// <param name="stats">The PlayerStats script from the local player who just died.</param>
1 reference
public void Die(int level, PlayerStats stats)
{
    respawnCanvas.gameObject.SetActive(true);
    respawnCanvas.ResetCount(level * 2 + 4, stats);
}
```

Figura 119: Càlcul del temps de mort

Com es pot veure a la Figura 120, en el cas de que no es vulgui esperar es pot comprar la reaparició a canvi d'una quantitat d'or calculada per la següent formula: $g = 500 + t * 9$, sent g la quantitat d'or requerida i t el temps en segons que falta per aparèixer.

```
/// <summary>
/// Spend money to respawn earlier
/// </summary>
0 references
public void BuyRespawn()
{
    stats.inventory.SpendGold(GetRespawnGoldCost());
    stats.Respawn();
    gameObject.SetActive(false);
}

/// <summary>
/// Returns the instant respawn gold cost for a given count.
/// </summary>
3 references
private int GetRespawnGoldCost() => 500 + 9 * (int)count;
```

Figura 120: Funció de comprar reanimació

6.3.14 Estat de la partida

L'*script* GameController és l'encarregat de mantenir un estat consistent en tota la partida. Aquest és un dels pocs *singletons* (veure Apartat 6.4.1) que existeixen al videojoc. Això li permet ser accedit de forma fàcil a tots els altres scripts del joc. Entre moltes altres coses, com es pot veure a la Figura 121, aquest script s'encarrega d'acabar la partida.

```
/// <summary>
/// This is called whenever a nexus gets destroyed.
/// If team is Other, it means surrender.
/// </summary>
2 references
public void FinishGame(Team nexusDestroyedTeam)
{
    Debug.Log("ACABANT LA PARTIDA");
    PostGameData.surrender = (nexusDestroyedTeam == Team.Other);
    PostGameData.looseTeam = nexusDestroyedTeam;
    SceneController.LoadScene(Scenes.POST_GAME);
}
```

Figura 121: Funció d'acabar la partida

6.4 Patrons de disseny

En aquest projecte s'ha fet servir diversos patrons de disseny per tal d'implementar les mecàniques del joc de la manera més escalable i flexible possible.

6.4.1 Singleton

Com s'ha exposat a l'Apartat anterior sobre l'estat de la partida, s'ha fet servir un patró singleton per implementar el GameController. Aquest patró permet que aquesta classe tingui una sola instància i que aquesta sigui accessible de forma global. Aquest patró també s'ha fet servir a dos scripts més: SceneController (manté un estat coherent entre escenes) i TooltipManager (munta i manté l'únic *tooltip* que es pot veure al joc). A les Figures 122 i 123 es mostren la implementació i l'ús en diversos scripts.

```
10 references
private static T instance;
24 references
public static T Instance
{
    get
    {
        if (instance == null)
            instance = FindObjectOfType<T>();
        if (instance == null)
            Debug.LogError($"Singleton<{typeof(T)}> instance has been not found.");
        return instance;
    }
}
```

Figura 122: Implementació del Singleton

Com es pot veure a la Figura 123, a la classe PlayerStats s'implementa tot el comportament que ha de tenir el personatge quan reapareix. Entre altres coses, el personatge cal que vagi cap al punt de reaparició. Aquest punt depèn de l'equip al que pertanyin. Aquesta classe demana el punt de reaparició exacte al GameController que els guarda i els posa a disposició de forma global i pública a tots els scripts del joc.

```
/// <summary>
/// This is triggered when the player respawns. It resets all the CC, heals HP and MP and moves it to the spawn.
/// </summary>
2 references
public void Respawn()
{
    {...
    if (view.IsMine)
    {
        transform.position = GameController.Instance.GetTeamSpawn(team).position;
    }
}
```

Figura 123: Ús del Singleton

6.4.2 Model – Vista – Controlador

A l'hora de mantenir un sistema complex com és el de les entitats, s'ha optat per separar els comportaments de la forma que dicta el patró Model – Vista – Controlador. Aquest patró determina com s'ha de separar el "com es guarden i s'estructuren les dades" de "com es presenten les dades a l'usuari" de "com es comporten i es modifiquen les dades". En aquest cas, pel model es faran servir les classes que hereten d'*ScriptableObject* com Character, Ability o Item. Aquestes s'encarreguen de guardar l'estructura general de la classe o el model de les dades i els valors concrets. Respecte a la vista, totes les classes que hereten de UIController o ItemUI s'encarreguen de presentar la informació de la manera més intuïtiva i interactiva per l'usuari. Finalment el controlador seria tota classe que agafi aquest model i li defineixi un comportament. Al projecte, totes les classes que hereten de Stats o AbilityCooldown són una implementació del comportament de cadascuna de les entitats o habilitats, respectivament.

6.5 Documentació amb Doxygen

Per tal de generar una pàgina on sigui fàcil i intuïtiu visualitzar el codi, s'ha utilitzat la eina *Doxygen*. Amb el *plugin* de *Graphviz* i un full d'estils *css* personalitzat, aquesta eina ens permet generar gràfics interactius dins d'una pàgina generada automàticament a partir del codi font del programa, com el que podem veure a la Figura 124. Aquest fitxer *css* personalitzat permet canviar entre tema clar i tema fosc amb un botó a la barra de la llista de classes.

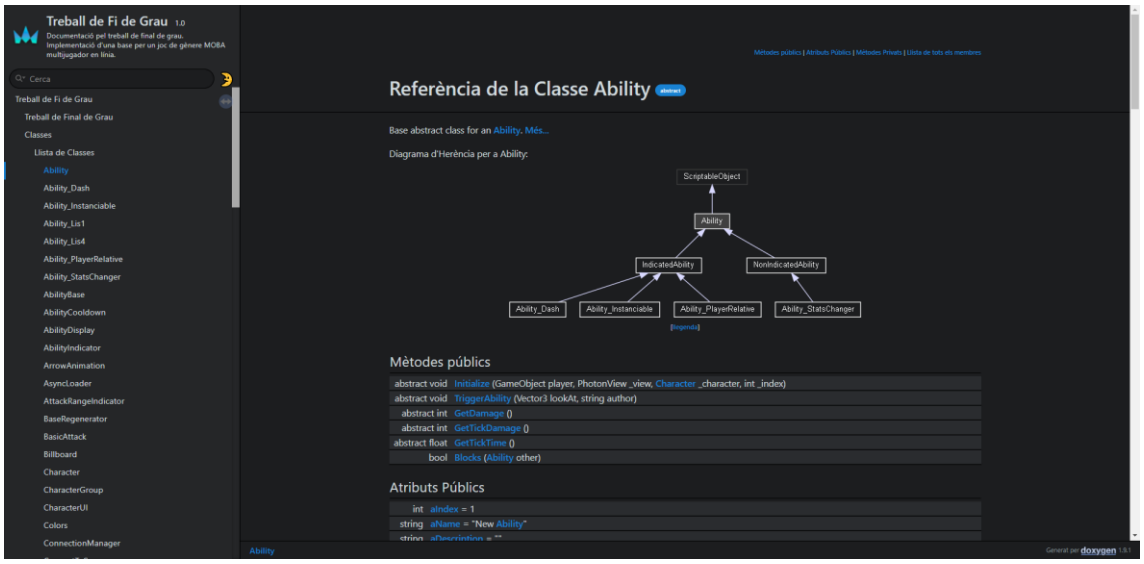

Figura 124: Pàgina de Doxygen de la classe Ability

7 Proves

Per tal de fer les proves de la manera més ràpida possible s'ha fet servir una sèrie d'estratègies que s'explicaran als apartats a continuació.

