TREBALL FINAL DE GRAU

Implementació d'un sistema de narrativa procedimental

Basat en el model de segmentació del usuari en videojocs roguelike

Alvaro Olivet i Gimeno
Grau en Disseny i Producció de Videojocs

CURS 2020-21

TecnoCampus
Escola Superior Politècnica

Centre adscrit a la

upf. Universitat Pompeu Fabra *Barcelona*

Abstract

This investigation has as its main objective, the creation of a project that works thoroughly the procedural narrative, as well as other random and algorithmic elements. Through a small *roguelike* videogame, the aim is to analyze the player based on the different models of user segmentation. With this, we want to foresee the main profile of the player and adapt the development of the game to his style.

Resum

Aquest treball té com a objectiu principal la creació d'un projecte que treballi a fons la narrativa procedimental, alhora que d'altres elements aleatoris i algorítmics. Mitjançant un petit videojoc de tipus *roguelike*, es pretén analitzar el jugador basant-se en diferents models de segmentació de l'usuari. Mitjançant això, es vol preveure el perfil principal del jugador i adaptar el desenvolupament del joc al seu estil.

Resumen

Este trabajo tiene como objetivo principal la creación de un proyecto que trabaje a fondo la narrativa procedural, a la vez que otros elementos aleatorios y algorítmicos. Mediante un pequeño videojuego de tipo *roguelike*, se pretende analizar al jugador basándose en los diferentes modelos de segmentación del usuario. Con esto, se quiere prever el perfil principal del jugador y adaptar el desarrollo del juego a su estilo.

Índex

INTRODUCCIÓ

Els videojocs com a medi d'entreteniment han captivat, avui dia, a molt més usuaris dels que en un principi la indústria hagués imaginat. El nivell de demanda ha proporcionat la creació de molts estudis que tracten d'enganxar als usuaris amb innovacions mai vistes, tant tècniques com artístiques. Aquest marc ha permès que molts videojocs perfeccionin idees que, a principis de segle, ningú hagués imaginat que arribarien tan lluny; el concepte de món obert, sistemes de decisions o ètics, i en el que es centra aquest treball: els sistemes procedimentals.

Quan parlem de procedimental, ens referim a la generació automàtica o algorítmica de quelcom. En els videojocs, aquest sistema pot aplicar-se en múltiples àmbits, tals com l'espai on es desenvolupa el joc, a la fauna i flora o personatges d'aquest o, en el cas del projecte, a la narrativa.

En aquest treball es vol implementar un petit videojoc (mitjançant el motor Unity) on la narrativa procedimental sigui el seu aspecte clau, i per poder fer-ho, s'agafarà el gènere de videojocs *Roguelike*. Mitjançant les característiques que defineixen aquest, s'obtindrà una base que ens permet treballar de manera còmoda la creació de mapes o masmorres procedimentals. D'aquesta manera, tant el món com tot el que existeix en ell, es veurà afectat per la narrativa que anirà desenvolupant-se mentre l'usuari juga.

Per analitzar aquest "usuari", s'utilitzarà els models de segmentació dels jugadors que divideix els jugadors en diferents perfils. D'aquesta manera, es pretén que el joc sigui capaç d'analitzar aquest "perfil" per poder basar el món i els seus reptes en això.

OBJECTIUS

Mitjançant un anàlisis dels diferents models de segmentació de l'usuari i un posterior disseny que segueixi les normés dels videojocs *roguelike*, es plantegen els objectius següents:

- Objectiu principal: desenvolupar un videojoc que implementi un sistema de narrativa procedimental.

Seguidament i amb el fi de detallar més els diferents passos que s'han de seguir, es troben els objectius específics. Es centren en diferents parts del treball i junts, han de validar l'objectiu general. Aquests són els següents:

- Identificar les diferents característiques del gènere *roguelike*.
- Definir els diferents models de segmentació de l' usuari.
- Dissenyar un model de segmentació especial pel projecte.
- Dissenyar un videojoc de tipus *roguelike*.
- Dissenyar l'algoritme d'elecció del perfil del jugador.
- Desenvolupar el videojoc.
- Testejar el videojoc.
- Analitzar el feedback aconseguit.

Els primers dos objectius fan referència al anàlisis de diferents autors i desenvolupadors amb el fi d'obtenir la informació necessària per poder treballar els quatre objectius posteriors. Per últim, es troba la part de testeig que tractarà de validar el videojoc portat a terme.

REFERENTS

Anualment es porta a terme unes conferències organitzades per la GDC (*Game Decelopers Conference*) on diferents acadèmics i desenvolupadors presenten conceptes i avanços en el camp dels videojocs. L'any 2017 es porta a terme una xerrada sobre les millors pràctiques per desenvolupar una narrativa procedimental. Aquesta va estar impartida per *Chris Martens* i *Rogelio E. Cardona Rivera,* dos investigadors de la Universitat estatal de Carolina del Nord. El seu punt de partida es basa en que molta gent vol implementar sistemes de narrativa procedimental sense saber com. A partir d'aquí, plantegen la ponència com una oportunitat per aprendre idees i tècniques que poden servir per a altres projectes.

Defineixen el concepte de narrativa procedimental com una realitat simulada, on les accions improvisades tenen un pes en el món. Afegeixen que aquesta, ha de suposar un balanç entre l'espontaneïtat i l'estructura (*Chris Martens,* 2017, min.01:20).

Definit el concepte de la xerrada, es plantegen diferents models d' actuació segons l'enfocament. Es presenta els dos següents models:

- Orientació centrada en la simulació
- Orientació centrada en l'argument o història

En el primer, s'expliquen els passos a seguir a l' hora d'idear els diferents elements necessaris per a un videojoc:

1. Dissenyar a fons quines són les característiques dels personatges i el món. Tot component ha de partir d'un estat inicial clar.
2. Marcar les normes que decidiran quines interaccions entre els dos elements son clau a l'hora de variar aquests.
3. Posar-ho en un projecte i comprovar-ho.

Per l'altre orientació, es comença presentant el concepte de la *Narrative agenda.* Aquesta tracta de resumir el camí ideal que el dissenyador construeix. Seguidament entren en joc dos elements clau:

- El component que recull tot allò que fa el jugador.

- El component que produeix canvis en el videojoc.

S'explica que aquests dos elements han de mantenir una conversació continua, ja que un depèn de l'altre. El resultat de la conversació suposaria un canvi real en el videojoc i per tant, el primer pas cap a la narrativa procedimental *(Rogelio E. Cardona Rivera,* 2017, min.15:18).

Tot el que s'explica en la conferència s'ha de tenir en compte en el moment de dissenyar el sistema encarregat de la narrativa procedimental. D'aquesta manera, es proposa un camí més organitzat a seguir quan es realitzi la seva creació.

L'altre referent que s'ha tingut en compte és el videojoc *The Binding of Issac.* Aquest *indie* desenvolupat per *Edmund McMillen* i *Florian Himsl*, va publicar-se el 28 de setembre del 2011 i, avui en dia, ja consta de 4 expansions. El joc es defineix com shooter de rol amb tocs de *roguelike*. Compta amb un gran nombre d'ítems, enemics, enemics finals, habilitats, etc... que es generen aleatòriament. El mapa, format per masmorres rectangulars, es generat també de manera procedimental. Amb tot això, el creador presenta una experiència de joc sota la frase: "mai jugaràs dos cops a la mateixa partida".

Tant el plantejament del mapa com molts dels elements que formen el joc s'agafen com a punt de partida a l'hora de dissenyar el videojoc. Amb això, es vol facilitar el disseny del treball a la vegada que es té en compte un dels *roguelike* més famosos que s'han creat.

MARC TEÒRIC

Videojocs roguelike

Pel desenvolupament del projecte és essencial conèixer a fons el gènere *roguelike* per aprofundir en les seves característiques i mecàniques. D'aquesta manera, es pretén poder identificar quins són els trets clau que amb els que el treball ha de comptar.

Quan es parla de qualsevol joc *roguelike* s'entén com aquell que directa o indirectament es veu influït en *Rogue*; un videojoc de 1980 desenvolupat per *Michael Toy* i *Glenn Wichman*. En aquest, el jugador ha d'aconseguir baixar fins l'últim nivell de les masmorres per trobar un amulet. Durant el transcurs, es poden trobar diferents armes, armadures o pocions que ajuden a aconseguir més fàcilment l'objectiu (Daniele Cortesi, 2018, pàg. 28).

A partir de *Rogue*, es defineix aquesta classe de videojocs com un subgènere dels *role-playing-games* (*RPG*) que comparteixen certes característiques especials com la mecànica de mort anomenada 'mort permanent' (*perma-death*) i l'existència de processos procedimentals i aleatoris.

Característiques

Per poder desgranar aquesta classe de videojocs, s'ha tingut en compte la *RogueLike Development Conference,* una conferència anual on es presenta i discuteix tot allò que afecta el gènere. La que aquest treball té en compte és la del 2020, anomenada *RogueLike Celebration* per la celebració dels 40 anys del joc inicial. En ella es troba la desenvolupadora *Lisa Brown* que és la principal dissenyadora de *ProbablyMonsters*. Brown (2020) anomena certes característiques principals que defineixen als *roguelike*:

1. **Dificultat alta:** el jugador morirà molts cops fins a aprendre com poder superar els reptes. Aquesta dificultat pot portar a l'usuari a la frustració, que segons l'autora, és també un element a tenir en compte. Per combatre-ho, és important dissenyar pensant en la duració de cada 'run'; tot aquell temps d'ençà que el

jugador ha començat un intent fins que mor. El temps mig de les *runs* hauria de ser proporcional a la corba d'aprenentatge que el jugador experimenta, és a dir, mentre més avanç presenta, més domina el *gameplay* del joc.

2. **Món generats de forma procedimental:** sentiment d'exploració i novetat per al jugador, ja que l'escenari i els seus elements es creen de forma aleatòria.

   Aquesta característica ve lligada amb el que l'autora remarca com un altre tret important dels *roguelike*: les decisions. Quan un jugador ha d'escollir el camí a seguir o quin objecte/ arma/ ítem agafar, es presenta una decisió que pot marcar el progrés del joc. Com a exemple, al pas de cada nivell en el títol *Nuclear Trhone*, el jugador ha de triar una mutació que dóna diferents habilitats. Per altra banda, en *Hades*, a l'aconseguir diferents vendicions dels déus, el jugador ha de decidir un entre totes elles.

Figura 1: Elecció de vendicions en el videojoc Hades.
Font: Videojoc Hades.

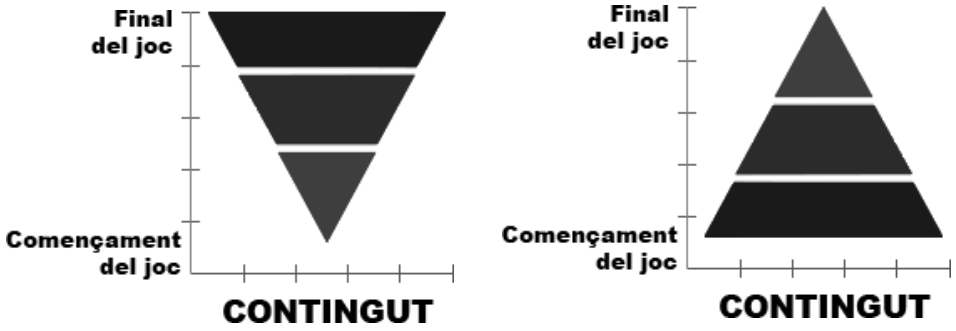
3. **Mecànica 'permadeath'**: si el jugador mor, perd tot el progrés i torna a començar des del principi. També es pot dir que l'usuari només té una oportunitat per aconseguir el seu objectiu.

   Aquesta característica és tractada amb més profunditat en la *Roguelike Celebration* a mans de *Andrew Aversa*, desenvolupador de videojocs com *Tagledeep,* productor, compositor i cofundador de *ISountrack*. *Aversa* (2020) explica com l'existència d'aquesta mecànica pot disminuir el teu públic objectiu

pel descontent de molts jugadors. Arran d'aquest fet, molts títols presenten l'opció de la *permadeath* opcional. Alguns títols que fan això serien *Moonlighter* o *Tagledeep.*

En cas de mantenir la mecànica, és important tenir-la en compte a l'hora de dissenyar el videojoc. Aquesta n'altera el seu transcurs a la vegada que trenca amb la idea tradicional que defineix als videojocs (Aversa, 2020, min.12:17).

Per explicar aquest concepte, *Aversa* (2020) parteix d'un eix vertical que mesura el transcurs del joc i un horitzontal que mostra el nivell de contingut. En la majoria de videojocs obtenim una piràmide invertida. Al principi, s'avança de forma ràpida per reptes senzills que ens porten a una aventura major a la vegada que augmenta el contingut del títol.



Gràfics 1: Relació contingut amb transcurs del títol en videojocs tradicionals i en roguelikes respectivament. Font pròpia.

Quan parlem d'un *roguelike* amb *permadeath* la piràmide es veu normal. Com el jugador morirà repetits cops la part que més veu és el principi del joc i per tant on apareixerà més contingut. Aquest fet ha de ser important a l'hora de dissenyar un videojoc amb aquesta mecànica.

Per últim, *Aversa (2020)* explica que alguns títols tracten de buscar alternatives a aquesta mecànica millorant un sistema de vides que es senti just o implementant un sistemes de guardat com en els jocs tradicionals.

4. **Mecànica 'run-base':** tornant amb Brown (2020), la dissenyadora defineix el concepte *run-base* com el procés on un jugador ha d'intentar quelcom repetits cops fins a aconseguir-ho. Aquest fet, com l'autora remarca en molts moments,

pot portar a la frustració. Saber lidiar amb ella i que l'usuari no caigui presa d'aquesta és responsabilitat del dissenyador.

Una manera de motivar a l'usuari són els *Short Time Horizons*, que consisteix en representar el progrés que s'ha aconseguit de forma visual i clara. Segons *Brown*, no només s'utilitza per encoratjar a l'usuari sinó que també ajuda a fer una idea del recorregut total del joc. Cada títol representa de maneres diferents aquest aspecte; *Slay the Spire* mostra exactament el camí escollit i fins on ha arribat el jugador. Això s'ensenya després de cada repte superat i serveix com eina de progrés i de presa de decisió. Per altra banda, en *Loot Rascals* es mostren les cinc zones que componen el joc i fins quina ha arribat el jugador. És una manera senzilla que permet veure quan proper s'ha quedat del final.

Figura 2: Short Time Horizon del videojoc Loot Rascals. Font pròpia.

Motivacions

Definits els principals trets clau, *Lisa Brown* parla de les diferents motivacions que porta als jugadors a gaudir d'aquest gènere. Es basa en el model de motivacions del jugador proposat per *Quantic Foundry.*

Aquesta és una empresa d'investigació de mercat centrat en les motivacions dels jugadors. Mitjançant tant les ciències socials com l'estadística i el anàlisis, tracten d'impulsar els videojocs amb els seus estudis (Nick Yee & Nicolas Ducheneaut, 2015).

El model que *Quantic Foundry* presenta es divideix en 6 seccions que Brown (2020) adapta les diferents definicions al gènere *roguelike*. S'ha de tenir en compte que la categoria *Social* no es té en consideració, ja que les característiques del gènere solen tenir preferència per un sol jugador.

Action: motivació basada en les accions ràpides amb feedback constant. Predilecció pel combat, sempre que sigui dinàmic i frenètic.

A l'hora de dissenyar pensant en jugadors com aquests s'ha de tenir en compte els *fast failure loops;* temps des que mors fins que tornés a jugar. Aquests han de ser ràpids i no tallar el transcurs del joc.

Les mecàniques agafen protagonisme per aquests usuaris i és important donar un *gameplay* variat. Per això, han d'existir diferents millores o habilitats que permetin donar novetat i no sobrecarregar a l'usuari amb el mateix.

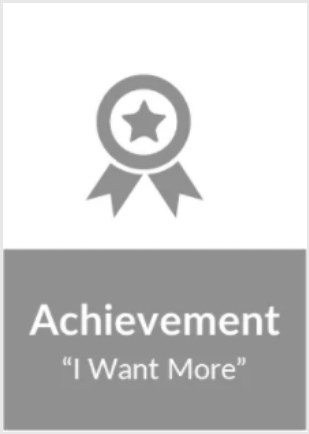
Títol relacionat segons l'autora: *Nuclear Throne.*

Maestry: motivació basada en l'aprenentatge i interiorització del *gamplay* i sistemes del joc per aconseguir superar satisfactòriament tots els reptes.

Jugadors poc espontanis. Es senten atrets per les decisions i premediten aquestes per escollir sempre la millor opció. El combat és només un medi per aconseguir superar un repte i no senten gran afinitat per aquest. El seu objectiu sol ser clar i no defalleixen fins a aconseguir-lo.

Títol relacionat segons l'autora: *Slay the Spire.*

Achievment: motivació basada en la resolució de tasques per aconseguir una recompensa. Aquestes poden ser monedes, punts, nivells de poder, ítems...



Els usuaris no solen ser res altruistes, ja que només es mouen per rebre quelcom a canvi.

El combat, de la mateixa manera que en el perfil *Maestry,* es converteix en una eina per aconseguir recursos.

A l'hora de dissenyar per tals usuaris, és important implementar petits reptes o missions sumades una economia interna. Aquest *loop,* permet una ràpida i fàcil obtenció de les recompenses.

Immersion: motivació basada en sentir-se part del món i història que et presenta el videojoc.

Els jugadors tracten de posar-se en la pell del personatge i actuar com ho faria ell/a. Valoren el *lore* del món, la narrativa que presenta, els personatges i els *NPC's*. Gaudeixen dels diàlegs, ja que profunditzen en el món del joc.

Títol relacionat segons l'autora: *Road Not Taken.*

Creativity: motivació basada en l'aprenentatge i descobriment dels diferents sistemes i mon del joc.

Tenen un alt coneixement del funcionament d'aquest i tracten de crear i provar tot allò que creuen possible. Gaudeixen personalitzant el jugador/a i tot allò que el joc els permeti. Es senten atrets per l'exploració.

Un cop clares les característiques i diferents motivacions que porten als usuaris a jugar a aquest gènere, es important tenir clar quines classes de jugador existeixen, pel que és important estudiar a fons les diferents taxonomies dels usuaris.

Models de segmentació del usuari

Abans de parlar dels models de segmentació de l'usuari, s'ha d'explicar l'origen d'aquests i la seva influència. La seva creació ve lligada amb la gamificació; l'ús de mecàniques de joc en espais o entorns no lúdics. Aquest terme, es comença a tractar al 1973, amb *Charles Coonrad*, considerat el pare de la gamificació. Aquell any, funda la consultora *The Game Of Work* on trasllada les mecàniques de l'esport al camp empresarial. Encara així, no es fins l'any 2002 quan *Nick Pelling*, inventor i programador britànic, crea el terme Gamificació, que segueix vigent avui en dia.

Tornant als 70, l'any 1978, *Richard Bartle y Roy Trubshaw* creen MUD-1, el primer joc multijugador virtual. Arran d'aquesta creació, i influenciat per les tendències que *Coonrad* presenta amb *Game Of Work*, *Bartle* pública un article anomenat: *"Hearts, Clubs, Diamonds, Spades: Players who suit MUDs"*. En ell presenta la seva famosa segmentació del jugador, que serviria d'exemple per molts altres autors (Ballester, 2020, pàg.16-18).

Richard Allan Bartle

Richard Allan Bartle és un escriptor, professor i investigador de videojocs d'origen britànic, famós per ser un dels primers autors a tractar en profunditat els models de segmentació dels jugadors en videojocs MUD ("*multi user dungeon*" o domini multiusuari). En aquest estudi Bartle (1990, pàg.6-18) parla sobre les diferents personalitats dels usuaris segons la seva manera d'interactuar amb el videojoc. El primer que fa és crear un eix d'abscisses que divideix en quatre els usuaris. En cada eix, presenta una confrontació o contraposició:

- **Interacció vs. Acció (***Eix X***):** La contraposició es basa en la relació amb els elements del joc:
    - **Acció:** relació directa sobre aquests.
    - **Interacció:** relació mútua o simbiòtica amb aquests.
- **Jugadors vs. Món (***Eix Y***):** La diferencia fonamental és la seva motivació a l'hora de relacionar-se en el joc:
    - **Jugadors:** preferència d'interacció amb altres jugadors.

- **Món:** preferència d'interacció amb el món del joc.

Un cop establerts els eixos, apareixen els quatre perfils d'usuaris:

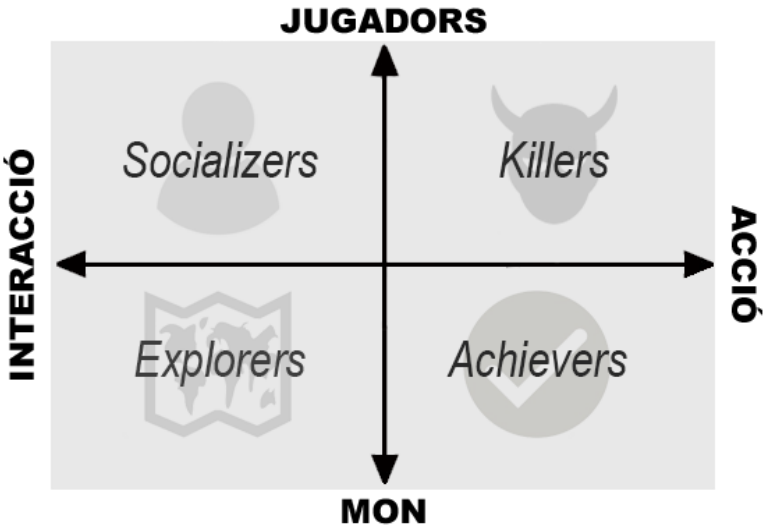


Figura 3: Model de segmentació de l'usuari segons Bartle. Font pròpia.

1. **Achievers (***Acció + Món***):** El seu objectiu és resoldre amb èxit els reptes del joc i ser recompensats per fer-ho, ja sigui amb punts, nivells, rangs en línia, etc... Són competitius.
2. **Explorers (***Interacció + Món***):** Tracten de descobrir i aprendre sobre el món i sistemes del joc. Són curiosos.
3. **Socializers (***Interacció + jugadors***):** Usuaris que volen jugar en comunitat. Tenen afinitat per les relacions socials per damunt de l'objectiu del mateix joc. Són sociables.
4. **Killers (***Accio + jugadors***):** Gaudeixen de competir contra altres usuaris. A diferència dels *Achievers*, la seva competitivitat ve directament relacionada amb el component social, ja que senten afinitat per la derrota del jugador adversari més que del seu propi triomf.

Aquest model, encara ser referent per altres autors, presenta dues mancances que el mateix *Bartle* identifica. Aquestes són:

- L'evolució del jugador a través del temps.

- La possibilitat de fragmentar els perfils en subtipus.

Identificats els problemes, divideix els 4 perfils anteriors basant-se en una nova contraposició; Implícit Vs. Explícit:

- **Implícit:** actua sense pensar.
- **Explícit:** actua amb una planificació o premeditació prèvia.

Aplicant aquest nou model de fragmentació de l'usuari, el model queda de la següent manera:

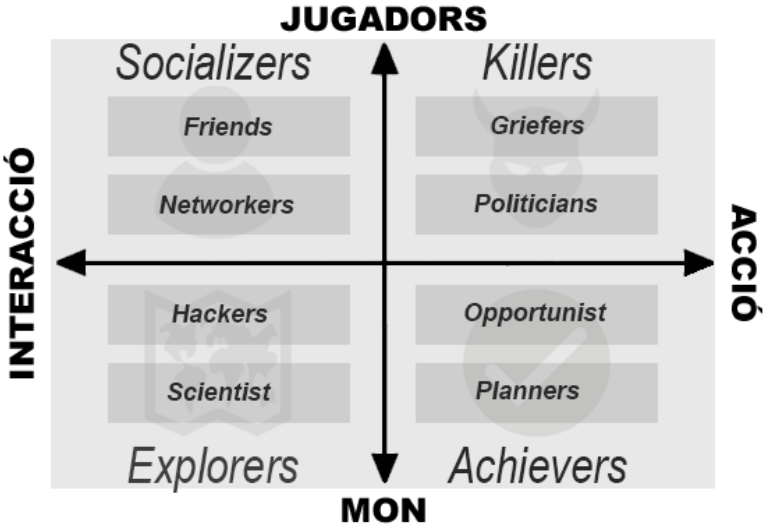


Figura 4: Model de segmentació 2 de l'usuari segons Bartle. Font pròpia.

1.1. Opportunist (Achiever implícit**):** Aprofiten les oportunitats de les quals disposen per resoldre els diferents reptes del joc i poder ser recompensats en conseqüència. No els agraden els obstacles.

1.2. Planners (Achiever explícit**):** Mediten i calculen els seus moviments per superar els reptes de manera satisfactòria. A diferència dels *Opportinists* els agraden el reptes, ja que els força a pensar com superar-los.

2.1. Hackers (Explorer implícit**):** volen descobrir i investigar els límits del joc, gràcies a un coneixement intuïtiu del joc i dels seus elements.

2.2. Scientist (Explorer explícit**):** volen descobrir i investigar els límits el joc de la mateixa manera que els *Hackers*, però gràcies a un coneixement del joc que adquireixen metòdicament amb proves i teories.

3.1. Friends (Socialaizer implícit**):** Busquen relacionar-se amb gent que coneixen bé i ja entenen en profunditat.

3.2. Networkers (Socializer explícit**):** A diferència dels *Friends*, busquen relacionar-se amb qualsevol amb qui pugui interactuar i aprendre d'ell a la vegada que juga.

4.1. Griefers (Killer implícit**):** Tracten de molestar i enfrontar-se a altres usuaris per aconseguir ser el centre d'atenció.

4.2. Politicians (Killer explícit**):** manipulen premeditadament a altres jugadors per aconseguir una reputació positiva en el joc.

La taxonomia que proposa *Bartle* és una de les més referenciades i utilitzades a l'hora de segmentar els usuaris d'un joc. Ha estat estudiada per molts altres autors, que mitjançant aquesta, proposen la seva visió respecte .

Andrzej Marczewski

Andrzej Marczewski, d'origen britànic, és un dels autors més importants actualment en el camp de la gamificació. Basant-se en el model de *Bartle, Marczewski* (2012) elabora una teoria més complexa basada en la predisposició inicial a jugar dels usuaris. Divideix el jugador en 6 perfils segons una contraposició entre si es guien per motivacions intrínseques o extrínsecs:

- **Motivació intrínseca:** usuaris motivats segons el que *Andrezj* defineix com RAMP, és a dir: Relació, Autonomia, Competència (*Maestry*) i Propòsit. En aquest camp es troben quatre dels sis perfils de l'autor.
- **Motivació extrínseca:** jugadors motivats per aconseguir quelcom que els doni una recompensa. En aquest camp es troben els dos últims perfils. Aquests es divideixen en quatre.

Mitjançant aquesta divisió, apareixen les taxonomies següents:

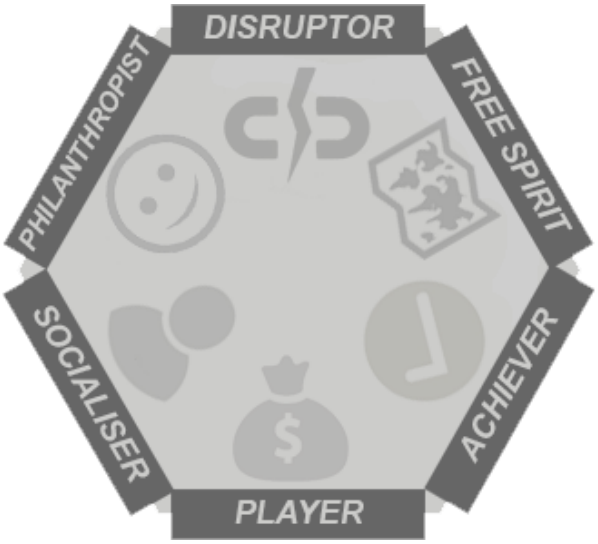


Figura 5: model de segmentació de l'usuari segons Andrzej. Font pròpia.

1. **Socialisers (**motivació intrínseca**):** usuaris motivats per les relacions i l'afinitat amb altres jugadors. De la mateixa manera que el perfil *Socializer* de *Bartle,* valoren més els mecanismes que ofereix el sistema per relacionar-se que l'objectiu o mecàniques del mateix joc.
2. **Free Spirit (**motivació intrínseca**):** jugadors motivats per l'autonomia de crear o explorar. Sense restriccions en l'àmbit del joc.
3. **Achievers (**motivació intrínseca**):** usuaris motivats pels reptes que presenta el joc. Volen dominar les mecàniques per poder arribar a la perfecció com a jugadors.
4. **Phlianthropists (**motivació intrínseca**):** usuaris motivats per un propòsit i significat. Tenen un perfil altruista, ja que tot allò que fan és sense esperar una recompensa a canvi.
5. **Players (**motivació extrínseca**):** jugadors motivats per les recompenses que proporciona el joc. A partir d'aquesta taxonomia, l'autor fa una divisió per especificar més les motivacions:

5.1. **Self-seeker:** aquest grup actua de forma semblant als *Phlianthropists,* però sempre esperant una recompensa a canvi.

5.2. **Consumers:** l'usuari busca ser recompensat de qualsevol manera que el joc presenti. Així i tot, tracten aconseguir-la fent alguna cosa repetitiva i senzilla.

5.3. **Networker:** de la mateixa manera que el perfil de *Bartle,* busquen relacionar-se amb altres jugadors però amb el fi de poder treure'n algun profit.

5.4. **Exploiter:** aquest perfil, de la mateixa manera que els *Free Spirit,* busca explorar els límits del sistema del joc, però motivats per la possibilitat d'una recompensa.

6. **Disruptors (**motivació extrínseca**):** usuaris motivats pel canvi. Busquen interrompre el sistema ja sigui directament o mitjançant altres usuaris. El canvi buscat pot ser negatiu o positiu.

6.1. **Griefer:** gaudeixen afectant negativament a altres jugadors per diversió o descontentament amb les normes del joc.

6.2. **Destroyers:** tracten de trencar directament el sistema. Solen utilitzar *bugs* o *hacks* per aconseguir-ho. Les seves motivacions per fer-ho són les mateixes que els *Griefers*; mostrar el seu desencant pel joc, o per simple diversió.

6.3. **Influencer:** aconsegueixen el canvi en el joc per mig de l'influencia sobre altres jugadors.

6.4. **Improver:** el seu objectiu és canviar el joc per solucionar problemes o millorar-lo.

Chris Bateman

Chris Bateman, dissenyador de videojocs britànic, crea un model de segmentació de l'usuari diferent als ja coneguts amb la finalitat que sigui entès de manera més senzilla.

Figura 6: model de segmentació de l'usuari segons Bateman. Font pròpia.

Per poder fer aquesta divisió, *Bateman (2014)* es basa en milions d'enquestes i recopilacions de dades que compara amb els últims avanços en neurobiologia. A partir d'aquesta comparació neix *BrainHex*: una segmentació en forma de rombe que tracta de simular els diferents estímuls que rep un cervell humà segons el tipus de taxonomia que sigui. Les divisions són les següents:

1. **Seeker:** Tracten de buscar quelcom que puguin despertar el seu interès o imaginació. Són curiosos.
2. **Survivor:** Gaudeixen del sentiment de por. Busquen posar-se en situacions angoixants per superar-les, i així, poder-se sentir segurs altre cop.
3. **Daredevil:** El seu objectiu és buscar l'emoció. Els agrada posar-se en situacions al límit sempre que estiguin controlades i puguin sortir-n'hi victoriosos.
4. **Mastermind:** Jugadors que els agrada l'estratègia, els reptes i resoldre trencaclosques. Sempre actuen amb premeditació.

5. **Conqueror:** Busquen superar enemics d'alta dificultat, ja siguin altres jugadors o enemics del propi joc. Prefereixen batalles renyides per poder aconseguir major satisfacció a l'hora de superar-les.
6. **Socializers:** Tal com expliquen *Bartle* o *Andrezj* , són jugadors que prioritzen les relacions socials a les mateixes mecàniques o objectius del joc. A diferència dels dos autors anteriors, aquests basen el seu comportament en la confiança.
7. **Achiever**: Usuaris que volen aconseguir ítems, diners o punts mitjançant reptes o tasques.

Amy Jo Kim

Amy Jo Kim és una investigadora i dissenyadora d'origen americà, especialitzada en la gamificació i les comunitats online. Escriu l'estudi *Social Engagment Verbs*"on explica el seu model de segmentació de l'usuari, que es basa en el de *Bartle. Amy (2014)* afegeix les motivacions dels jugadors per donar un gir a la divisió prèviament establerta. Considera que els anteriors models no funcionen bé en jocs seriosos o socials. A partir d'aquí organitza les taxonomies en dos eixos amb les mateixes contraposicions que *Bartle:* Jugadors Vs. Món i Interacció Vs. Acció. La diferència principal entre ambdós models és que *Amy* elimina els perfils i els substitueix per verbs que responen als interessos dels jugadors. Aquests es troben en una llista que l'autora

remarca que pot ser ampliable sempre que es trobin altres que concordin. Els verbs principals que conformen la divisió són els següents:

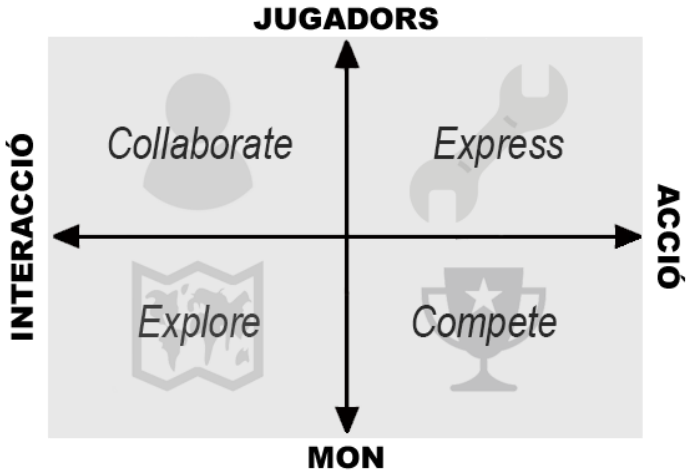


Figura 7: model de segmentació de l'usuari segons Amy Jo Kim. Font pròpia.

1. **Express (**perfil *Killer* de *Bartle***):** Jugadors motivats per les mecàniques que ofereix el joc. Verbs relacionats: escollir, crear, decorar, dissenyar, construir...
2. **Compete (**perfil *Achiever* de *Bartle***):** Tracten d'aprendre i desenvolupar les seves habilitats en el joc per comparar-les amb altres jugadors. Verbs: presumir, guanyar, competir, demostrar...
3. **Explore (**perfil *Explorer* de *Bartle***):** Jugadors idèntics al perfil que *Bartle* proposa. Gaudeixen d'investigar els límits del joc a la vegada que aprenen sobre aquest. Verbs: inspeccionar, veure, qualificar...
4. **Collaborate (**perfil *Socializer* de *Bartle***):** Es basen en les relacions socials a l'hora de complir els seus objectius. Els agrada treballar en equip. Verbs: compartir, ajudar, agrair, donar...