7.1 Sales d'una sola persona

Al principi del desenvolupament, el joc tardava uns 30 minuts en compilar i generar un executable. Això feia inviable el fet de provar les noves funcionalitats amb dos clients a dins de cada sala. La solució a aquest problema va ser construir la plataforma per tal que acceptés qualsevol nombre de clients dins d'una sala. Amb aquesta idea al cap es va anar construint tot el model del joc mirant que funcionés sempre en els dos casos. Primer es provava que tingués el resultat esperat amb un sol client i després s'exportava el joc i es provava amb dos jugadors. Amb el segon ordinador, aquest temps de compilació s'ha reduït significativament a uns 11 segons, fent possible el fet de treballar directament amb dues persones. Encara així, fins al final del projecte s'ha treballat amb la mateixa estratègia que abans per agilitzar més el procés de desenvolupament.

7.2 Maniquí

El problema amb les sales d'una persona és que no es pot comprovar el què li passaria a un enemic quan aquest és atacat, ja que no hi ha altres enemics a la partida. Per tal de fer possible una interacció entre jugadors inexistents, s'ha creat un maniquí. Com es pot veure a la Figura 125, Aquest funciona de la mateixa exacta manera que un jugador, però amb la variació de que no té connectivitat ni es pot moure.

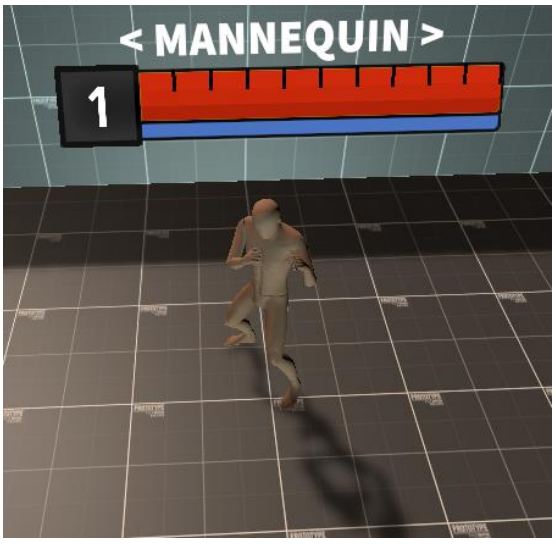

Figura 125: Maniquí a l'escena de proves

Aquest canvi ha facilitat enormement el desenvolupament del joc, ja que s'ha pogut comprovar que les habilitats, els atacs bàsics i totes les mecàniques que afecten al jugador local també afecten a l'enemic.

7.3 Torneig

Per tal de trobar la màxima quantitat d'errors, s'ha organitzat un torneig entre els companys de la classe i amics. Aquests han lluitat entre ells en una competició organitzada per dos grups. Aquests han anat fent partides entre ells en una eliminatòria fins a arribar a la gran final. No hi havia cap tipus de premi pel guanyador, només servia per trobar errors no desitjats que es podrien no haver trobat fins al moment. Al següent Apartat es mostra la llista dels errors més importants trobats pels participants del torneig.

7.3.1 Bugs més importats

En aquesta llista es descriuran els errors corregits més importants que s'han trobat durant totes les partides del torneig.

7.3.1.1 Combinació d'habilitats

Al mostrar l'indicador d'una habilitat i després clicar un altre botó al mateix temps, l'indicador canvia i quan es deixi anar el botó de la primera habilitat s'hauria de llençar. Anteriorment, com es pot veure a la Figura 125 quan la segona habilitat sortia, l'indicador es mostrava de forma errònia quan hauria de desaparèixer. Aquest error s'ha arreglat ignorant tots els intents de mostrar una habilitat mentre hi ha un altre indicador mostrant-se.

Figura 126: Bloqueig de l'habilitat i mostra de l'indicador errònia

7.3.1.2 No es poden connectar dos jugadors a la mateixa partida

Aquest error succeïa al mostrar la pantalla de càrrega i carregar l'escena per tots els personatges a la partida. L'error no permetia jugar a dues persones a la mateixa partida. El problema residia en que el jugador s'instanciava a l'escena persistent en comptes de a l'escena del joc i aquest, a l'intentar fer les crides a *GetComponent* o GameObject.*Find,* no trobava res ja que es trobava a una altra escena. Aquesta escena persistent és la que s'encarrega de mantenir la pantalla de càrrega i mantenir els events de desconnexió de la partida i connexió al servidor entre altres funcionalitats.

8 Resultats

8.1 Assoliment d'objectius

En aquet Apartat s'avaluarà el nivell d'acompliment de cada objectiu proposat a l'Apartat 1.2.

8.1.1 Estudiar el comportament intern d'un joc MOBA

Per tal d'assolir aquest coneixement, s'ha indagat als comportaments de diversos videojocs MOBA com els exposats a l'Apartat 2.4.1. Aquest objectiu es considera assolit ja que el projecte ha acabat tenint el model de joc que s'esperava: un model de joc semblant a jocs del gènere.

8.1.2 Estudiar els sistemes multijugador online

A l'Apartat 6.2 es mostra clarament com es comporta el joc en aquest apartat. A més la plataforma del videojoc requereix molt fortament un sistema multijugador per funcionar correctament. Per aquest motiu aquest objectiu també es considera assolit.

8.1.3 Crear un sistema complex que interconnecti totes les mecàniques

A l'Apartat 6.3 s'expliquen totes les implementacions de les mecàniques que conformen el joc. Totes les interaccions entre jugadors en línia (amb ítems i habilitats), *minions*, torretes i nexe formen part del nucli del qual expandir el sistema del videojoc. Per tot el mencionat anteriorment, aquest objectiu també es considera assolit.

8.2 Resultat final

A continuació es podrà veure una sèrie de captures de pantalla del videojoc per veure com ha resultat el projecte.

Figura 127: Pantalla de càrrega inicial

Figura 128: Pantalla d'inici

Figura 129: Menú principal

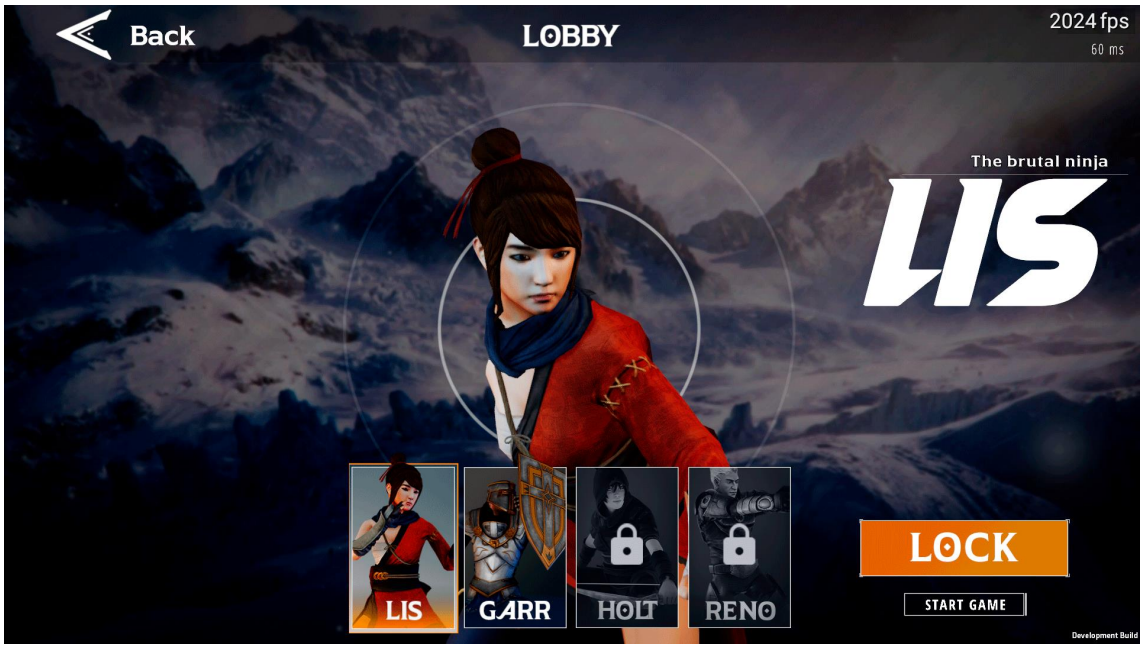

Figura 130: Pantalla de Lobby seleccionant a Lis

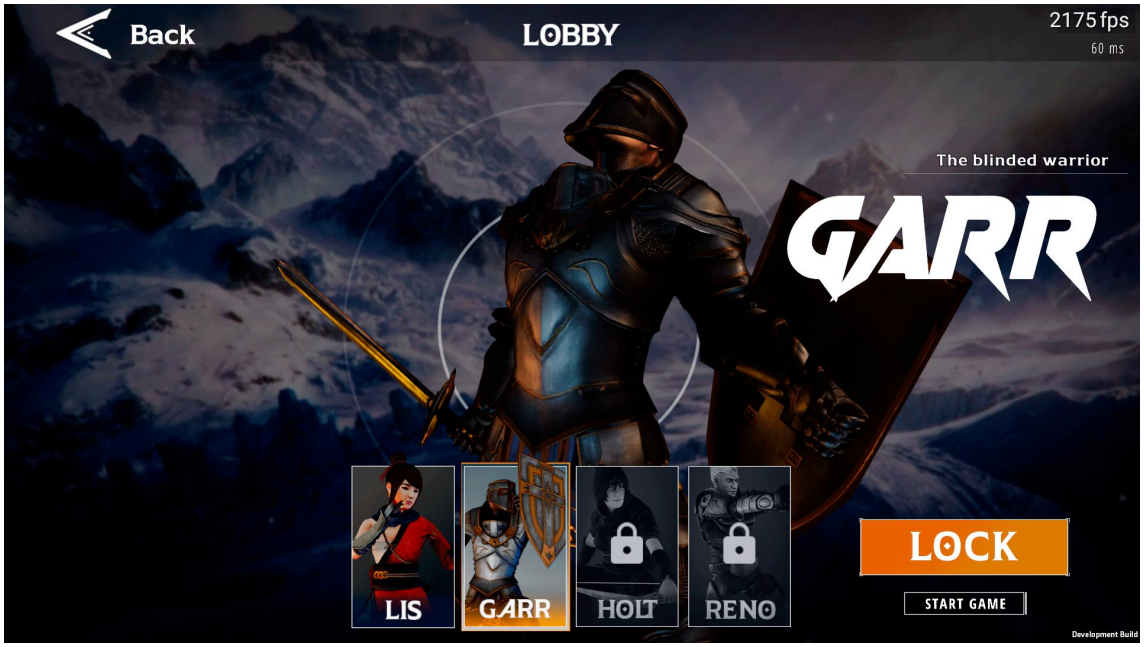

Figura 131: Pantalla de Lobby seleccionant a Garr

Figura 132: Botiga del joc

Figura 133. Garr atacant als minions amb Long Sword

9 Conclusions

En aquest treball s'ha posat en pràctica tot el contingut après durant el Grau de Disseny i Desenvolupament de Videojocs. Encara hi ha molt per aprendre i constantment estan apareixen idees i conceptes revolucionaris que fan que s'hagi de tornar aprendre constantment.

Quan aquest concepte va començar al febrer de l'any 2021, es tenia una vaga idea de la feina requerida i de tota la complexitat de les mecàniques dels jocs en línia. Després de molt d'esforç i feina d'investigació, disseny de mecàniques, art, programació i, sobretot, molta dedicació, s'ha pogut treure endavant aquest gran projecte. Personalment, estic molt orgullós de la feina feta.

Per desenvolupar un projecte d'aquest calibre ha calgut començar el projecte amb antelació i és per això que es va optar per començar un any abans de la entrega. Definitivament, aquesta va ser la millor decisió de tot el desenvolupament.

Durant el llarg del desenvolupament del projecte s'ha anat modificant el pla de disseny inicial per tal de anar ajustant comportaments i millorar mecàniques fent us de la metodologia àgil. Aquesta ha permès deformar el pla inicial a les necessitats del moment de forma fàcil i intuïtiva.