Jon Radoff

Jon Radoff és un emprenedor, investigador i dissenyador de videojocs d'origen americà. El seu estudi està enfocat a les comunitats i medis online. *Radoff* (Ballester, 2020, pàg.28-29), partint del model de *Bartle,* crea una nova taxonomia de l'usuari tenint en compte les motivacions del jugador que analitza l'investigador *Nick Yee*. Aquesta nova divisió és presentada en el llibre *Game On.* Una de les particularitats del model és la seva capacitat d'englobar no només jocs *MMOs*, sinó qualsevol altre gènere.

Per crear la segmentació *Radoff* planteja dos eixos, X i Y, fet que divideix l'usuari en quatre perfils. Cada eix representa el següent:

- **Eix X**: Constitueix el nombre de jugadors que té el videojoc. Cada extrem representa una quantitat de usuaris que va de pocs (part esquerra) a molts (part dreta).
- **Eix Y**: Mostra les diferents motivacions del jugador. En la part superior col·loca les més qualitatives mentre que en la inferior es troben les quantitatives.

Un cop establerts els dos eixos, apareixen els següents perfils:

Figura 8: model de segmentació de l'usuari segons Radoff. Font pròpia.

1. **Immersió (**Pocs jugadors + motivacions qualitatives**):** jugadors que busquen assumir rols, explorar, descobrir el joc i els seus sistemes, desenvolupar-se i aprendre.
2. **Achievement (**Pocs jugadors + motivacions quantitatives**):** usuaris que volen polir les seves habilitats en el videojoc. Gaudeixen del triomf.
3. **Cooperació (**Molts jugadors + motivacions qualitatives**):** jugadors que destaquen pel seu altruisme i predilecció pel treball en equip.
4. **Competició (**Pocs jugadors + motivacions quantitatives**):** usuaris que busquen superar reptes i ser recompensats per fer-ho, ja sigui amb recursos o fama. Els agrada la rivalitat amb altres jugadors fet que els impulsa a superar-se i ser millors.

Bart Stewart

Bart Stewart és un dissenyador i programador de videojocs famós per ser l'autor de *Gamasutra*, un dels blocs sobre gamificació més importants en l'actualitat. Stewart (s. d.) proposa una segmentació de l'usuari que suma la proposada per *Bartle* amb els diferents patrons de personalitat del psicòleg *David Keirsey.* Aquests són proposats per *Keirsey* als 70 en el llibre *Please Understand Me*.

Per construir el model, *Stewart* planteja dos eixos en el que presenta dues contraposicions. Aquestes són:

- **Eix X:** motivacions internes (preferència per allò abstracte) i externes (preferència per quelcom real i concret).
- **Eix Y:** aquest es parteix segons la inclinació de l'usuari cap al Canvi (personalitat lligada a la llibertat o oportunitat) o cap a l'Estructura (personalitat lligada a les regles i l'organització).

Mitjançant aquesta divisió, apareixen els següents perfils:

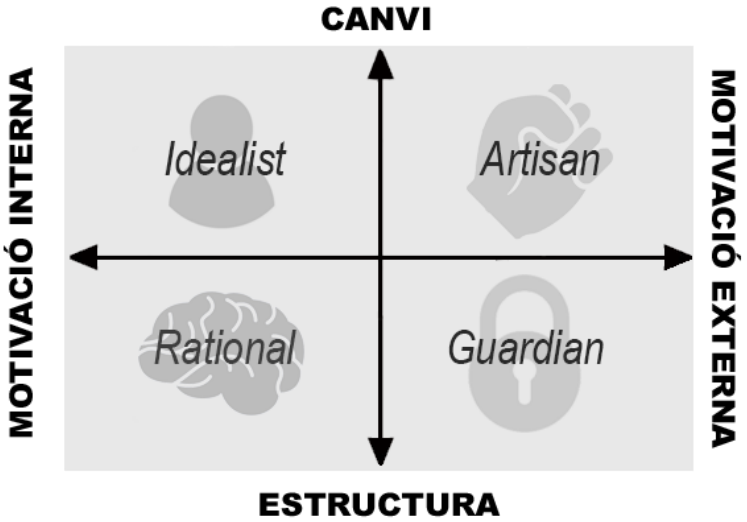


Figura 9: model de segmentació de l'usuari segons Stewart. Font pròpia.

1. **Artisan (**Canvi + motivacions externes**):** Gaudeixen de l'acció i dels estímuls sensorials. Solen ser jugadors realistes, impulsius i altament manipuladors**.**
2. **Guardian (**Estructura + motivacions externes**):** Usuaris que busquen la seguretat orientada a processos ja coneguts o que puguin aprendre. Adoren l'ordre i la logística.
3. **Rational (**Estructura + motivacions internes**):** Jugadors enfocats a l'aprenentatge dels sistemes i mon del joc. Són persones lògiques, estratègiques i innovadores.
4. **Idealist (**Canvi + motivacions internes**):** Usuaris en qui predomina la interacció social amb altres persones a la vegada que busquen la seva pròpia identitat. Són imaginatius i diplomàtics.

Arran de la taxonomia proposada, *Stewart* identifica certs factors que tenen en comú la gran majoria de segmentacions que s'han proposat al llarg del temps. Una de les característiques més comuna és la partició en quatre que molts autors/es solen fer. També troba moltes semblances amb els perfils que *Keirsey* proposa en el seu estudi. Amb tota aquesta reflexió, l'autor desenvolupa la següent taula que tracta de resumir els diferents punts en comú que tenen els diferents perfils dels jugadors:

MOTIVACIÓ	RESOLUCIÓ DE PROBLEMES	OBJECTIU
PODER	RENDIMENT	FER
SEGURETAT	PERSISTENCIA	TENIR
CONEIXEMENT	PERCEPCIÓ	SABER
IDENTITAT	PERSUACIÓ	ARRIBAR A SER

Taula 1: comparativa entre diferents models segons Stewart. Font pròpia.

DISSENY METODOLÒGIC

DISSENY

Disseny de la taxonomia d'usuari per videojocs *roguelike*

Aclarits els diferents models de segmentació de l'usuari és té una idea més clara de les diferents característiques que comparteixen els diferents models presentats. Per altra banda i tenint en compte tota la informació extreta de la *RogueLike Celebration*, es pot crear una taxonomia dels usuaris que juguen al gènere *roguelike.*

Es parteix del model general que proposa *Bart Stewart* on presenta les característiques que tenen en comú les diferents taxonomies. Per altra banda, s'agafen les diferents definicions que dóna *Lisa Brown* sobre les motivacions dels usuaris en videojocs *roguelike.* Si comparem els diferents models s'aconsegueix la següent taula:

MODEL BART STEWART	MOTIVACIONS LISA BROWN
PODER	ACTION
SEGURETAT	ACHIEVEMENT
CONEIXEMENT	MAESTRY
IDENTITAT	CREATIVITY

Taula 2: comparativa entre els models de Stewart i Lisa Brown. Font pròpia.

Com es pot observar en la taula 2, sobre el model de *Quantic Foundry* que utilitza *Brown,* s'elimina l'apartat de *Immersion*. Aquesta decisió ve donada ja que el prototip del projecte és purament acadèmic i no consta de història ni *NPC's*.

Un cop establerts els 4 perfils és important ubicar-los segons una estructura lògica que ens permeti visualitzar i entendre aquests de manera més senzilla. Per fer-ho, es té en compte el model de *Bartle* on es dividia els perfils en quatre gràcies a dos eixos que presentaven una contraposició. En l'eix X es troba un enfrontament entre jugadors que prefereixen interactuar amb el joc contra els que prefereixen interactuar amb altres jugadors. Tal com s'ha explicat abans, el projecte no engloba l'apartat social ni online, per tant, es presenta una nova contraposició entre Món i Gameplay. Aquesta nova motivació respon als jugadors que tenen predilecció pels sistemes contra els que prefereixen el mon que presenta el joc.

Amb això s'obté la següent organització:

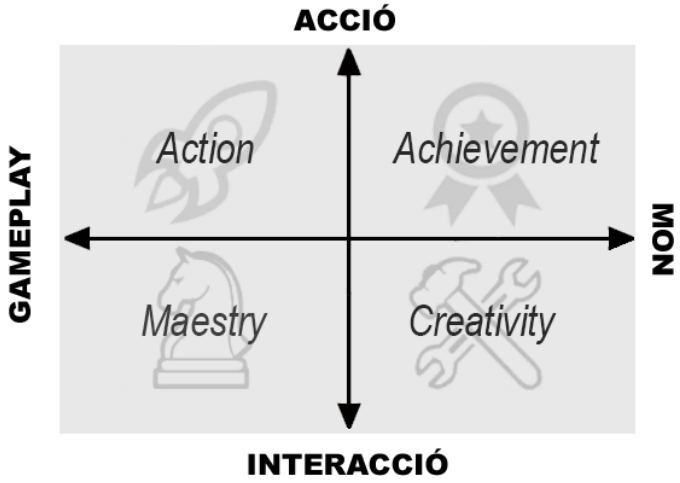


Figura 10: Model de segmentació del usuari per videojocs roguelike. Font pròpia.

Establerts els quatre perfils, és important destacar quines seran les diferents mecàniques i característiques del títol. Amb això, es vol facilitar el disseny dels indicadors que en el moment de la creació del sistema de narrativa procedimental.

El projecte tracta d'un videojoc d'un sol jugador de gènere *roguelike* amb perspectiva *topdown.* El món està creat per sales o masmorres que es col·loquen aleatòriament. L'objectiu del jugador és avançar en aquest i baixar fins a l'últim pis per poder sortir-n'hi. Per fer-ho, haurà de trobar les diferents escales que connecten els pisos mentre combat amb enemics i supera reptes.

Entesa la idea base del joc, s'ha de definir i dissenyar cada element individualment amb el fi que compleixi de forma correcta la seva funció en el sistema.

Jugador principal

L'usuari controla un personatge que capaç de disparar diferents projectils i utilitzar certes habilitats per aconseguir combatre i superar els diferents reptes. Per controlar aquest, es farà servir les següents tecles: W, A, S, D. Amb aquestes, el personatge es pot moure en els dos eixos sense problemes.

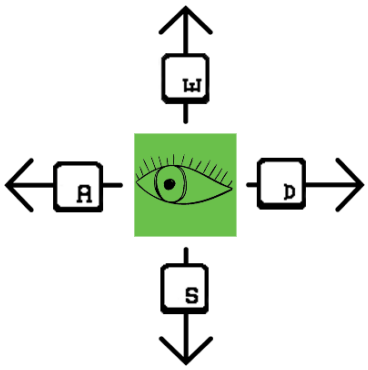


Figura 11: Representació dels controls del jugador. Font pròpia.

Per disparar es fa servir les diferents fletxes del teclat, que permet que es creïn les bales en els dos eixos. Aquests controls són els mateixos que presenta el videojoc *The Binding of Isaac,* ja que donen un control total del jugador que ajuda a l'hora d'enfrontar els diferents combats.

Figura 12: Representació dels controls de dispar. Font pròpia.

També es disposa d'un *parry*. Aquesta habilitat permet rebotar certs projectils dels enemics. Aquesta característica ajuda al jugador a no haver d'esquivar tots els atacs i presentar a la vegada un contraatac. Per activar-lo a s'utilitza el *SPACE* del teclat.

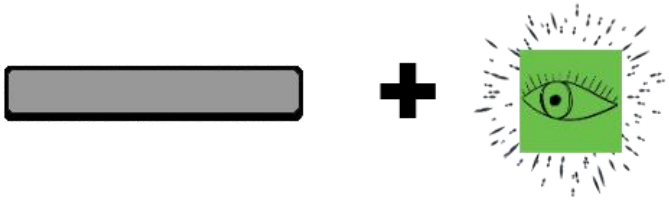


Figura 13: Representació dels controls de parry. Font pròpia.

Per poder executar de forma correcta aquest és necessari identificar quins projectils són els que permeten el *parry* i efectuar-lo quan la bala es trobi en un rang proper al jugador.

Per últim el es pot utilitzar certes habilitats. Aquestes es divideixen en tres subgrups:

- **Actives:** Habilitats que canvien el mode en el que el personatge combat contra els enemics. No són acumulatives així que el jugador ha de triar quina és la que més li convé.
- **Passives:** Habilitats que augmentaran les diferents estadístiques del jugador. Són acumulatives i permeten millorar el personatge fins un límit.
- **Especials:** Aquestes només podran ser utilitzades durant un curt període de temps. No seran acumulatives i per tant l'usuari haurà de triar quina vol tenir assignada. Poden utilitzar-se amb el boto *'E'*. Un cop gastades s'han de carregar eliminant enemics.

Explicades les habilitats, s'ha d'identificar i definir quines són aquestes per poder desenvolupar-les més endavant.

Habilitats actives

Hi ha 6, i funcionen de la següent manera:

1. **Doble tir:** Permet al jugador disparar dos projectils paral·lels en comptes d'un com ve predeterminat.
2. **Tir simultani:** L'usuari pot disparar una bala en la direcció escollida i una en la direcció contrària al mateix temps.
3. **Tir superior:** Les bales tindran una major mida i un major índex de dolor. Per altra banda el seu abast es inferior fet que porta al jugador a haver d'apropar-se més als enemics.
4. **Tir congelant:** Aquest permet congelar a l'enemic després d'encertar-l'hi un cert nombre de tirs, invalidant-lo d'atacar o moure's durant un període de temps.
5. **Minimum:** fa petit al jugador a la vegada que augmenta la seva velocitat al moure's disparar. Per altra banda el mal infligit als enemics serà menor.

6. **Maximum:** farà gran a l'usuari. Contràriament a la habilitat anterior la velocitat al moure's i disparar disminueix. També es veu disminuït l'abast. Per últim l'índex de dolor augmenta.

Aquestes només es poden trobar en sales especials on apareixen de manera aleatòria.

Taula 3: Habilitats actives. Font pròpia.

Habilitats passives

Com s'ha explicat, milloraran les diferents estadístiques del jugador. Aquestes són les següents:

- **Velocitat**: suma 0,5 a la velocitat del jugador. Aquest parteix amb una velocitat de 4 punts que podrà augmentar fins un màxim de 86
- **Vida:** afegeix un espai de vida addicional fins a un màxim de 5 i suma mitja vida.
- **Velocitat al disparar:** aquest factor determina la rapidesa amb el que el jugador crea una bala. Ve predeterminat amb una bala cada 0,4 segons i es pot baixar a 0,2. Va disminuint en rangs de 0,05.

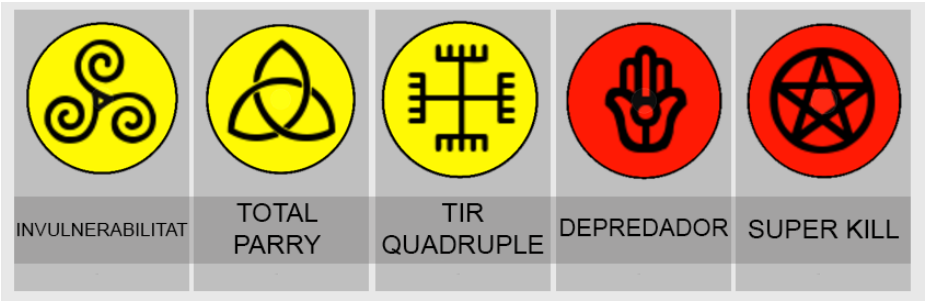
Taula 4: Habilitats passives. Font pròpia.

Aquestes funcionen de manera independent i són acumulatives. Es poden trobar en les sales especials de manera aleatòria o en les botigues del joc.

Habilitats especials

Aquestes habilitats només es poden utilitzar durant un període de temps determinat. Depenent de cada una té un cost i una funcionalitat diferent:

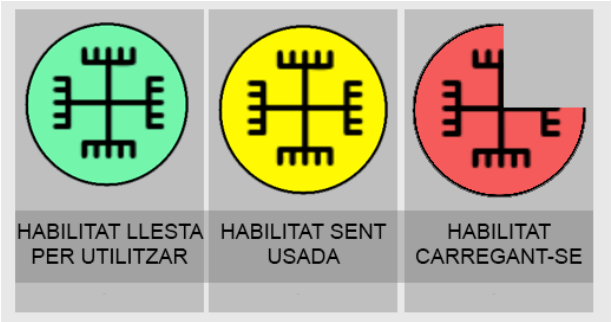
- **Invulnerabilitat:** Els atacs enemics no afecten al jugador; ni el contacte cos a cos ni les diferents bales enemigues.
- **Total Parry:** Tots els tirs que entren en contacte amb l'usuari reboten sense necessitat d'efectuar el *parry*.
- **Tir Quàdruple:** El jugador pot disparar simultàniament des dels quatre costats una ràfega de bales molt ràpida i letal.
- **Depredador:** L'usuari pot eliminar als enemics tocant-los. Té un cost de mitja vida al ser utilitzat.
- **SuperKill:** Tots els enemics que envoltin al jugador en un radi determinat moren instantàniament. Té un cost de mitja vida.



Taula 5: Habilitats especials. Font pròpia.

Aquestes habilitats només es poden desbloquejar aconseguint-les en les diferents botigues de cada pis o superant certs reptes especials. Quan s'aconsegueixi una apareixerà en el *HUD* del personatge.

L'habilitat podrà veure's de diferents colors segons el seu estat; en cas de trobar-se de color verd estarà llesta per ser utilitzada, en groc, marcarà que s'està utilitzant i per últim en vermell que s'està carregant. Per poder tornar-la a utilitzar serà necessari eliminar a deu enemics.



Taula 6: Estats de l'habilitat especial. Font pròpia.

Enemics

Els enemics conformen el principal repte del jugador i l'element que permet a aquest gènere funcionar tal com ho fa. Per aquest projecte es presenta una varietat diferent d'antagonistes que s'organitzen en 3 segons les seves capacitats.

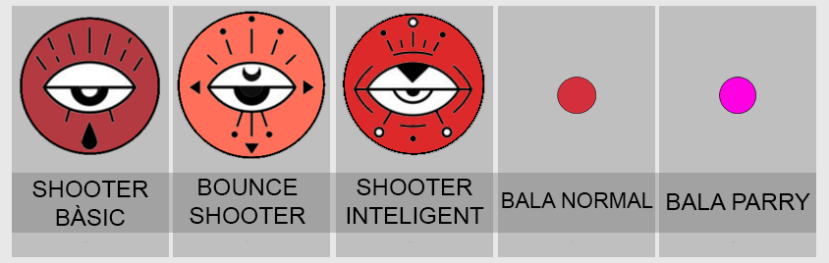
1. **Shooters:** enemics amb la capacitat de disparar. Són estàtics.
2. **Followers:** enemics que tractaran d'atacar l'usuari cos a cos.
3. **Specials:** aquest últim grup es basen en molestar mitjançant certes habilitats. Els seus atacs afecten el estat del jugador o a la seva vitalitat.

Shooters

Aquest subgrup d'enemics es divideix en tres segons les seves habilitats a l'hora de disparar. Cal remarcar que els tres consten de dues classes de bales diferents; la normal, que a l'impactar al jugador causarà dany i la bala *parry.* Aquesta última és d'un color diferent i pot ser rebotada pel jugador si s'efectua l'acció de *parry* en el moment adequat. En cas de no efectuar-lo causaria dany al jugador. Aclarit això, els diferents comportaments són els següents:

1. **Shooter bàsic**: dispara en una única direcció (en els eixos X o Y) segons la posició que ocupi.
2. **Shooter intelligent:** al disparar, a diferència de l'anterior, la bala apunta cap al jugador i per tant resulta més perillós que el *bàsic*.

3. **Bounce Shooter:** la bala apunta cap al jugador tal com fa l'*intelligent* amb la diferència que al contactar amb un mur rebotarà buscant a l'usuari. Pot donar fins a tres rebots abans d'eliminar-se.

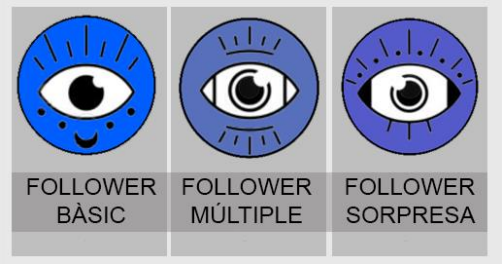


Taula 7: Diferents enemics Shooter i bales. Font pròpia.

Followers

Aquest subgrup es destaca gràcies a la seva principal característica; no són estàtics. Tots persegueixen al jugador per la sala per treure-li vida i eliminar-lo. Existeixen tres classes diferents segons el seu comportament:

1. **Follower bàsic:** persegueixen al jugador amb una velocitat normal.
2. **Follower múltiple:** persegueixen a l'usuari amb una velocitat més baixa que el *bàsic.* Al morir és divideix en dos fins a un màxim de dos cops. Amb cada subdivisió augmenta la velocitat.
3. **Follower sorpresa:** aquest té un comportament semblant al *bàsic* amb la diferència que al morir pot instanciar un enemic de classe *shooter*.



Taula 8: diferents enemics Followers. Font pròpia.

Specials

L'últim subgrup es destaca per les seves habilitats que tracten de molestar al jugador. Tots ells són estàtics. Existeixen tres de diferents:

1. **Special inverter:** si l'usuari s'apropa a una distància determinada aquest explota deixant unes espores que invertiran durant un període de temps els controls de moviment. A l'explotar l'enemic desapareix. No té efectes en la salut del jugador.
2. **Special squid:** de la mateixa manera que *l'inverter* explota a l'apropar-se el jugador. Aquest taca la pantalla i disminueix la visió durant un temps determinat. No té efectes en la salut de l'usuari.
3. **Special mother:** aquest, d'una mida més gran, crearà en períodes curts de temps petits enemics que perseguiran al jugador. Aquests, constaran només amb un punt de vida i per tant, són fàcils d'eliminar. En cas que el jugador s'apropi a una determinada distància de l'enemic gran, aquest explota i resta vida a l'usuari.

Taula 9: Diferents enemics Especials. Font pròpia.

Mon

Un cop explicats tant el jugador principal com els diferents antagonistes que apareixeran, s'ha de parlar del món on transcorre el videojoc. Com s'ha tractat en el marc teòric, el gènere *roguelike* es destaca pels mapes procedimentals, que busquen donar al jugador novetat en cada intent. Per aquest projecte s'implementa un sistema de sales procedimentals molt semblant al que podem trobar en la primera versió del videojoc *The Binding of Isaac.* Cada nivell consta amb un nombre determinat de sales (entre un mínim i un màxim) que va augmentant mentre el jugador va superant els nivells. La seva disposició és totalment aleatòria.

Per aconseguir aquesta estructura es parteix d'unes sales rectangulars que responen a la següent forma:

Figura 14: Imatge de la sala bàsica. Font pròpia.

Mitjançant aquesta, s'aconsegueix un total de 13 sales diferents només variant l'estat de les portes.

Per formar el món i distribuir els seus elements, es troben diferents classes de sales amb vàries funcions. Aquestes són les següents:

- **Sala normal:** buides o amb decoració. Serveixen de descans per al jugador.
- **Sala d'enemics:** en aquestes es troben els diferents antagonistes del joc. Per afegir varietat en el món, existeixen 3 classes diferents:

    - **Sala de *Shooters*:** només els *shooter* podran aparèixer en aquestes a excepció del *special inverter i special squid*. La sala segueix la següent organització:

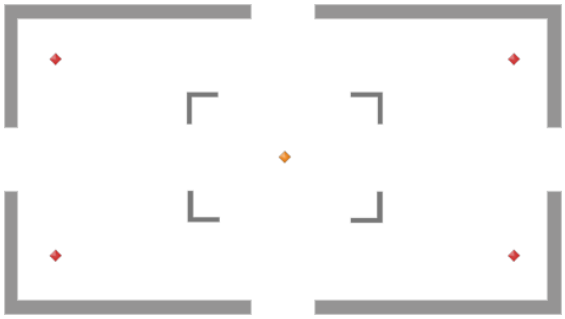


Figura 15: Imatge sala d'enemics de Shooter. Font pròpia.

Per entendre aquest esquema s'ha de revisar la definició dels diferents elements que la formen en la taula 10:

ELEMENT	CLASSE	DEFINICIÓ
	Punt d'enemic (*shooter)*	De forma aleatòria es determina si apareix un enemic. En cas afirmatiu, pot ser un dels tres *shooters* (bàsic, intelligent o *bounce*).
	Punt d'enemic 2 (*shooter* + *special)*	De forma aleatòria es determina si apareix un enemic. En cas afirmatiu, pot ser un *shooters* (només intelligent o *bounce*) o un *special* (només *inverter* o *squid*).
	Obstacle	De forma aleatòria es decideix si apareix l'obstacle. Aquest tracta de donar novetat en la sala.

Taula 10: Diferents elements de la sala Shooter. Font pròpia.

- **Sala de *Followers*:** Només poden aparèixer enemics de tipus *Followers* a excepció dels *Special*. L'organització de la sala segueix l'esquema següent:

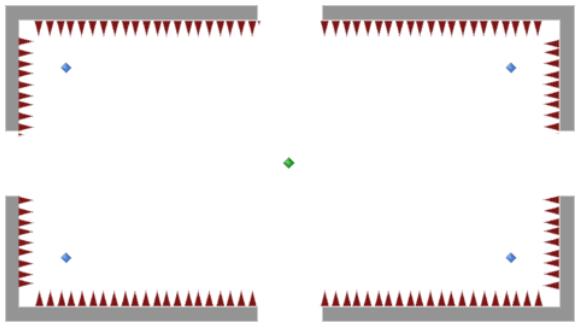


Figura 16: Imatge de la sala de Followers. Font pròpia.

Per entendre aquest esquema s'ha de revisar la definició dels diferents elements que el formen en la taula 11:

ELEMENT	CLASSE	DEFINICIÓ
	Punt d'enemic 3 (*follower)*	De forma aleatòria es determina si apareix un enemic. En cas afirmatiu, pot ser un dels tres *followers* (bàsic, múltiple o sopresa).
	Punt d'enemic 4 (*special*)	De forma aleatòria es determina si apareix enemic. En cas afirmatiu, pot ser un dels tres *special* (*inverter, squid* o *mother*).
	Obstacle	De forma aleatòria es decideix si apareix l'obstacle que rodejarà les parets de la sala. Si el jugador toca les pues perd vida.

Taula 11: Diferents elements en la sala Follower. Font pròpia.

- **Sala Comuna:** Aquesta sala permet l'aparició de les tres classes d'enemics. Segueix l'organització següent:

Figura 17: Imatge de la sala comuna. Font pròpia.

Per entendre aquest esquema s'ha de revisar la definició dels diferents elements que el formen en la taula 12:

ELEMENT	CLASSE	DEFINICIÓ
	Punt d'enemic 5 (*shooter* + *follower)*	De forma aleatòria es determina si apareix un enemic. En cas afirmatiu, pot ser un dels tres *shooters* (bàsic, intelligent o *bounce*) o un dels tres *followers* (bàsic, múltiple o sorpresa).
	Punt d'enemic 6 (*shooter* + *special)*	De forma aleatòria es determina si apareix un enemic. En cas afirmatiu, pot ser *shooters* (només intelligent o *bounce*) o *special* (només *mother*).

Taula 12: Diferents elements que formen la sala comuna. Font pròpia.

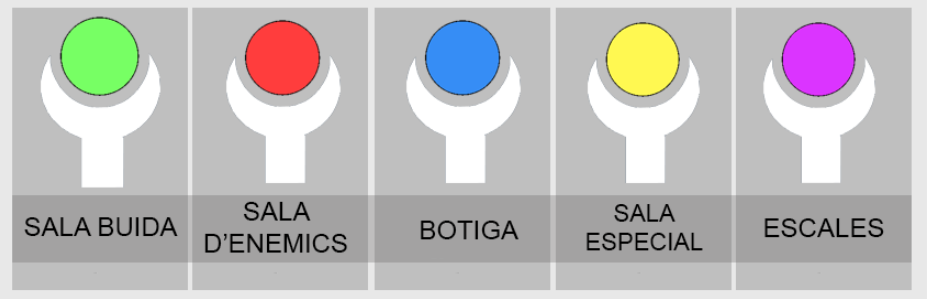
Cal destacar que totes les sales d'enemics, independentment de la classe, tenen dos estats que es determinen de forma aleatòria:

- **Sala oberta:** el jugador pot abandonar la sala sempre que vulgui.
- **Sala tancada:** un cop l'usuari estigui dins, les sortides es bloquegen fins que tots els enemics siguin eliminats.

- **Sala Botiga:** aquesta només es troba un cop per pis. En ella es pot comprar diferents objectes amb monedes (s'explica amb detall més endavant). Els diferents elements que es troben en la botiga són els següents:
  - **Habilitat especial:** es decideix de manera aleatòria una de les cinc. El preu és de 20 monedes.
  - **Habilitat passiva:** de la mateixa manera que amb l'especial, es pot trobar una de les tres possibles. El preu es de 10 monedes.
  - **Vida:** aquest ítem sumarà una vida al nostre jugador. No confondre amb l'habilitat passiva que afegeix un espai de vida addicional. El preu és de 5 monedes.
- **Sala Especial:** aquesta, tal com la Botiga, només apareix un cop per pis. Permet al jugador aconseguir una de les cinc habilitats actives o una de les tres passives de forma gratuïta i aleatòria. També es troba un mag que a canvi de dos cristalls et proporciona una habilitat d'especial.

- **Sala Final:** en aquesta es troba les escales que porten al jugador al següent pis. Només apareix una per nivell.

Per identificar les diferents sales a la vegada que es guia al jugador pel món, s'estableix un sistema de llums que es col·loca a les portes de les sales. Aquestes apareixen tancades i s'encenen quan el jugador passar per la porta en qüestió. D'aquesta manera, l'usuari pot identificar quines sales ha visitat. Per altra banda, les llums tenen diferents colors que determinen quina és la sala que es troba a l'altra banda. Donat que existeixen cinc classes diferents, s'obté la següent organització:

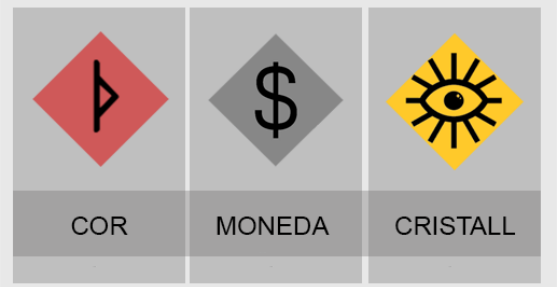


Taula 13: Diferents colors de les llums segons la sala que precedeixen. Font pròpia.

Objectes

Durant el joc existeixen certs elements que el jugador pot trobar o aconseguir mentre navega pel mapa. Aquests serveixen tant per intercanviar com per afectar el estat del jugador:

- **Cor:** sumen a l'usuari mig punt de vida. Es poden trobar a la botiga o eliminant enemics.
- **Moneda:** serveixen únicament per la botiga. Es poden aconseguir en missions o al matar enemics.
- **Cristall:** aquest element només apareixen un cop per pis. Serveix per intercanviar per una habilitat especial amb el mag.



Taula 14: Classes d'objectes. Font pròpia.

Missions

El sistema de missions és un altre afegit al projecte que permet al jugador completar certs reptes a canvi d'una recompensa. Les missions base del videojoc són les següents tres:

1. Matar a cert nombre d'enemics (que s'estableix de manera aleatòria). La recompensa són cinc monedes.
2. Efectuar correctament un cert número de *parrys* (el número també és aleatori). La recompensa són cinc monedes.
3. Visitar totes les sales del nivell actual. La recompensa són set monedes.

Aquestes missions es poden veure en el menú de *Pausa*. Allí es troba la definició, el número de sales/ *parrys*/ enemics que s'ha d'assolir i el nombre de sales/ *parrys*/ enemics que es porta. També es veu la recompensa.

Per altra banda només pot estar una missió activa. Fins que aquesta no s'hagi superat, no apareix un altre. L'elecció de la missió activa és completament aleatori. Per últim, cada cop que el jugador passa de nivell aquestes missions es reinicien, amb nous valors i ordre.

HUD i menús

En el joc es troben diferents menús que donen informació al jugador sobre el seu estat, nivell en el qual es troba, habilitats que té, etc... Aquest es divideixen en els següents:

- **HUD de pantalla:** En ell s'estableix les vides del jugador (tant els espais disponibles com la vida en si) i l'estat de l'habilitat especial, en cas de tenir alguna assignada.
- **Menú de Pausa:** Es pot obrir mitjançant el botó '*P'*. S'organitza en tres parts segons la informació que dóna al jugador:
    - Part dreta: Apareixen les diferents missions amb la informació prèviament explicada.
    - Part central: Es mostren els diferents controls del videojoc.
    - Part esquerra: Es troba totes les dades relacionades amb el jugador principal, tals com: la vida, la velocitat, el temps de dispar, les monedes, el nombre de cristalls, nivell actual i l'habilitat passiva i especial en cas de tenir alguna.
- **Menú de càrrega:** Apareix al principi de cada nivell. En ell es pot trobar l'objectiu principal del joc en angles: "*Are you able to reach to the final level?*". D'aquesta manera es recorda al jugador quina és la meta a superar. També et mostra el nivell en què et trobes actualment i un *Short Time Horizon.* Aquest ensenya els nivells que s'han superat fins ara de forma semblant a una línia cronològica. També apareix el personatge principal posicionat en el nivell actual.
- **Menú de Game Over:** Surt quan l'usuari perd la partida. Apareix també una frase que variarà segons el nivell que hagi assolit el jugador. Des d'aquest menú es podrà seleccionar l'opció de tancar el joc o tornar a intentar-ho.
- **Feedback HUD:** És l'encarregat de donar diversa informació a l'usuari mitjançant la pantalla principal. Aquests missatges poden referir-se a missions superades, nous elements desbloquejats, cristalls trobats, etc...
- **Menú principal:** Es mostra en començar el joc. Permet començar un intent o sortir de l'aplicació.

- **Menú d'informació:** Ensenya totes les dades de l'algoritme; els diferents indicadors, el percentatge dels diferents perfils de joc la posició en l'eix de coordenades que ocupa.
- **Menú Final:** Es mostra a l'arribar al nivell 10. En ell apareix el perfil que l'algoritme ha escollit segons les teves accions en el joc i una petita definició d'aquest. També es pot seleccionar tornar a jugar o sortir del videojoc.

Disseny de les mecàniques dels diferents perfils

Un cop establert els diferents elements bàsics que existeixen en el joc, s'ha de buscar quines mecàniques apareixeran segons el perfil d'usuari de cada jugador. Per enfrontar tal repte, s'ha d'aclarir les diferents possibilitats que poden aparèixer un cop un usuari estigui jugant. Tenint en compte l'eix d'abscisses anteriorment establert, existeixen els següents dos casos:

- **Jugador amb perfil únic:** l'algoritme estableix que l'usuari es troba en una única regió, i per tant, adquireix únic perfil.
- **Jugador amb perfil mixt:** usuari que donat la seva manera de jugar es posiciona en un espai molt proper a dos perfils.