Unity, el motor utilitzat per desenvolupar el projecte, ha estat una de les eines a les que estic més agraït per fer tot això possible. La comunitat, els fòrums i els tutorials d'internet han estat de gran ajuda per tal d'implementar petites mecàniques o errors puntuals que, per compte pròpi serien molt difícils de trobar. Tots els amics que han acompanyat el desenvolupament del projecte han estat un gran suport per a què aquest acabi resultant com ho ha fet. Cada trucada compartint pantalla, cada captura de pantalla preguntant opinió d'un disseny artístic o d'una implementació han portat a aquest projecte a ser com és finalment.

Finalment, cal dir que aquest projecte no ha acabat i estic segur de continuar el desenvolupament d'aquest en un futur, com es pot veure a l'Apartat següent de treball futur. Tots els objectius inicials han sigut assolits, ara cal posar-se'n de nous i continuar endavant.

10 Treball futur

A continuació s'exposarà la llista de propostes i millores potencials per fer la expansió del videojoc un potencial projecte viable en un futur.

10.1 Servidor i base de dades d'usuaris

Ara qualsevol es pot posar el nom que desitgi i fer-se passar per algú altre en qualsevol moment. Amb aquesta millora es posaria un sistema de comptes amb usuari i contrasenya, i cadascú s'hauria de registrar amb un correu electrònic. També s'implementarien les interfícies dins del joc que ho acompanyarien, com es pot veure a les Figures 134 i 135.

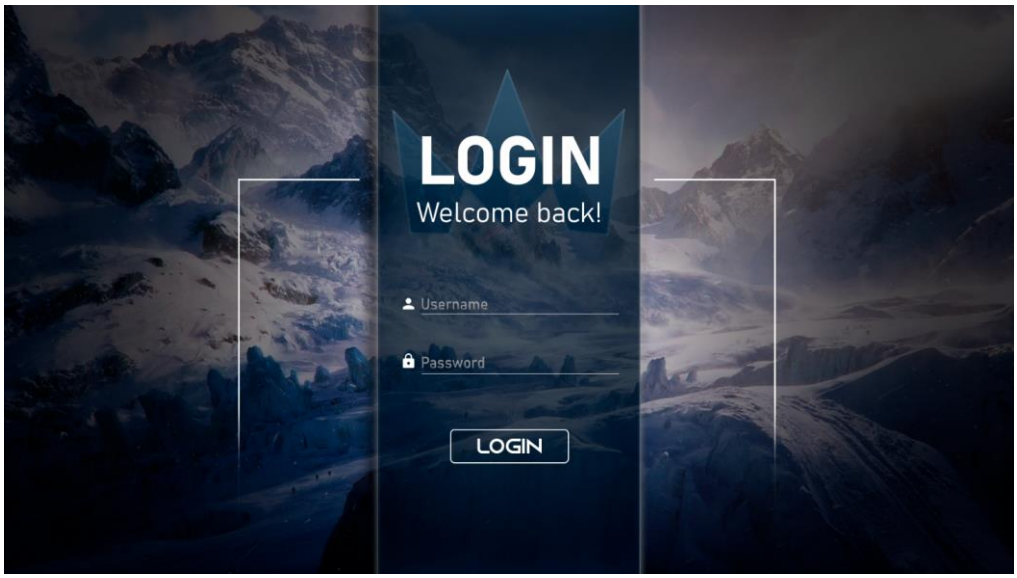

Figura 134: Pantalla de login

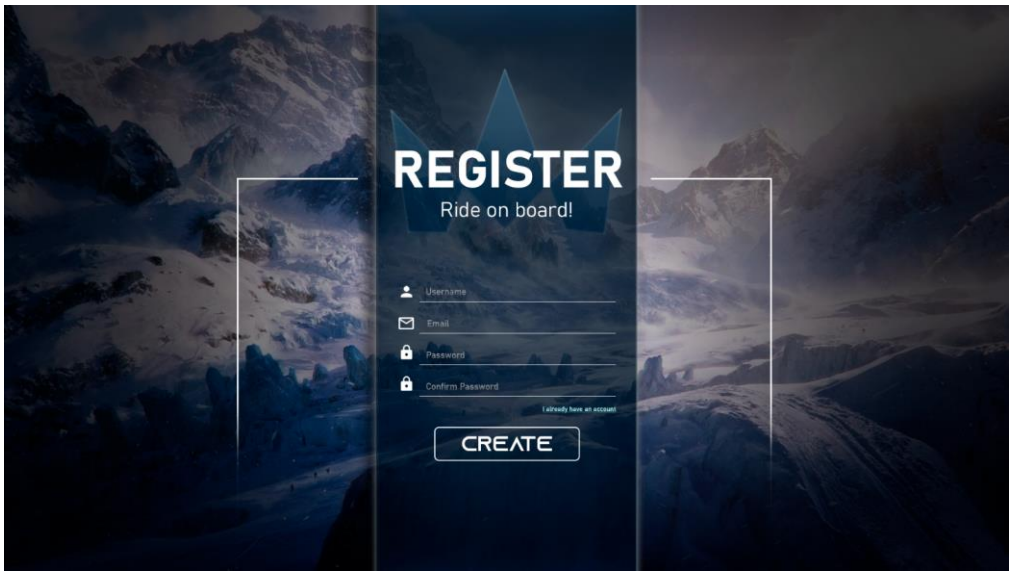

Figura 135: Pantalla de register

De la mateixa manera, aquest sistema introduirà la capacitat de comprar personatges amb una moneda que existirà fora de la partida. Aquesta moneda es podrà aconseguir jugant al joc o comprant-la amb diners. Així mateix, també s'introduirà un sistema de progressió de nivell. A cada nivell el jugador rebrà recompenses addicionals i podrà desbloquejar un personatge nou.
Aquest sistema s'acompanyaria amb una llista negra pels jugadors que fan trampes. També es podria guardar la IP per a que no puguin tornar a jugar amb cap altra compta creada expressament. S'hauria de tenir en compte la llei de protecció de dades i guardar les contrasenyes i targetes de crèdit de forma segura per tal de que ningú pugui accedir a cap informació a la que no hagi de tenir permís.
També s'inclourà una llista d'amics a la qual es pugui veure què estan fent els demés en tot moment. Aquesta llista s'actualitzaria a cada moment que el jugador entrés o sortís d'una partida.