Tenint en compte aquests dos casos s'ha de dissenyar no només pensant en els jugadors que es decanten cap a una sola regió, sinó tenint en compte perfils mixtos.

Aclarit aquest punt, s'ha enfocat el disseny de les mecàniques de la següent manera: cada perfil consta de dos nivells segons el grau d'implicació del jugador. D'aquesta manera permet al videojoc establir que un jugador que tendeix lleugerament cap al perfil *Creative* la vegada que s'apropa al *Achivment*, tingui només el grau de nivell 1 en les mecàniques de cada perfil.

Amb tot això, i segons les preferències de cada taxonomia, s'estableix els següents nivells:

Mecàniques per jugadors Action

Jugadors que es troben en la zona que engloba *acció* i *gameplay.* Predilecció per les accions ràpides, el combat, feedback constant del que estan fent i interacció amb el sistemes que el joc proposa. Per aquest perfil s'ha establert els següents dos nivells:

- **Nivell 1:** Els enemics *Followers* deixen anar unes taques que redueixen la velocitat del jugador quan es troba en contacte amb elles. D'aquesta manera es vol donar novetat i dificultat al *gameplay* prèviament dissenyat.
- **Nivell 2:** Els percentatges de sales tancades, sales d'enemics, i la probabilitat de crear un enemic augmenten al 85% (vénen predeterminats al 50%). Amb això es vol donar molta importància al combat.

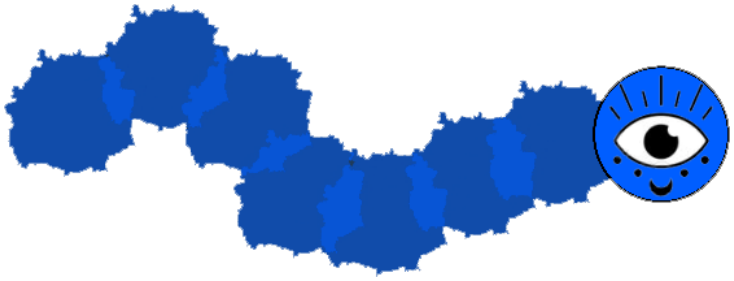


Figura 17: Representació del nivell 1 del perfil Action. Font pròpia.

Mecàniques per jugadors Maestry

Jugadors que es troben en la zona que engloba *interacció* i *gameplay*. Tenen una relació mútua o simbiòtica amb els elements del joc i predilecció per polir al màxim el seu estil. Per aquest perfil es proposa els següents dos nivells:

- **Nivell 1:** Els enemics *Shooters* (a excepció del bàsic) es creen amb un escut que impedeix que les bales del jugador encertin. Per eliminar aquest, és necessari encertar una bala enemiga rebotada amb el *parry*. Per facilitar el procés s'augmenta el percentatge de bales rebotables que disparen els antagonistes. Amb això es vol potència la mecànica de *parry* que requereix domini i coordinació per part de l'usuari.

- **Nivell 2:** Els enemics augmenten el dany envers el personatge. Amb aquests canvis, la dificultat del videojoc s'eleva considerablement a un nivell que només els usuaris més hàbils poden superar.

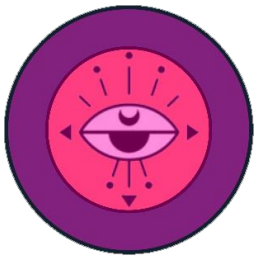


Figura 18: Representació del Parry Shield, nivell 1 del perfil Maestry. Font pròpia.

Mecàniques per jugadors Achievement

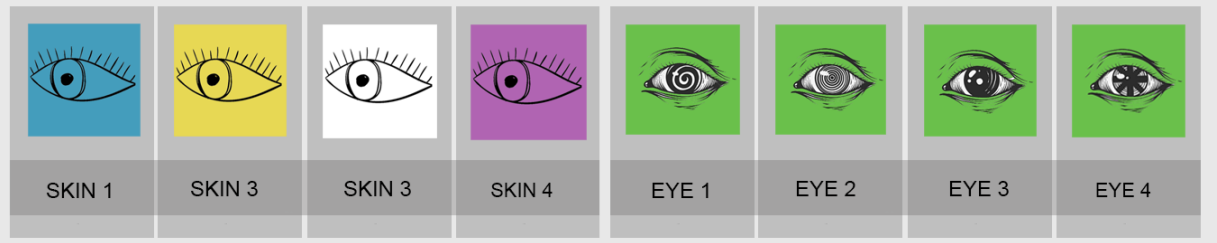
Perfil que es troba en la zona que engloba *acció* i *món*. Jugadors amb l'objectiu d'aconseguir una recompensa amb la resolució de tasques, reptes o missions. Per aquest, s'estableix els dos següents nivells:

- **Nivell 1:** apareix un nou repte que el jugador pot resoldre a canvi d'una recompensa. Aquestes noves missions (que s'anomenen *Hard missions)* es divideixen en tres:

    1. Superar el nivell sense perdre vida. Amb una recompensa de 7 monedes.
    2. Superar el nivell abans d'un temps marcat. Amb una recompensa de 7 monedes.
    3. Trobar la botiga o sala especial abans d'un temps marcat. Amb una recompensa de 7 monedes.

    Només apareix una missió per nivell i es determina aleatòriament. Per altra banda, tenen la particularitat de poder fallar-se, i en tal cas, no es suma cap recompensa. Per poder accedir a un altre *Hard mission* si no s'ha superat s'ha d'esperar fins el següent nivell.

Per últim es desbloquegen dos nous ítems a la botiga que permeten canviar el color del personatge principal i el seu dibuix. Els colors possibles són: blau, groc, blanc i lila mentre que de tatuatges existeixen quatre models diferents. El preu és de 8 monedes per cada un dels productes.



Taula 13: Diferents estils i colors que pot adoptar el personatge principal. Font pròpia.

- **Nivell 2:** es desbloqueja un nou model de missions anomenades *Large missions* que presenten els següents reptes al jugador:

    1. Millorar al màxim la velocitat del personatge.
    2. Millorar al màxim la vida del personatge.
    3. Millorar al màxim el temps de dispar del personatge.

    Les tres tenen una recompensa de 10 monedes donada la seva dificultat. També, i seguint la lògica de les altres missions, apareix aleatòriament una. No es reinicien amb el canvi de nivell i tan sols es pot accedir a un altre superant l'actual.

Mecàniques per jugadors Creativity

Perfil que es troba en la zona que engloba *interacció* i *món.* Són jugadors amb l'objectiu d'investigar tant el món que construeix el videojoc com els seus sistemes. Per aquest perfil es té els següents dos nivells:

- **Nivell 1:** Apareix un nou mag en la sala especial, que per dos cristalls et permet aconseguir canviar de color o dibuix de manera totalment gratuïta.

- **Nivell 2:** Apareix una nova mecànica que permet al jugador, mitjançant el botó “M”, mirar un mapa. Aquest ensenya només les sales que el jugador ha visitat i ajuda a guiar-se per les diferents sales del món. També augmenta la probabilitat que els enemics deixin anar vides o diners al morir.

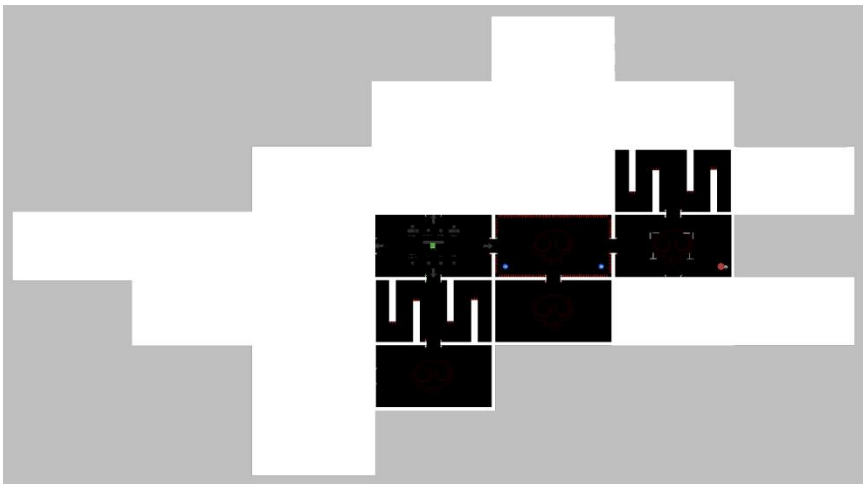


Figura 19: Mapa del joc, nivell 2 del perfil Creativity. Font pròpia.

Disseny del algoritme

Arribat a aquest punt només queda determinar com es plantejarà l’elecció del perfil mentre l’usuari estigui jugant. Per això s’ha de tenir en compte els diferents eixos que prèviament s’han establert per organitzar les diferents taxonomies. Es recorda que són els següents:

- **Eix X:** contraposició entre jugadors que tenen predilecció pel Món contra els que prefereixen el Gameplay.
- **Eix Y:** contraposició entre jugadors que prefereixen l’Acció contra els que es senten més atrets per la Interacció amb el joc.

Amb això, i si es té en compte els eixos com una representació matemàtica de quelcom, s'aconsegueix englobar cada perfil segons dos valors. Considerant aquesta visió, s'obté com a resultat el següent esquema:

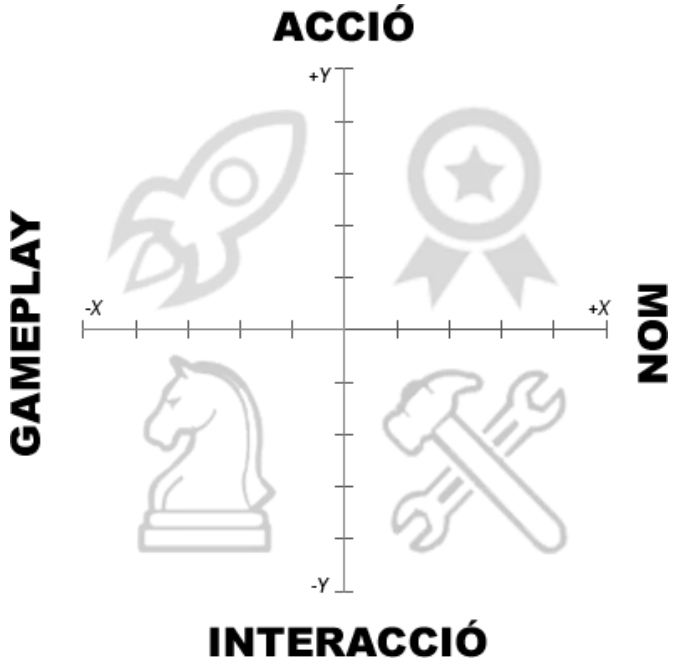


Figura 10: Representació de les diferents taxonomies en un eix de coordenades. Font pròpia.

Cada taxonomia compta d'un valor X i un valor Y (positius o negatius tenint en compte la zona) que determinen la posició exacta del jugador. Per aconseguir-los, es considera que cada eix compta amb certs indicadors que retornen un percentatge. Aquests només engloben l'apartat positiu, ja que s'entén que si *Món* (per tant x positiva) és 84%, s'obtindria que *Gameplay* és un 16%. Cal aclarir també que si *Món* i *Gameplay* són iguals al 50%, el jugador presentaria la mateixa predilecció per ambdues bandes, i per tant, es considera que X = 0.

Amb aquestes característiques es pot resumir que les formules per trobar tant el valor X com Y segueix la següent lògica:

$$MON\ (\%\ global) = \frac{\sum, indicadors X}{Número\ total\ d'indicadors\ X} \cdot 100 \qquad (1.1)$$

$$GAMEPLAY = (100 - MON) \qquad (1.2)$$

$$x = \left(\frac{MON}{2}\right) \cdot \left(\frac{-GAMEPLAY}{2}\right)$$

(1.3)

En la *(1.3)* és important que el segon valor, en aquest cas *Gameplay*, sigui negatiu, ja que es troba en la part negativa de l'eix de coordenades. Amb aquestes fórmules s'obté un valor entre 0 i 50 que determina el grau cap a què tendeix el jugador en cada eix.

Aclarides les fórmules, s'ha de definir els diferents indicadors que es fan servir per cada eix. Aquests, que es poden veure en la taula 13, es tenen en compte cada dos nivells, ja que es considera que amb un de sol és molt difícil poder determinar certs valors. Cada cop que s'efectuï el càlcul de coordenades el jugador es posicionarà en una zona determinada i per tant desbloquejarà els diferents nivells que s'han presentat en l'apartat anterior.

MON (x positiva)	**GAMEPLAY (x negativa)**
1. Número de sales visitades / número de sales totals	1. Número de **NO** sales visitades / número de sales totals
2. Número de cristalls trobats / número de cristalls totals	2. Número de cristalls **NO** trobats / número de cristalls totals
3. Diners aconseguits / Diners totals que pots aconseguir	3. Diners **NO** aconseguits / Diners totals que pots aconseguir
4. Compres efectuades (diners gastats) / Total de diners del personatge	4. Compres **NO** efectuades (diners estalviats) / Total de diners del personatge
5. Habilitat especial aconseguida (100 % **SI**)	5. Habilitat especial aconseguida (100 % **NO**)
ACCIÓ (y positiva)	**INTERACCIÓ (y negativa)**
1. Número d'enemics eliminats / número d'enemics totals	1. Número d'enemics **NO** eliminats / número d'enemics totals
2. Número de *parrys* **MAL** efectuats / número total de *parrys* efectuats	2. Número de *parrys* ben efectuats / número total de *parrys* efectuats
3. Número total de vides **PERDUDES** / número de vides possibles	3. Número total de vides **ACTUALS** / número de vides possibles
4. Número de bales **NO** encertades / número de bales totals disparades	4. Número de bales encertades / número de bales totals disparades
5. Sales d'enemics resoltes (tots els enemics eliminats) / Total de sales d'enemics	5. Sales d'enemics **NO** resoltes (tots els enemics eliminats) / Total de sales d'enemics

Taula 14: Indicadors de cada eix. Font pròpia.

Tenint en compte que es parteix amb un perfil (0,0) i que per tant cap d'aquestes es troba activada, es contemplen els següents casos:

- **Cas 1:** El jugador tendeix cap a un perfil de manera lleugera. Per entendre el cas millor s'exposa el següent exemple:

  Si s'obté un valor de X = -6 i Y = 15, el jugador es troba en el perfil de *Action.*

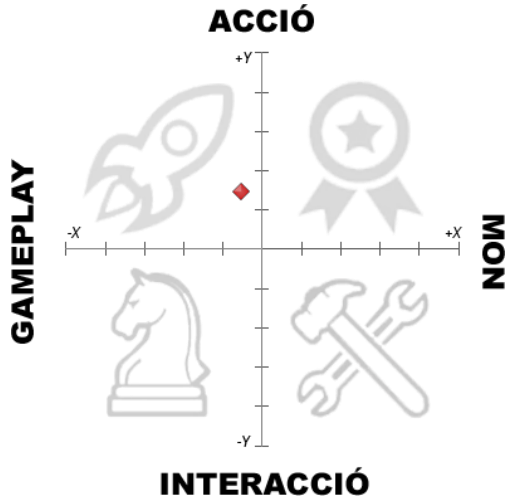


Figura 21: Eix de coordenades pel Cas 1. Font pròpia.

Donada la posició, es desbloqueja el nivell 2 de *Action*. Per altra banda, ja que es segueix estant a prop dels eixos de coordenades també s'obté el nivell 1 de *Achievment* i el nivell 1 de *Maestry.*

- **Cas 2:** El jugador tendeix cap a un perfil lleugerament a la vegada que s'allunya d'un dels dos eixos. Exemple:

  Si s'obté un valor de X = -15 i Y = 28, el jugador es troba en el perfil de *Action.*

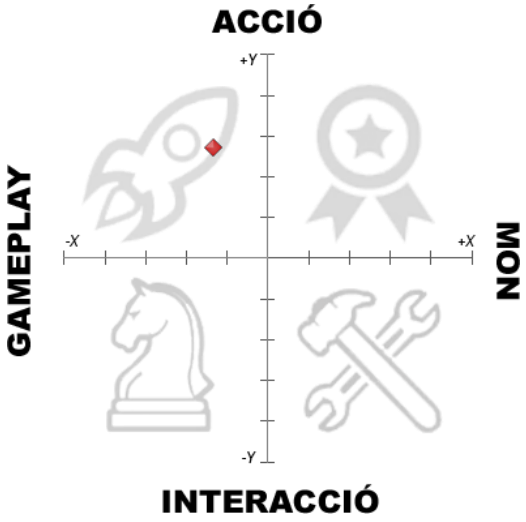


Figura 12: Eix de coordenades pel Cas 2. Font pròpia.

Ja que el valor en l'eix X es superior a 25, es considera que s'allunya de *Maestry.* En aquest cas s'obté el nivell 2 de *Action* i només el nivell 1 de *Achievment.*

- **Cas 3:** El jugador tendeix lleugerament cap a dos perfils. Exemple:

  Si s'obté un valor de X = -1 i Y = 12, el jugador es troba en el perfil de *Action.*

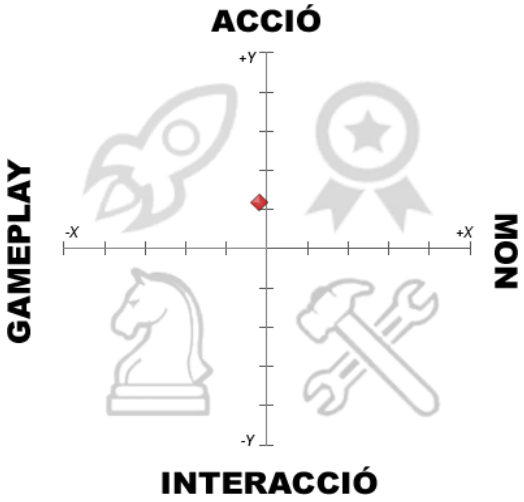


Figura 23: Eix de coordenades pel Cas 3. Font pròpia.