10.2 Implementar un sistema de so

Ara mateix el joc no té cap tipus de so. Això comporta que l'experiència de l'usuari no sigui el màxim d'immersiva com hauria de ser. Caldria implementar un sistema d'àudio en 3d per saber, respecte a la càmera, d'on venen les habilitats. Només s'hauria d'escoltar el què passa dins del rang de visió de la càmera i res més. S'hauria de composar musica pels menús i per dins del joc, a part de generar efectes de so per cada habilitat i atac bàsic de cada personatge. També s'hauria de generar un so de fons per donar cohesió i vida a l'entorn. Aquest so faria que el joc mai quedés en silenci, encara que no estigui passant res en aquell moment.

La UI també hauria de tenir sons així com de clic a cada botó, passar per damunt d'un element interaccionable, comprar i vendre ítems i obrir i tancar la botiga.
Caldria posar sons a cada personatge al morir, al reaparèixer, al pujar de nivell, al destruir una torreta, al guanyar or i al matar un *minion*. Al fer mal, es podria posar un so dinàmic per cada tipus de mal aplicat que sonés més o menys potent respecte a la quantitat de mal. També es podria posar una variant per si el mal és crític.

També es podria contractar actors de doblatge per a que li donin vida a cadascun dels personatges posant-los-hi veu. Podrien tenir frases aleatòries quan porten temps sense dir res i frases contextuals al comprar un ítem, al trobar-se amb un enemic en concret o al ser atacats d'una certa manera.
Cadascun d'aquests sons s'haurien de repartir en els diferents canals que hi haurà al sistema d'àudio i així, poder-se regular des d'un panell d'opcions.

10.3 Sistemes de partícules

Els efectes al pujar de nivell, al comprar un ítem o al tirar una habilitat farien que l'experiència del jugador fos molt més immersiva. S'ha cercat diversos recursos gratuïts a la Asset Store de Unity i s'ha trobat el què es pot veure a les Figures 136, 137, 138 i 139.

Figura 136: Unity Particle Pack

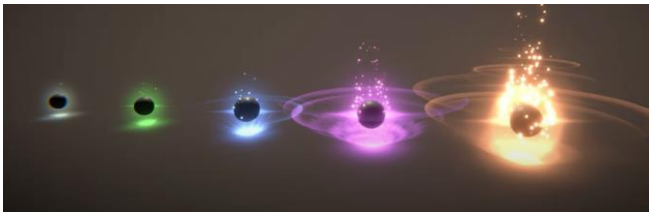

Figura 137: Efectes amb VFX Graph

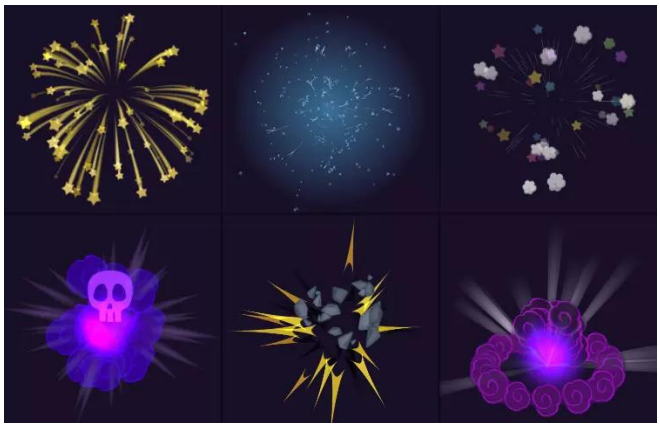

Figura 138: Cartoon FX

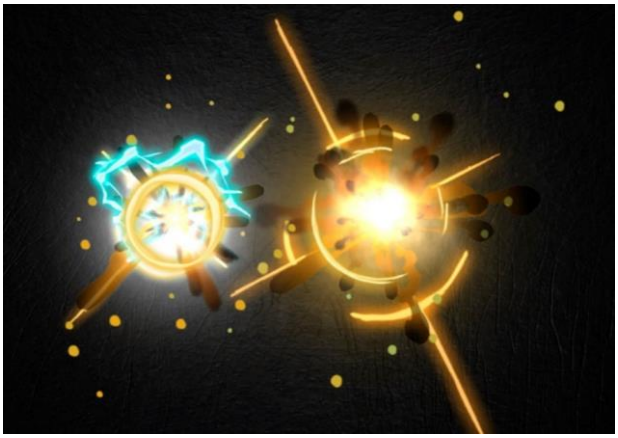

Figura 139: 2D Impact FX

10.4 Moviment a les estructures

Per a què les estructures com el nexe o les torretes no romanguin estàtiques durant la partida, es podria implementar una petita animació aplicant una funció sinusoidal a l'eix Y amb una fase aleatòria, per distingir-se de la resta. Tal i com es veu a la Figura 140, els cristalls de la torreta serien un bon candidat per animar. De la mateixa manera, com es veu a la Figura 141, es podria animar la pedra del nexe i fer servir les cinemàtiques inverses o físiques per animar les cadenes. Quan es tingui la animació d'aquestes estructures, seria tan senzill com implementar-les amb un component *Animator* a dins del joc.

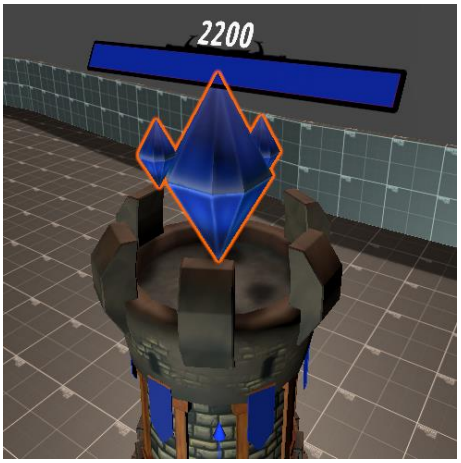

Figura 140: Diamants de les torretes

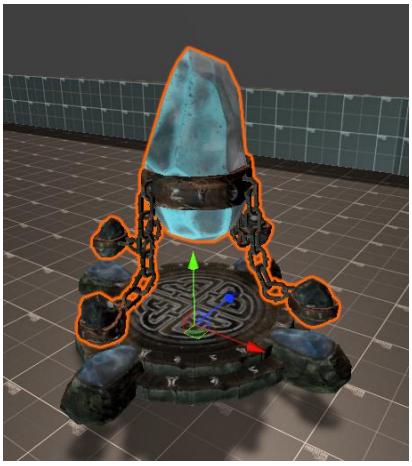

Figura 141: Pedra del nexe

10.5 Canviar el fons del menú principal

Es podria canviar el personatge que surt al menú principal de forma dinàmica per tal de donar vida al joc. Cada cop que surti una nova actualització amb un personatge nou, aquest fons podrà canviar per incitar al jugador a comprar-lo i jugar amb ell.

10.6 Llista de partides i *matchmaking*

Es podria implementar un sistema de cerca de partides automàtica o buscar sala de forma manual, mostrant un llistat com es pot veure a la Figura 142.