Com s'observa el valor X és menor que -5, i per tant molt proper l'eix de coordenades. Aquí, de la mateixa manera que el cas anterior, només es desbloqueja el nivell 2 de *Action* i l'1 de *Achievment.*

- **Cas 4:** El jugador tendeix fortament cap a un perfil. Exemple:

  Si s'obté un valor de X = -32 i Y = 28, el jugador es troba en el perfil de *Action.*

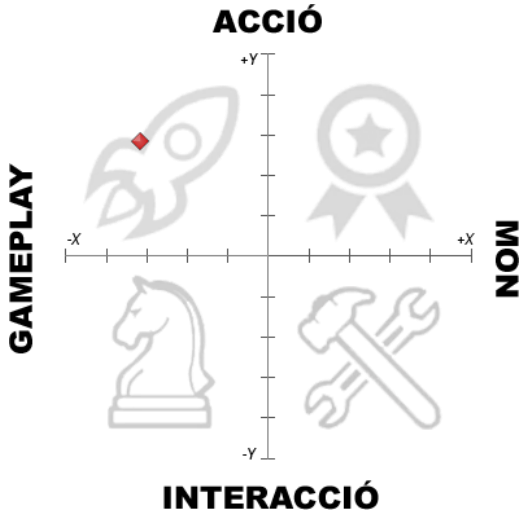


Figura 24: Eix de coordenades pel Cas 4. Font pròpia.

Donat que ambdós valors superen el 25, es considera que el jugador té un perfil únic i per tant només es desbloqueja el nivell 2 de *Action*.

DESENVOLUPAMENT

Un cop s'ha dissenyat tots els aspectes del videojoc, s'ha de desenvolupar de manera que tot el prèviament explicat es vegi reflectit en un projecte jugable. Per portar a terme tal feina s'utilitza el motor de de *Unity*, en concret la versió *2020.2*. Gràcies a les seves característiques es pot crear de forma dinàmica i ràpida tots els elements que es precisen. Els diferents blocs que s'han portat a terme es presenten en el mateix ordre que es troben en l'apartat de disseny.

Jugador principal

Per la creació del personatge, segons les diferents característiques prèviament definides, és necessari que contingui els següents elements:

- **Codi principal** (*ProtoPlayerScript)*: encarregat de controlar tant el moviment com els diferents estats que aquest pot tenir.
- **Player Blackboard** (*ProtoBLACKBOARD_Player*): emmagatzemador de les diferents variables (velocitat, vida màxima, monedes actuals...), estats (habilitat passiva, habilitat especial...), *skins* (color, dibuix, mida...) i tot allò que té a veure amb el jugador. Sempre que el codi principal o altres elements del joc necessitin accedir a qualsevol dada del personatge, s'ha d'extreure d'aquí.
- **Player HUD** (*HUD_MANAGER*): controlador dels diferents elements del videojoc que apareixen en els menús; vida per pantalla, habilitat especial seleccionada, menú de Pausa, etc...
- **Player Sound Manager** (*PlayerSoundManager*): encarregat d'emmagatzemar els diferents sons que pertanyen al jugador: so al disparar, al rebre mal, al aconseguir quelcom...
- **Collider 2D:** element que permet a *Unity* detectar les col·lisions entre els diferents elements del projecte.

- **RigidBody 2D:** característica que atorga al personatge les físiques per poder tenir moviment. Al ser 2D només permet que es mogui en els eixos X i Y.
- **Etiqueta** (tag)**:** paraula de referència que es pot assignar a un element o un conjunt i que permet la seva diferenciació respecte als altres a la vegada que facilita el seu accés per altres codis o entitats. En aquest cas s'assigna l'etiqueta "Player".
- **Transform:** Element comú en tots els objectes de *Unity*. Permet controlar la posició, la rotació i l'escala d'aquest.
- **Sprite Renderer:** Component que permet assignar una imatge i controlar les seves característiques.

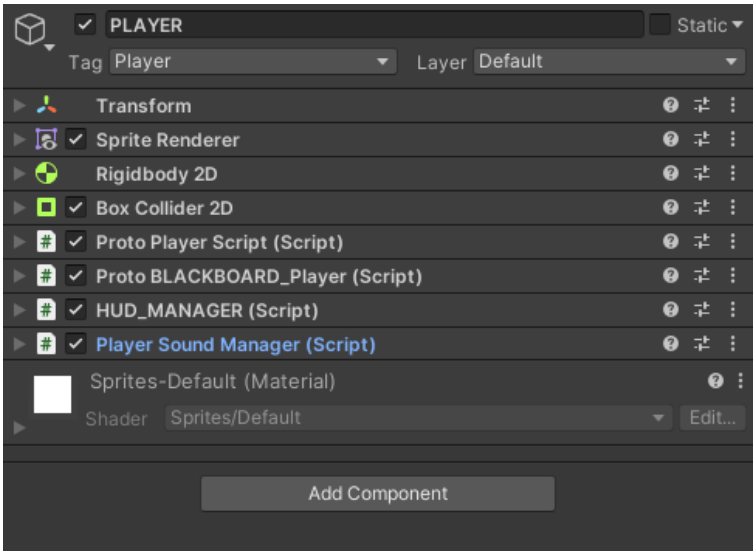


Figura 25: Captura de Unity amb els elements que componen al Player. Font pròpia.

A part de tot això, *Unity* permet relacionar diferents elements independents que funcionen a mode de fills que, en aquest cas, el personatge controla. Les seves funcions són facilitar altres comportaments del jugador. Aquests són:

- **Bullet points:** Guarda les diferents posicions des d'on es creen les bales al ser disparades.
- **Parry Collider:** encarregat de detectar les bales rebutjables.
- **Parry Particles:** partícules que es creen a l'efectuar el *parry.*
- **Player Draw Skin:** dibuix del qual consta el personatge.

- **SuperKill Collider:** responsable de detectar els diversos enemics que el personatge té al voltant.

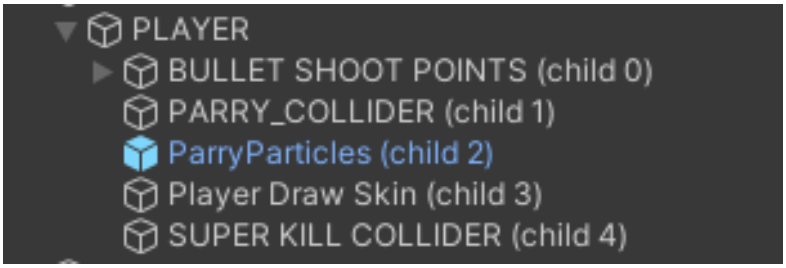


Figura 26: Captura de Unity amb els elements que complementen al Player. Font pròpia.

Definits els diferents elements que formen el jugador principal s'ha d'aclarir el seu funcionament més detalladament.

Moviment

Aquest funciona gràcies a una funció que relaciona el component *RigidBody2D* amb els diferents controls que s'han assignat. Al clicar les tecles, un Vector2 guarda la posició en la direcció escollida i ho suma a la posició actual del jugador a la vegada que es multiplica per la velocitat i el temps.

Aquesta funció es crida a la funció Update i es veu de la següent manera:

```
//MOVMENT
1 reference
void Movment()
{
    movment.x = Input.GetAxisRaw("Horizontal"); // A o D
    movment.y = Input.GetAxisRaw("Vertical");// W o S
    rb2d.MovePosition(rb2d.position + movment * speed * Time.fixedDeltaTime);
}
```

Figura 27: Captura del codi ProtoPlayerScript() amb la funció Movment(). Font pròpia.

Dispar

Aquest comportament s'efectua mitjançant les fletxes del teclat. Al ser clicades, el codi crida la funció *BasicShoot*() (o una variant depenent de l'habilitat activa) que instància una classe de bala en una direcció marcada.

```
void BasicShoot(bool shootInXAxe, bool isVelocityPositive, GameObject typeOfBullet)
```

Figura 28: Captura del codi ProtoPlayerScript() amb la funció BasicShoot(). Font pròpia.

Donat que existeixen diferents classes de dispar segons l'habilitat, un controlador s'encarrega de seleccionar quina és la funció que ha de cridar i quines variables ha de passar. Aquest comportament és possible mitjançant un variable *float* anomenada *habilityType* (emmagatzemada a la *Blackboard* del personatge) i una estructura de tipus *switch.*

```
//CONTROLS OF SHOOTING
1 reference
void ShootController()
{
    if (Input.GetKey(KeyCode.UpArrow) && !quadShoot)
    {

        switch (BlackBoardPlayer.habilityType)
        {
            case 0: BasicShoot(true, true, BlackBoardPlayer.Bullet); break;
            case 1: DobleShoot(true, true); break;
            case 2: SimultaneousShoot(true, true); break;
            case 3: BasicShoot(true, true, BlackBoardPlayer.superiorBullet); break;
            case 4: BasicShoot(true, true, BlackBoardPlayer.freezeBullet); break;
            case 5: BasicShoot(true, true, BlackBoardPlayer.minimumBullet); break;
            case 6: BasicShoot(true, true, BlackBoardPlayer.maximumBullet); break;
                //...
        }
    }
```

Figura 29: Captura del codi ProtoPlayerScript() amb la funció ShooterController(). Font pròpia.

Cada bala conté el seu propi codi que controla el moviment, velocitat, dany, etc... Per tant, un cop el jugador n'ha creat una, és ella mateixa qui s'encarrega del comportament d'aquesta.

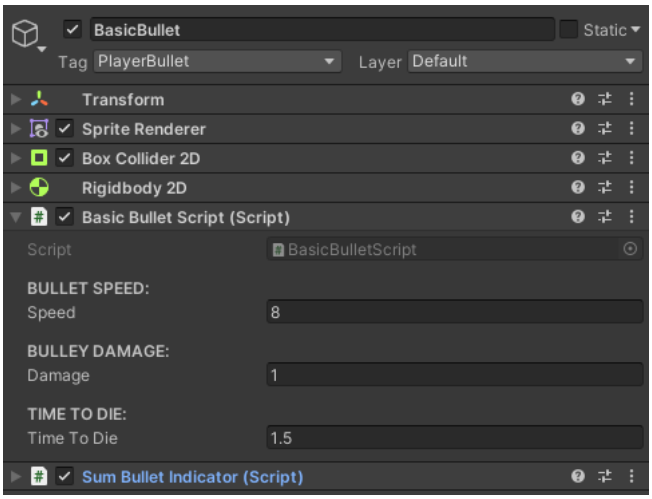


Figura 30: Captura de Unity dels diferents components de BasicBullet. Font pròpia.

Parry

El *parry* es pot efectuar amb el botó "SPACE". Aquest crida a la funció *Parry()* que activa durant un temps determinat el component *Parry Collider* i *Parry Particles*. Existeix cert temps d'espera per no sobrecarregar aquesta habilitat.

```
void ParryController()
{
    if (Input.GetKey(KeyCode.Space) && parryTimer >= 0.5)
    {
        Parry();
    }
    else
    {
        BlackBoardPlayer.p_Collider.gameObject.SetActive(false);
        p_timer = 0;
    }
}
```

Figura 31: Captura del codi ProtoPlayerScript() amb la funció ParryController(). Font pròpia.

El rebot de les bales s'efectua en el seu propi *script*.

Habilitats

Donat que existeixen diferents classes d'habilitats cada una consta d'un controlador, a excepció de les passives. Les habilitats actives s'han explicat en l'apartat de *Dispar*.

Les habilitats passives sumen les diferents estadístiques del personatge segons les característiques explicades en l'apartat de *Disseny*. Es calculen en el component *OnTriggerEnter2D*, encarregat de les col·lisions del jugador mitjançant el sistema de *tags*. Amb un *switch(),* tria quina estadística ha de canviar:

```
switch (other.GetComponent<PassiveHabilityScript>().estadisticType)
{
    case 1: speedSum = speedSum + 0.5f; loadingHability = true; loadingHabilityTimer = 0;
            Destroy(other.gameObject); hudManager.ActiveSpeedPLusFeedback(); break;
    case 2: delaySum = delaySum - 0.05f; loadingHability = true; loadingHabilityTimer = 0;
            Destroy(other.gameObject); hudManager.ActiveDelayPLusFeedback(); break;
    case 3: SumLife();  hudManager.ActiveLivePLusFeedback();
            if(BlackBoardPlayer.characterSpaceLife < 5)
            { BlackBoardPlayer.characterSpaceLife = BlackBoardPlayer.characterSpaceLife + 1f; }
            loadingHability = true; loadingHabilityTimer = 0;
            Destroy(other.gameObject); break;
}
```

Figura 32: Captura del codi ProtoPlayerScript() amb la funció que detecta i assigna les diferents habilitats passives. Font pròpia.

Per últim, les habilitats especials consten d'un controlador que s'organitza de la mateixa manera que el de les actives. Depenent de quina posseeixi el jugador, i si aquest l'ha activat utilitzant el botó "E", crida una de les diferents funcions que s'encarreguen de tals comportaments.

```
switch(BlackBoardPlayer.specialStateType)
{
    case 0: WaitingForNextSpecialHability(); break;
    case 1: InvencibleState(); break;
    case 2: SuperParry(); break;
    case 3: QuadShoot(); break;
    case 4: HunterState(); break;
    case 5: SuperKill(); break;
}
```

Figura 33: Captura del codi ProtoPlayerScript() amb la funció SpecialHabilityController(). Font pròpia.

Altres comportaments

El personatge també s'encarrega dels controls del teclat restants (menú de Pausa, mapa, informació, música ...) i mitjançant el seu component *Collider2D*, de totes les col·lisions que l'impliquen: contacte amb els enemics, detecció de monedes o cors, detecció de les escales per canviar de nivell, etc... Tots aquests funcionen de manera semblant al prèviament explicat. En el cas dels menús, es crida als diferents components guardats en *HUD_Manager*.

Enemics

Tot enemic, igual que el jugador principal, consta de certs components bàsics a excepció del script principal, que varia segons la classe que sigui. Aquestes són els següents:

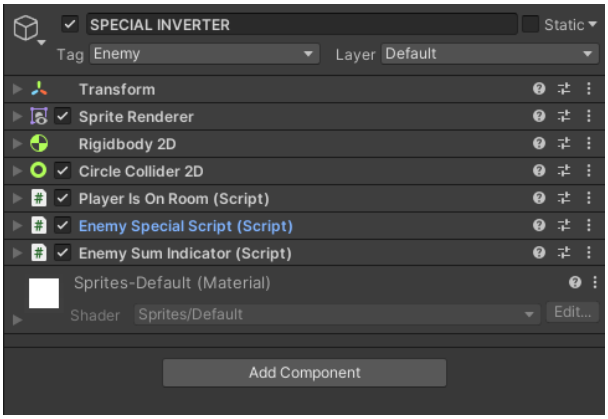


Figura 34: Captura de Unity amb els diferents elements dels enemics. Font pròpia.

A part dels elements ja coneguts, tenim dos scripts nous:

- *IsPlayerOnRoom*(): detecta si el jugador es troba a la sala on està.
- *EnemySumIndicator()*: suma 1 a un comptador que guarda el nombre d'enemics que existeixen en cada nivell.

A partir d'aquí, cada grup consta d'elements especials i únics per complementar els comportaments que aquest necessita.

Shootters

Cada *shooter* compta amb elements comuns que s'utilitzen només en cas que aquest o necessiti. Els elements són:

- Wall Checkers: *Colliders* que s'encarreguen de detectar la posició relativa dels *shooterBasic* i així evitar dispars en l'eix erroni.
- Eye: Dibuix que tot enemic té per identificar-se.
- EnemyCollider: Detecta les col·lisions només entre enemics.
- Parry Shield: Escut que s'activa al voltant dels *Bounce* i *Intelligent* en el nivell 1 del perfil *Maestry*.

- Freeze Feedback: Efecte de congelació que apareix quan el jugador principal dispara mitjançant les *freeze Bullets.*

Figura 35: Captura de Unity amb els components complementaris dels enemics Shooter. Font pròpia.

El script encarregat del comportament dels *Shooters* s'anomena *EnemyShooterScript()*. Aquest s'estructura sempre igual (independentment de la classe d'enemic que sigui). Conté una variable *float* que determina la classe de comportament que ha de seguir; 1 = *Shooter Basic,* 2 = *Shooter Intelligent* i 3 = *Shooter Bounce*. Mitjançant aquest, un controlador amb estructura de tipus *switch()* decideix quins valors ha de donar a les diferents variables i el seu comportament durant el joc. L'assignació de valors a les variables es produeix a la funció *Start()* mitjançant el *StartMetod()*:

```
void StartMetod()
{
    switch(enemyType)
    {
        case 1: life = BlackBoardEnemy.GetComponent<BLACKBOARD_ENEMYS>().sh_LifeBasic;
            timeToShoot = Random.Range(0.5f, 0.7f);
            ready = false;
            break;
        case 2:
            life = BlackBoardEnemy.GetComponent<BLACKBOARD_ENEMYS>().sh_LifeBounce;
            timeToShoot = BlackBoardEnemy.GetComponent<BLACKBOARD_ENEMYS>().sh_TimeToShootBounce;
            ready = true;
            timer = BlackBoardEnemy.GetComponent<BLACKBOARD_ENEMYS>().sh_TimeToShootBounce - 0.2f;
            break;
        case 3:
            life = BlackBoardEnemy.GetComponent<BLACKBOARD_ENEMYS>().sh_LifeIntelligent;
            timeToShoot = BlackBoardEnemy.GetComponent<BLACKBOARD_ENEMYS>().sh_TimeToShootIntelligent;
            ready = true;
            timer= BlackBoardEnemy.GetComponent<BLACKBOARD_ENEMYS>().sh_TimeToShootIntelligent- 0.2f;
            break;
    }
```

Figura 36: Captura del codi EnemyShooterScript() amb la funció StartMetod(). Font pròpia.

Per altra banda el comportament es calcula a la funció *ShooterController()* que segueix el mateix model que el controlador de la figura 36. Aquest es troba al *Update()*.

Alhora de disparar, cada enemic té les seves característiques, i segons el comportament, funciona diferent:

- Basic Shoot: Mitjançant el *WallCheckers* determina i decideix un eix cap al que disparar. A partir d'aquí crea bales només en aquesta direcció en un període de temps aleatori que s'assigna en el *StartMetod()*.
- Intelligent Shoot(): L'enemic crea una bala especial de tipus intel·ligent. Aquesta calcula en la seva funció *Start()* la direcció que prendrà en un Vector2. Per fer-ho, resta la posició del jugador menys la seva i després normalitza aquest valor per multiplicar-lo per la velocitat.

```
moveDirection = (target.transform.position - this.transform.position).normalized * speed;
```

Figura 36: Captura del codi EnemyShooterScript() del Vector2 que guarda la direcció de les bales . Font pròpia.

El vector es passa després al component *RigidBody2D* per poder moure l'objecte.

```
void IntelligentShoot()
{
    rb2d.velocity = new Vector2(moveDirection.x, moveDirection.y);
}
```

Figura 38: Captura del codi EnemyShooterScript() amb la funció IntelligentShoot(). Font pròpia.

- Bounce Shoot: Aquest funciona de manera semblant al *Intelligent*. L'antagonista crea una bala de tipus *bounce* que calcula la posició del jugador al ser creada. Al xocar contra un mur, es torna a calcular la direcció cap al jugador. Amb això s'aconsegueix l'efecte de rebot i la impressió que la bala va darrere el jugador.

Per últim, existeixen altres funcions que s'han de mencionar, ja que la majoria d'enemics (siguin del grup que siguin) consten d'elles. Aquestes són:

- *Frezze()*: impedeix que l'enemic segueixi disparant durant un període de temps si ha estat congelat.
- *SpawnObj()*: crea un cor o moneda al morir en un de cada deu casos.
- *OnTriggerEnter2D()*: encarregat de detectar i gestionar les diferents col·lisions.

Followers

Tal com s'ha presentat en els *Shootters*, consten d'elements lligats a ells que complementen i ajuden amb certs comportaments. Aquests, que també es troben en la classe anterior i són els de la imatge:

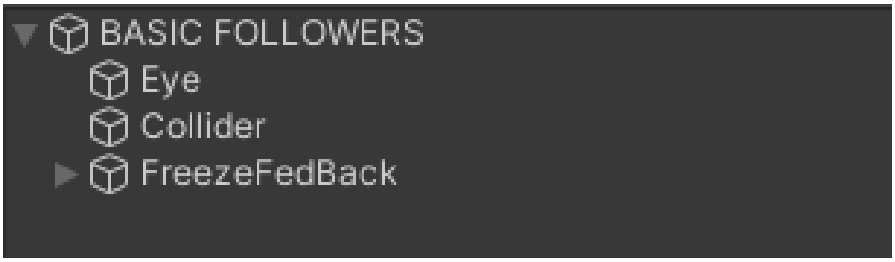


Figura 39: Captura de Unity dels elements que complementen al enemic Follower. Font pròpia.

El script encarregat del comportament d'aquest grup s'anomena *EnemyFollowerScript*() i s'organitza igual que l'anterior classe; consta d'una variable *float* que determina el valor de les seves variables (al *StartMetod()*) i el comportament que segueix (al *FollowerController*). Cada enemic es mou de la mateixa manera; només varia la velocitat, la vida, o l'acció de morir. Per fer-ho, es calcula la direcció que ha de seguir tal com es fa en les bales dels *shooters*. L'única diferència és que al calcular-se la direcció en temps real, és a dir, en la funció *Update()*, l'enemic segueix el jugador en comptes de fixar una única direcció.

```
void Follow(GameObject x)
{
    moveDirection = (x.transform.position - this.transform.position).normalized * speed;
    rb2d.velocity = new Vector2(moveDirection.x, moveDirection.y);
}
```

Figura 40: Captura del codi EnemyFollowerScript() amb la funció Follow(). Font pròpia.

A l'hora de crear aquest comportament apareix el problema del solapament. Aquest passa quan dos enemics que persegueixen l'usuari xoquen i un es sobreposa a l'altre.

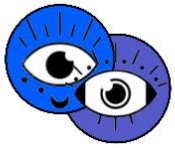


Figura 41: Representació del solapament de dos enemics. Font pròpia.

Aquest efecte s'evita gràcies al component *EnemyCollider* que tots els antagonistes comparteixen. Quan un enemic (de tipus *Follower*) entra en contacte amb aquest, efectua una petita esquiva endarrere i evita, en el major dels casos, l'efecte descrit.

Per últim, i de la mateixa manera que els *Shooters*, aquests també consten de funció *Freeze*, *SpawnObj* i *OnTriggerEnter2D.*

Specials

D'entrada, comparteixen els mateixos components que els *Followers.* El script encarregat del comportament s'anomena *EnemySpecialScript()* i funciona de la mateixa manera que els dos anteriors.

A diferència dels altres, aquest grup només actuen a l'entrar en contacte amb el jugador. Aquesta col·lisió es calcula en el jugador de la mateixa manera que els diferents estats:

- Inverter: Per aconseguir aquest efecte, durant un període de temps, la velocitat del jugador passa a ser negativa, i per tant, els controls s'inverteixen sense necessitat de variar res més.
- Squid: Al xocar, una imatge amb l'opacitat al 60% dóna la sensació de falta de llum dificultant la visió del jugador.
- Mother: Aquest no causa res al jugador. Crea petits fills que es comporten com un enemic *Follower*, i per tant, té tots els seus elements i scripts.

Per últim, comparteixen les funcions *SpawnObj()* i *OnTriggerEnter2D* dels seus companys, però al no poder-se congelar no consten de funció *Freeze()*.

Aquest model d'implementació dels enemics permet al desenvolupador canviar el comportament només modificant la variable *enemyType*. D'aquesta manera s'engloben les mateixes classes de comportaments en scripts més grans i s'evita la duplicació innecessària i redundant de funcions.

Mapa

A l'hora de tractar el següent punt s'ha de tenir clar els diferents elements que formen part de la creació del mapa.

1. Sales: Element principal. Compta amb diferents components que ajuden en la creació del món.
2. El script *RoomTemplates()*: S'encarrega de crear, reiniciar i controlar el món del joc. Aquest es troba en un *GameObject* al marge dels altres comportaments del joc.
3. Complements: Afegits que es col·loquen en les sales un cop el mapa està acabat. Aquests donen sentit al món i al joc: enemics, botiga, escales, obstacles, llums, etc...

Sales

Existeixen 3 classes de sales segons la seva funció en la creació del mapa, i aquestes són:

- Entry Room: Sala principal on el jugador comença i punt d'inici per la creació del mapa. Només existeix una per nivell. Es compon dels següents elements:
    - Spawn Points: Encarregats, mitjançant el script *RoomSpawner(),* de crear una sala per cada porta. Cada un conté una posició relativa; *up, down, right o left* que determina quina classe d'habitació podrà crear. A l'hora de col·locar-ne una primer es comprova que no hi hagi ja una sala creada. En cas negatiu, escull i crea aleatòriament una que connecti amb aquesta. En cas que dos *Spawn Points* es trobin, gràcies a la posició que ocupen, un tindrà preferència respecte a l'altre i per tant, s'evitarà que es creïn dues sales en el mateix lloc.
    - Camera Point: Scritp encarregat dels diferents comportaments de cada sala, tals com: comprovar si el jugador es troba o no en ella, crear el sistema de llums per guiar l'usuari, crear i numerar els enemics, etc...
    - Room Components: Imatges que formen les parets de la sala.

- Invisible Doors: *Colliders* que es col·loquen a les portes i s'encarreguen d'impedir que els enemics o les bales surtin de la sala.

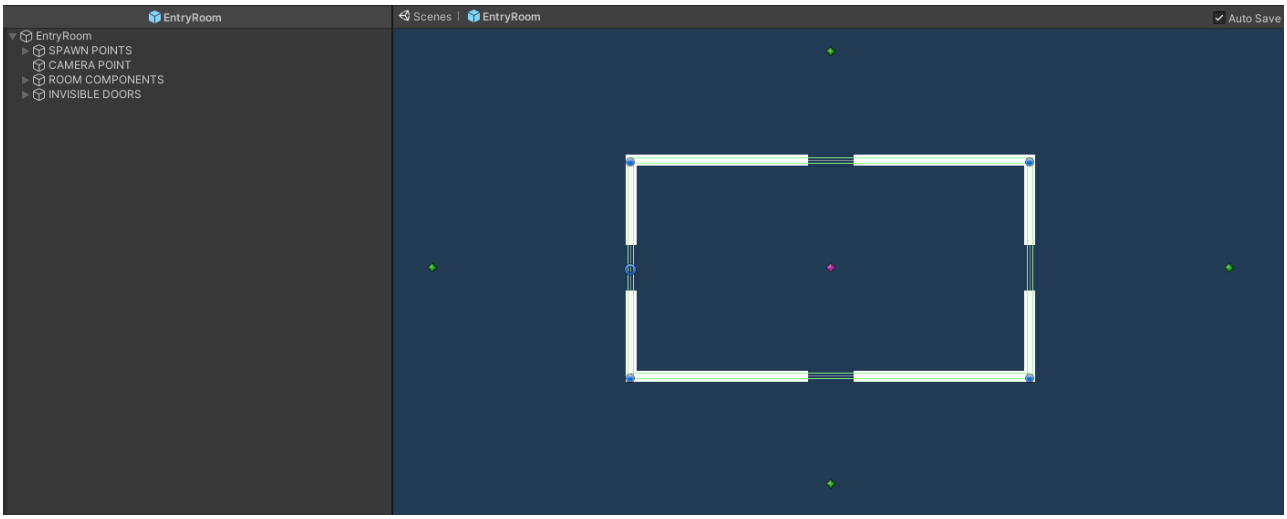


Figura 42: Captura de Unity del component EntryRoom. Font pròpia.

- Normal Room: Sales que conformen el mapa. Existeixen 14 classes diferents segons la distribució de les portes. Pot haver-n'hi tantes com l'algoritme determini. Compta amb els mateixos elements que la *EntryRoom*.
- Close Room: Aquest últim model no comparteix cap dels elements que s'han presentat. La seva funció és impedir el pas al jugador tancant permanentment portes de sales.

Room Templates

Explicat el funcionament de les diferents sales, queda crear un mapa en blanc segons les especificacions marcades en l'apartat de *Disseny*. Per fer-ho, es pot organitzar les diverses funcions en 3 accions diferents:

1. Crear del mapa.
2. Eliminar el mapa
3. Manteniment del mapa

Per la creació, el primer que fa el codi és preguntar-se si existeix una *EntryRoom*, ja que com s'ha aclarit en l'apartat anterior, és el punt d'inici. En cas de no existir cap, ell mateix crearia una en la posició actual del jugador.

Quan aquesta entra en joc, són els *Spawn Points* els encarregats de crear i organitzar les diferents sales. Mentre aquestes van apareixent es va guardant en un llistat el nombre de sales que es troben en el joc. D'aquesta manera es pot crear una funció que comprovi en tot moment si el nombre d'habitacions és major a un mínim, i menor a un màxim. Aquest comportament es troba en una funció de tipus *bool* anomenada *MapIsReady()*.

```
public bool MapIsReady()
{
    if(rooms.Count != 0)
    {
        if(rooms.Count > MaxNumOfRooms)
        {
            RestartMap();
            return false;
        }
        else
        {
            if(rooms.Count <= MaxNumOfRooms && rooms.Count >= MinNumOfRooms)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

    }
    else
    {
        return false;
    }

}
```

Figura 43: Captura del codi RoomTemplates() amb la funció MapIsReady(). Font pròpia.

Quan aquesta funció ens retorna un valor *true,* els *Spawn Points* deixen de crear sales normals i col·loquen *close rooms*, que trenquen el creixement del mapa.

En cas que el valor retornat fos *false* perquè el nombre de sales no segueix els nombres establerts s'ha de reiniciar el món. Per fer-ho, primer s'elimina tots els elements del joc en un ordre establert pel desenvolupador. Un cop no es tingui la *entry room*, la funció que busca si n'existeix una entraria en joc, i es començaria a crear el mapa de nou.

Per poder eliminar tots els elements, és necessari que durant el joc es guardi una referencia d'aquests en el codi. Aquest procés és l'anomenat manteniment del joc. Mitjançant el sistema de *tags*, es busca els diferents components del món i es guarden en uns llistats.

```
void ElementsOnMap()
{
    entryRoom = GameObject.FindGameObjectWithTag("EntryRooms");
    enemysOnMap = GameObject.FindGameObjectsWithTag("Enemy");
    shopOnMap = GameObject.FindGameObjectsWithTag("Shop");
    enemyRoomsOnMap = GameObject.FindGameObjectsWithTag("EnemyRoom");
    passiveHabilitysOnMap = GameObject.FindGameObjectsWithTag("PassiveHability");
    activeHabilitysOnMap = GameObject.FindGameObjectsWithTag("ActiveHability");
    specialHabilitysOnMap = GameObject.FindGameObjectsWithTag("SpecialHability");
```

Figura 44: Captura del codi RoomTemplates() amb la funció ElementsOnMap (). Font pròpia.

Quan s'ha d'eliminar-los, es ressegueix la llista amb un *foreach* i és borra un a un els diferents elements.

```
void DestroyRooms()
{
    if(rooms.Count > 0)
    {
        foreach(GameObject x in rooms)
        {
            rooms.Remove(x);
            Destroy(x);
        }
    }
}
```

Figura 45: Captura del codi RoomTemplates() amb la funció DestroyRooms(). Font pròpia.

Complements

Un cop tenim un mapa que conté les característiques marcades pel desenvolupador, s'ha d'omplir aquest de tots els elements que converteixen el món en un *roguelike* jugable. Tal com s'explica en l'apartat de *Disseny,* existeixen diferents models de sales segons la seva funció. Tenint en compte el nombre de cops que aquestes apareguin, es té dues classes:

- Apareix un cop per nivell: Botiga, escales, sala especial.
- Apareix múltiples cops: sala d'enemics i sales buides.

El primer grup és gestionat pel mateix *RoomTemplates()*, un cop el mapa buit s'ha creat. Diverses funcions que segueixen la mateixa lògica es pregunten si existeix alguna botiga, escala osala especial en el món, i en cas negatiu, crea una segons els paràmetres marcats. Això s'aconsegueix gràcies al llistat de sales que el mateix codi conté, ja que permet determinar en quina habitació es crea cada un dels elements, i assegurar que no coincideixen.

```
public void InstantateStairs()
{
    if(spawnStairs == false && MapIsReady())
    {
        Instantiate(stairs, rooms[rooms.Count-1].transform.position, Quaternion.identity);
        MapIsFinished = true;
        spawnStairs = true;
    }
}
```

Figura 46: Captura del codi RoomTemplates() amb la funió InstantateStairs(). Font pròpia.

En el cas de l'escala, aquesta apareix en l'última sala que s'ha creat. Per altra banda la sala especial pot aparèixer en un rang que compren des de la cinquena habitació fins a la meitat total , mentre la botiga des de la meitat més u (perquè no coincideixin) fins a la cinquena sala començant pel final. Un cop aquests elements han estat col·locats en el mapa, queda assignar les sales buides i les d'enemics. Aquest procés és gestionat pel component *CameraPoint()*. Seguint una probabilitat marcada en la *Blackboard* dels enemics cada una determina si té, o no, enemics en ella. En cas positiu, i de manera aleatòria, s'assigna quin tipus de sala serà (*shooter, follower o comuna*).

```
void TypeOfEnemyRoom()
{
    switch(enemyRoomTypeRndVar)
    {
        case 1:
            if(!oneTime)
            {
                Instantiate(enemyBrain.GetComponent<BLACKBOARD_ENEMYS>().shooterRoom, this.transform.position, Quaternion.identity);
                oneTime = true;
            }
        break;
        case 2:
            if(!oneTime)
            {
                Instantiate(enemyBrain.GetComponent<BLACKBOARD_ENEMYS>().followerRoom, this.transform.position, Quaternion.identity);
                oneTime = true;
            }
        break;
         case 3:
            if(!oneTime)
            {
                Instantiate(enemyBrain.GetComponent<BLACKBOARD_ENEMYS>().superRoom, this.transform.position, Quaternion.identity);
                oneTime = true;
            }
        break;
    }
}
```

Figura 47: Captura del codi CameraPoint() amb la funció TypeOfEnemyRoom(). Font pròpia.

D'aquesta manera s'aconsegueix que el procés de col·locar els enemics sigui totalment aleatori i fàcil de manipular i variar gràcies a les diferents probabilitats que es troben en la *Blackboard* dels enemics.