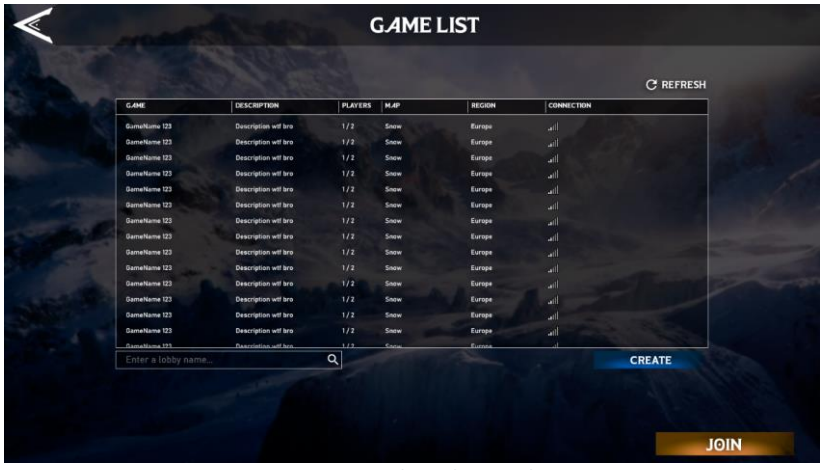

Figura 142: Llista de partides

10.7 Afegir més interfícies meta

Per tal de millorar la sensació del jugador al interaccionar amb el videojoc cal afegir quantes més interfícies de tipus meta (veure Apartat 5.5.2.4), millor. A la següent llista es recullen una sèrie d'efectes que podrien millorar la immersió notablement:

- Un flash de color vermell cada cop que es rep una certa quantitat de mal.
- Un *screen shake* cada cop que es rep mal i que escali de magnitud respecte la quantitat de mal rebut.
- Un número flotant de mal mostrant el mal que s'acaba d'aplicar amb el color del tipus de mal, quan es puja de nivell o es guanya or.
- Càmera suau que acompanya al jugador en ves d'estar emparentat directament.
- Càmera adaptativa que mostra tot el que està davant del jugador en ves d'estar just al mig. Aquesta opció podria ser molt útil per als jugadors ja que podria optimitzar molt el contingut que es mostra en pantalla.

10.8 Col·lecció

La col·lecció serà un espai per veure el què fan les habilitats d'un personatge. El mateix lloc també farà de botiga, com es pot veure a les figures 143, 144 i 145.

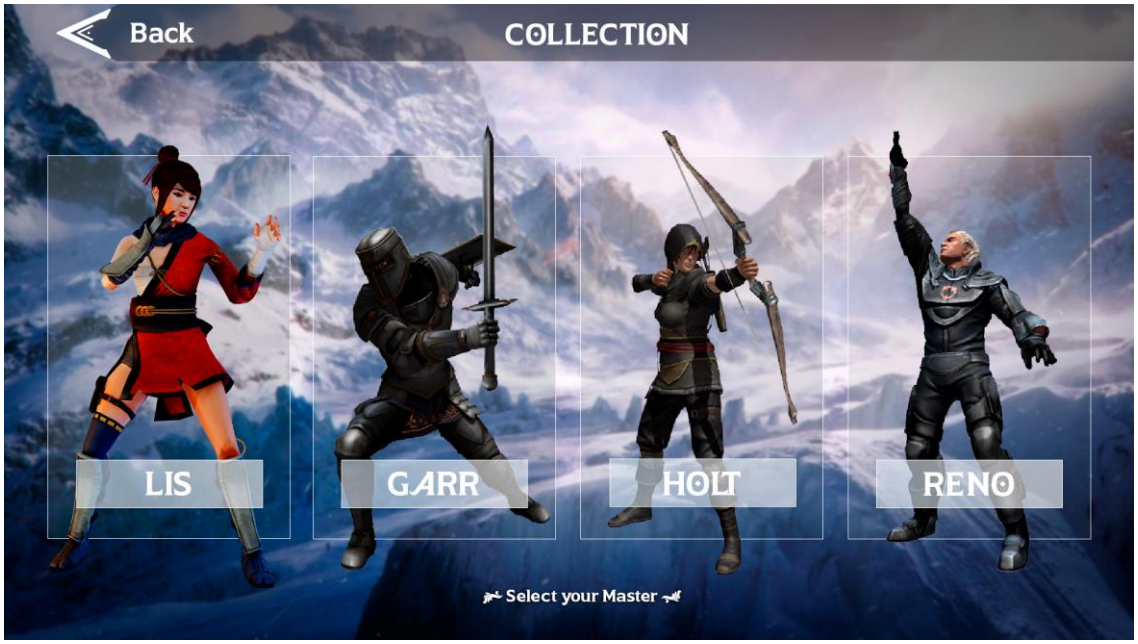

Figura 143: Selector de personatge a la col·lecció

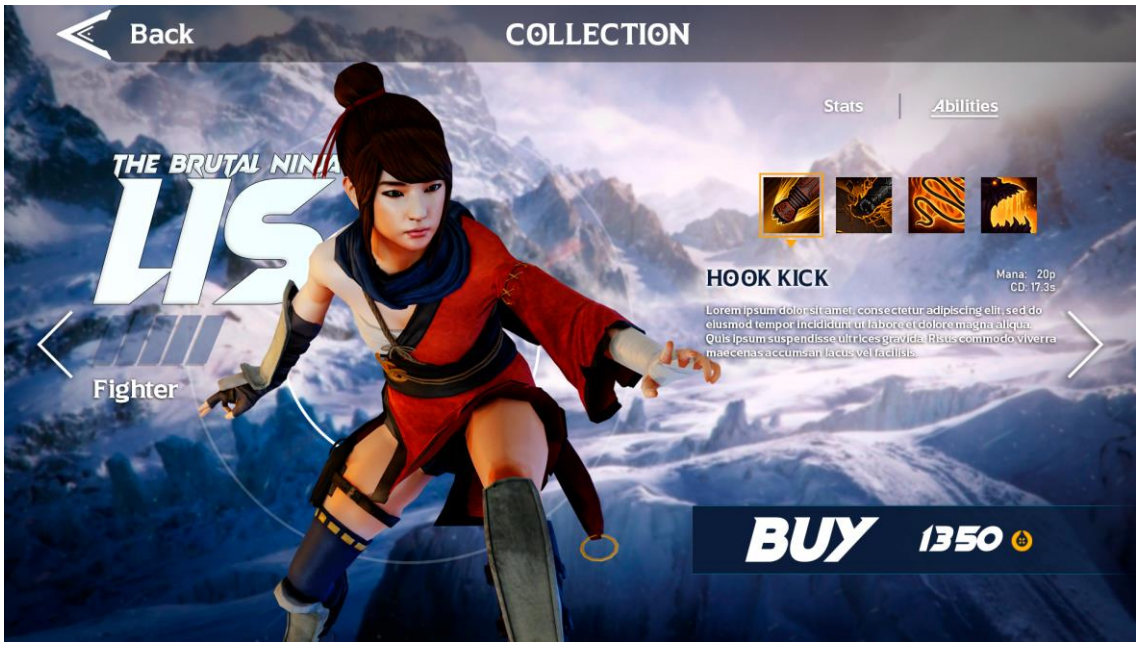

Figura 144: Habilitats del personatge seleccionat

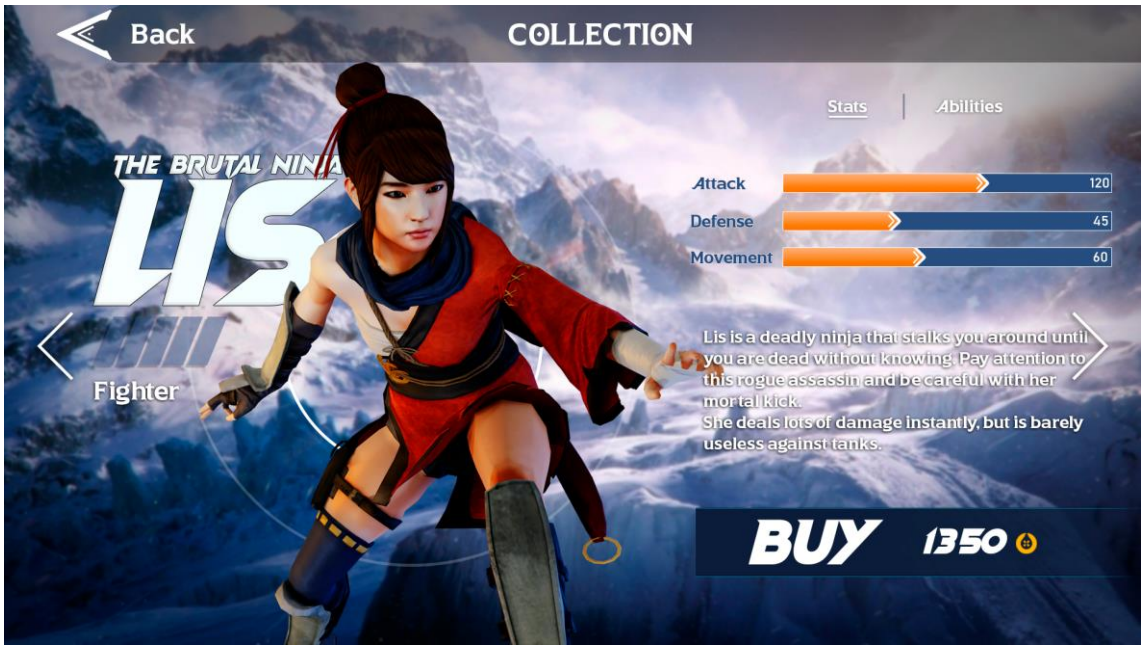

Figura 145: Estadístiques del personatge seleccionat

10.9 Xat de text i veu

Per potenciar la interacció entre els jugadors es posarà un xat de text i de veu per cada sala del joc. Aquest es mostrarà com una capsa de text a la lobby i a dins del joc. Pel xat de veu es podrà mantenir la tecla G i es compartirà el so del micròfon amb els integrants de l'equip.

10.10 Pantalla de configuració de la partida

Es podria afegir una pantalla d'opcions (veure Figura 146) per configurar el joc a les necessitats i gustos del jugador. Aquesta pantalla tindrà diferents opcions com les llistades a continuació:

- Configurar tots els canals d'àudio
- Mostrar o ocultar els FPS i el *ping*
- Canviar la resolució de la pantalla
- Activar o desactivar la pantalla completa
- Canviar la qualitat gràfica del joc
- Activar o desactivar interfícies meta
- Mostrar o ocultar indicadors de les habilitats (llençar directament)
- Mostrar o ocultar els indicadors de les torretes
- Poder canviar els controls de cada tecla
- Afegir controls d'accessibilitat (mode daltònics, augmentar la mida del text, ...)
- Afegir controls per a comandament de consola

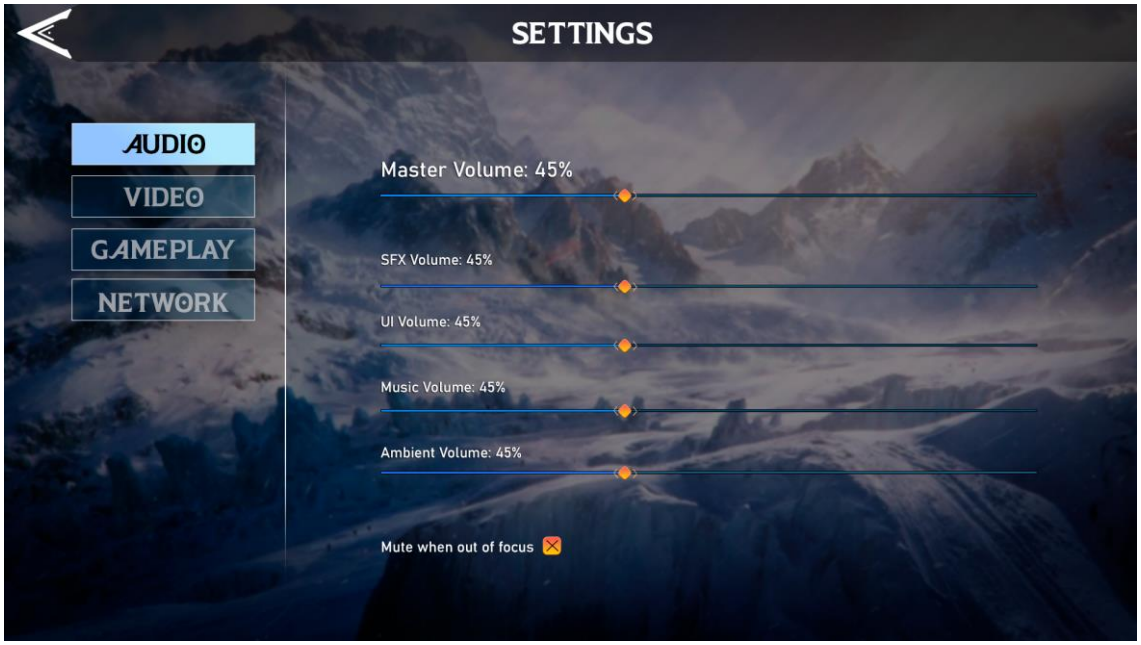

Figura 146: Menu d'opcions

10.11 Tutorial

Es podria afegir un tutorial per mostrar als nous jugadors les mecàniques principals del joc. Es podria accedir a aquest mitjançant el botó que ara està bloquejat al menú principal.

10.12 Afegir més personatges i mecàniques

Per afegir més diversitat al joc es podrien implementar més tipus de personatges i afegir mecàniques al mapa. Això inclouria implementar els dos personatges dissenyats però no implementats: Holt i Reno. També es podrien dissenyar i implementar altres personatges.

10.13 Modes de joc

Com es pot veure a la Figura 147, es podrien crear diversos modes de jocs per atraure més perfils de jugadors, aportant altres punts de vista a la mateixa mecànica de joc.

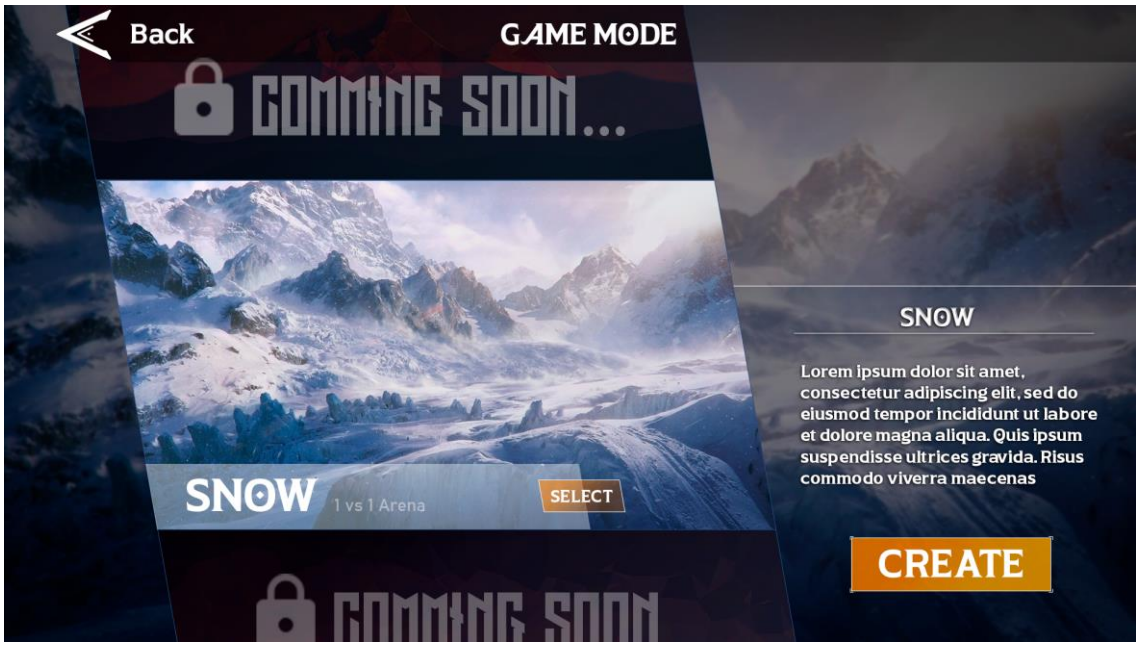

Figura 147: Menú de mode de joc

11 Bibliografia

1. Dimitri van Heesch, Doxygen. https://www.doxygen.nl/

2. Graphviz. https://graphviz.org/

3. Jothepro, Doxygen Awesome CSS. https://github.com/jothepro/doxygen-awesome-css

4. Introduction, Photon Engine Pun 2. https://doc.photonengine.com/zh-cn/pun/current/getting-started/pun-intro

5. Adobe, Mixamo. https://www.mixamo.com/

6. Unity, Unity Particle Pack. https://assetstore.unity.com/packages/essentials/tutorial-projects/unity-particle-pack-127325

7. Jean Moreno, Cartoon FX Free. https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-109565

8. Inguz Media, Free 2D Impact FX. https://assetstore.unity.com/packages/vfx/particles/fire-explosions/free-2d-impact-fx-201222

9. Gabriel Aguiar Prod., Unity VFX Graph - Loot Drop Effect Tutorial. https://www.youtube.com/watch?v=Qig-g9EzND4

10. Blackthornprod, 9 EASY Steps to create a multiplayer game with Unity & Photon - Tutorial. https://youtu.be/93SkbMpWCGo

11. Dota2, Game Heroes. https://www.dota2.com/heroes

12. Phil Nixon, PARAGON - Iconography, UI & HUD - EPIC GAMES. https://www.behance.net/gallery/49956685/PARAGON-Iconography-UI-HUD-EPIC-GAMES

13. Joey Eckert, VALORANT - Agent Select UI Redesign. https://www.behance.net/gallery/96643389/VALORANT-Agent-Select-UI-Redesign

12 Annexos

Com a annex del projecte s'ha adjuntat el projecte de Unity, la documentació generada amb Doxygen i l'executable de la exportació final.

13 Manual d'usuari i d'instal·lació

13.1 Manual d'usuari

Als menús:
Es pot clicar amb el ratolí a qualsevol botó actiu.
A la partida:

- WASD: Control de moviment al pla
- Números del teclat alfanumèric: Llençar habilitats
    - Mantenir per mostrar l'indicador
    - Deixar per llençar
    - Botó dret del ratolí per cancel·lar
- B: Obrir i tancar botiga
- Clic esquerre: Atacar entitats amb l'atac bàsic
- A la botiga:
    - Clic dret del ratolí a un ítem: mostrar la seva informació
    - Clic esquerre o doble clic del ratolí: comprar ítem
- Botó dret a un ítem de l'inventari: Vendre ítem
- Passar el ratolí per damunt de les habilitats, ítems o efectes: Mostra més informació amb un *tooltip*.
- Botó ESC: Menú de sortir de la partida.

13.2 Manual d'instal·lació

Aquest executable només funciona per ordinadors Windows.

Passos per executar el joc a Windows:

- Descarregar el fitxer ZIP.
- Descomprimir el contingut del fitxer ZIP a una carpeta.
- Executar el fitxer "Masters Arena.exe"
- Si apareix un missatge del sistema respecte a que no es coneix l'autor del fitxer cal anar a "Més informació" i després a "Executar".
- Si apareix algun missatge de que el procés necessita permisos per accedir a la xarxa, cal donar-los.
- En aquest punt s'hauria d'estar executant correctament el joc.