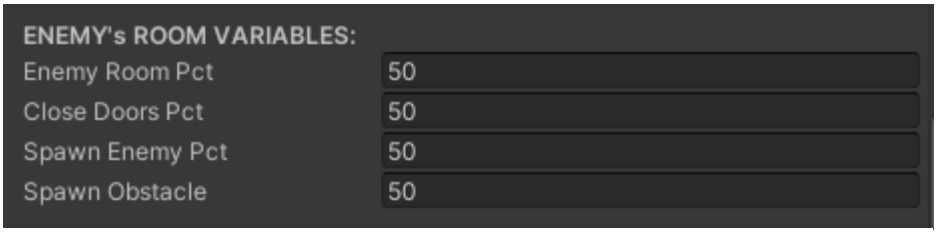


Figura 48: Captura de Unity de diferents valors continguts en la ENEMY_BLACBKOARD(). Font pròpia.

Tornant al *CameraPoint(),* si s'obté que no serà una sala d'enemics, automàticament defineix aquesta com una sala buida. A partir d'aquí, i de la mateixa manera que amb els antagonistes, es tria aleatòriament quin serà l'obstacle de la sala i es col·loca en ella

En cas que es vulgui crear quelcom en una sala on prèviament existeixi un altre element amb preferència (botiga, sala especial o escales), no s'efectua l'acció i es queda amb l'element que contingui.

Amb tot això, ja es té un mapa amb tots els elements distribuïts, només queda presentar les diferents sales amb el seu funcionament intern.

Classes de sales

Sala d'enemics

Un cop s'assigna que una habitació té antagonistes, es crea l'element *EnemyRoom(),* encarregat de triar quina classe de sala serà. També mira que la zona no estigui ocupada per un altre element. Cada *EnemyPoint()*, l'element encarregat de crear l'enemic, té assignat una referència de tots ells, i, segons la classe de punt que sigui, accedeix a uns o altres. Aquest element és també l'encarregat de col·locar els obstacles en cas d'haver-n'hi. Segons com s'hagi creat l'habitació, aquestes serien 3 possibilitats de com es poden veure en el joc:

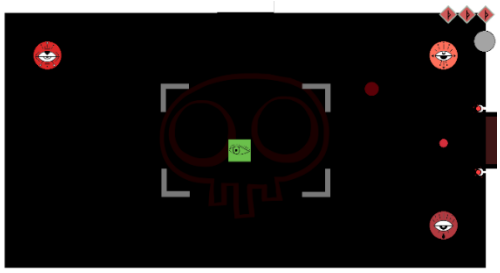


Figura 49: Imatge del videojoc, sala de Shooters. Font pròpia.

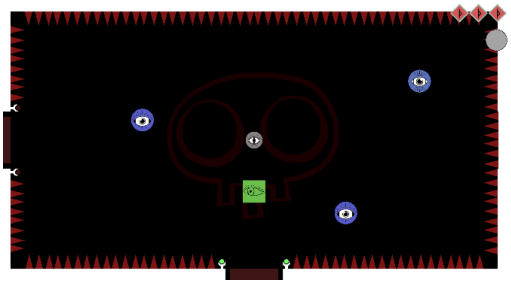


Figura 50: Imatge del videojoc, sala de Followers. Font pròpia.

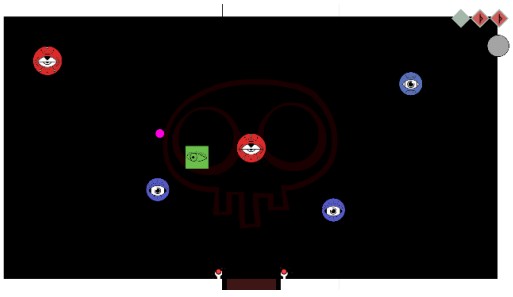


Figura 51: Imatge del videojoc, sala comuna. Font pròpia.

Sala buida

De la mateixa manera que en l'anterior, un element anomenat *NormalRoomManager()* decideix quina classe de sala buida és partint de diferents dissenys que prèviament s'han creat. Existeixen 6 de diferents:

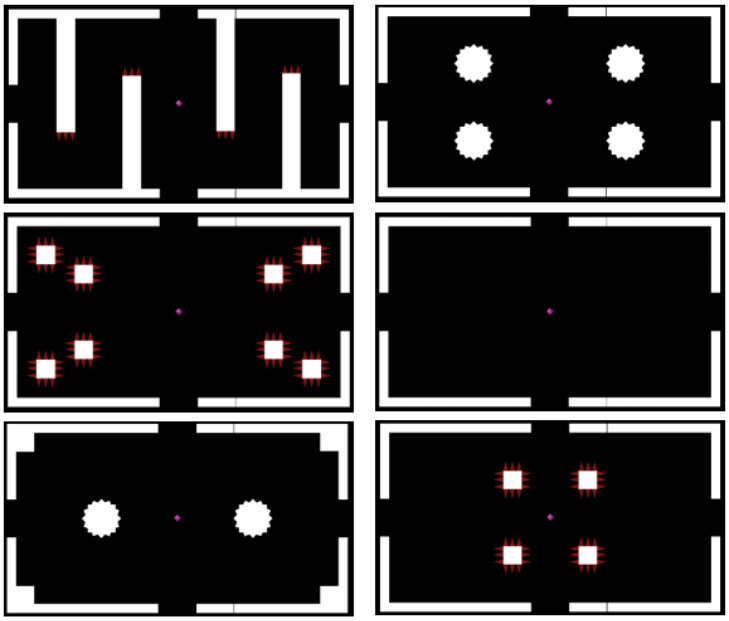


Figura 52: Imatge del videojoc, diferents formes de la sala buida. Font pròpia.

Aquests es troben referenciats en l'element, i mitjançant una variable *float* i una estructura de tipus *switch()* s'assigna de les diferents decoracions.

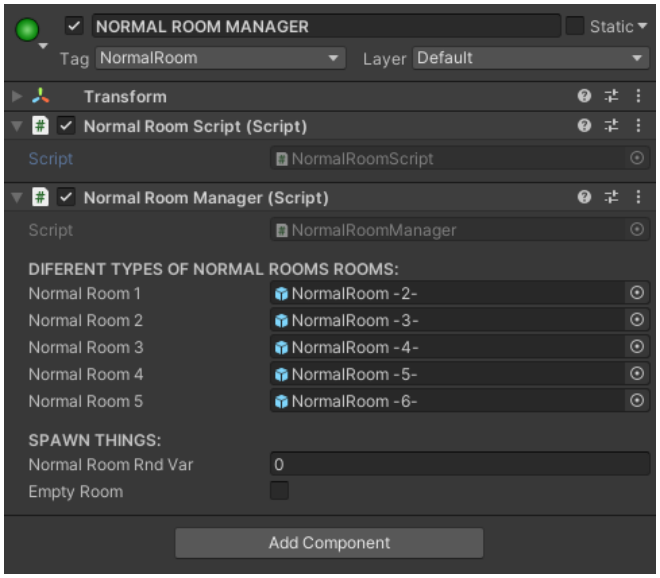


Figura 53: Captura de Unity del component NormalRoomManager(). Font pròpia.

Botiga

Consta de 5 *SellPoints()* que creen un element aleatòriament i segons el que s'ha decidit. Per altra banda, compta amb un *Collider* que impedeix accedir als components si no es té els diners suficients. Perquè el jugador pugui saber a què pot accedir, un petit *HUD* determina el preu de cada producte. Segons el color que d'aquest, l'usuari sap que pot comprar. Els colors són el blanc en cas positiu, i el vermell en cas negatiu.

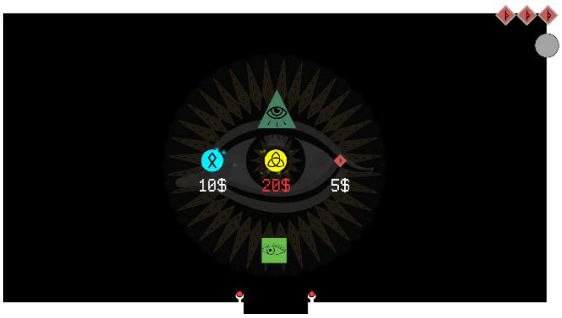


Figura 54: Imatge del videojoc, botiga.
Font pròpia.

Depenent del perfil que l'algoritme decideixi per l'usuari, es desbloquegen el quart i cinquè punt. Per últim, en cas que el jugador compri un element, aquest desapareix i no es torna a crear cap altre fins al pròxim nivell.

Sala especial

Està formada per un *SpecialRoomPoint()* que s'organitza i funciona com els anteriors. També existeixen 2 mags; un sempre actiu i l'altre que es desbloqueja en el primer nivell del perfil *Cretivity*. El *point,* crea de manera aleatòria una habilitat activa o passiva, i en cas de fer negocis amb el mag, substitueix aquesta per una habilitat especial o un element per variar l'aspecte. Un petit *HUD* informa al jugador sobre els serveis dels mags. Aquest text només actiu quan el jugador es troba a la sala i desapareix al

donar-li els dos cristalls al mag. L'habilitat passiva o activa també desapareix a l'aconseguir-se i no es crea cap altre fins al pròxim nivell.

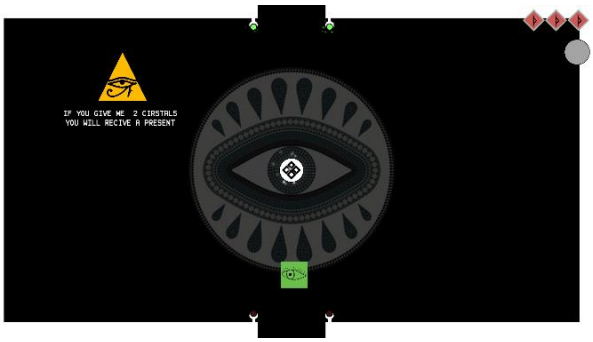


Figura 55: Imatge del videojoc, sala especial. Font pròpia.

Escales

L'habitació està formada per elements decoratius i l'escala. Aquesta conté un script anomenat *StairsScript()* que s'encarrega de passar de nivell un cop el jugador entra en contacte amb ella. Per fer-ho, accedeix al codi *RoomTemplates()* per reiniciar el mapa. Aquesta funció consta de dues parts: eliminar i crear mapa.

```
public void RestartMap()
{
    if(!allDeleted )
    {
        RestartMissions();
        if(!onlyOneTime)
        {
            DificultyUprage();
            PorfileUprage();
            onlyOneTime = true;
        }
        DeleteMap();
    }
    else if(allDeleted && timerForRestart >= timeToRestart)
    {
        CreateMap();
    }
}
```

Figura 56: captura del codi RoomTemplates() amb la funció RestartMap(). Font pròpia.

Al eliminar tots els elements també s'ha de tenir en compte que cal reiniciar o modificar diferents valors de cara al pròxim nivell. Per una banda trobem el *RestartMissions(),*

que reinicia totes les missions per poder-se tornar a jugar. També es té el *DificultyUprage()*, que augmenta el nombre de sales màximes, vida dels enemics, la rapidesa amb la qual es mouen, etc... Amb aquests canvis s'augmenta la dificultat al llarg dels nivells. La funció *PorfileUprage()* es presenta amb detall més endavant. Per últim tenim les funcions *DeleteMap()* i *CreateaMap()* que ja s'han explicat en l'apartat anterior.

Amb tot això aquest és l'aspecte que presenta la sala de les escales:

Figura 57: Imatge del videojoc, sala final.
Font pròpia.

Algoritme d'elecció del perfil

El codi encarregat de tots els càlculs que pertoquen l'algoritme d'elecció del perfil es troben en un codi anomenat *MasterBrainScript()*. Aquest està ubicat en un *GameObject* independent sota el nom de *MasterBrain*. Per enfocar aquest script, cal dividir-lo en les 3 principals feines que porta a terme:

1. Càlcul dels indicadors
2. Càlcul del perfil
3. Desbloqueig de les mecàniques especials segons el perfil

Cada indicador (en total 10) consta de la mateixa estructura i retorna la mateixa classe de variables. Si s'agafa com exemple el nombre de sales visitades s'obté la següent funció:

```
void Indicator1()
{
    roomsSeen = player.GetComponent<ProtoBLACKBOARD_Player>().numOfRoomsSeenInTheLevel;
    totalRooms = roomManager.GetComponent<RoomTemplates>().sizeOfList + 1;

    roomPctMon = (roomsSeen/totalRooms) * 100;

    roomPctGameplay = 100 - roomPctMon;
}
```

Figura 58: Captura del codi MasterBrainScript() amb la funció Indicator1(). Font pròpia.

Com s'observa en la figura 58, cada indicador consta de dues variables de tipus *float* que referencien valors que poden estar en 3 codis en particular: *ProtoBlackBoard_Player(), BLACKBOARD_ENEMY()* o *RoomTemplates()*. A partir d'aquests, es calcula el percentatge de l'eix positiu (en aquest cas Món). Per aconseguir l'eix negatiu es resta 100 al valor prèviament aconseguit.

Cada eix consta de 5 valors diferents, que mitjançant la *(1.1)* i la *(1.2)*, ens retorna el percentatge total de cada perfil.

```
void GlobalPctMon()
{
    pctMon = (roomPctMon + cristalPctMon + moneyPctMon + wastedMoneyPctMon + specialHabilityPctMon) / 5;
}
```

Figura 59: Captura del codi MasterBrainScript() amb la funció GlobalPctMon(). Font pròpia.

Un cop es té els 4 resultats globals, una funció anomenada *PctProfiles()* converteix aquets en els percentatges de cada taxonomia d'usuari.

```
void PctProfiles()
{
    pctProfileAction = (pctAction+ pctGameplay) / 2;
    pctProfileMaestry = (pctGameplay+ pctInteraction) / 2;
    pctProfileAchievement = (pctAction+ pctMon) / 2;
    pctProfileCreativity = (pctMon+ pctInteraction) / 2;

}
```

Figura 60: Captura del codi MasterBrainScript() amb la funció Indicator1PctProfiles(). Font pròpia.

Aquests valors són únicament informatius, ja que no tenen cap pes en la fórmula final. L'usuari pot comprovar-los clicant la 'I' per obrir el menú d'informació. Aquest, que recopila també els diferents percentatges dels indicadors, es veu de la següent manera:

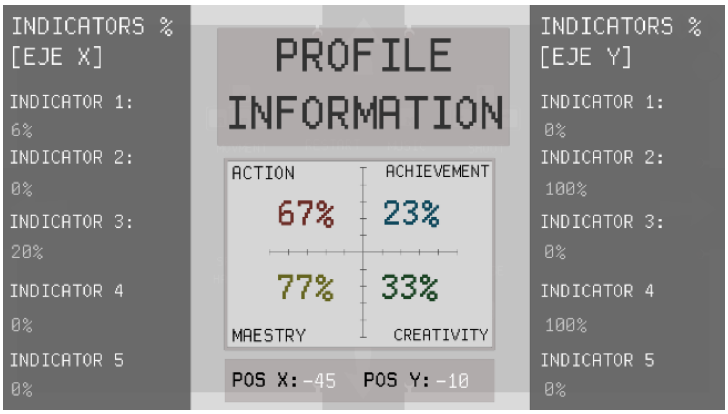


Figura 61: Captura del codi MasterBrainScript() amb la funcións Xpos() i Ypos(). Font pròpia.

Per aconseguir la posició és necessari tornar a utilitzar els percentatges globals de cada eix. Amb aquests i mitjançant *(1.3)*, sobté com a resultat les següents funcions:

```
void Xpos()
{
    xPos = (pctMon/2) + (-pctGameplay/2);
}
1 reference
void Ypos()
{
    yPos = (pctAction/2) + (-pctInteraction/2);
}
```

Figura 62: Captura del videojoc, menú d'informació. Font pròpia.

Aquestes posicions es calculen a temps real durant tot el transcurs del joc. Encara així, només es tenen en compte cada dos nivells, ja que es considera que en aquest temps el jugador haurà pogut realitzar totes les accions que poden fer variar els indicadors. La funció *TaxonomyChange()* es l'encarregada de guardar la posició obtinguda al finalitzar el nivell marcat i varia les mecàniques del joc activant o desactivant els diferents nivells dels perfils.

Cada regió calcula independentment de les altres quin nivell ha de desbloquejar o amagar segons la posició donada. Primer es considera si el jugador està dintre del perfil marcat. Si s'agafa com a exemple *Achievement,* tant la X com la Y haurien de ser positives, i per tant estar entre 0 i 50.

```
if(xPos >= 0 && xPos <= 50 && yPos >= 0 && yPos <= 50)
{
    Debug.Log("ACHIEVMENT LV2");
    achievementProfile = true;
    player.GetComponent<ProtoBLACKBOARD_Player>().activeShopColorSkins = true;
    player.GetComponent<ProtoBLACKBOARD_Player>().activeShopDrawSkins = true;
    player.GetComponent<ProtoBLACKBOARD_Player>().activeHardMissions = true;
    player.GetComponent<ProtoBLACKBOARD_Player>().activeLargeMissions = true;

}
```

Figura 63: Captura del codi MasterBrainScript() amb part de la funció AchievementProfile(). Font pròpia.

Tenint en compte aquest valor s'activaria els dos nivells del perfil, però en cas que la posició no es trobés en aquest quadrant, s'ha de comprovar en quin cas es troba la posició. Mitjançant una estructura de *if else* s'aconsegueix filtrar els casos segons els valors recollits.

```
else
{
    if(xPos < -25 || yPos < -25)
    {
        Debug.Log("ACHIEVMENT -FORA DELS LIMITS-");
        achievementProfile = false;
        player.GetComponent<ProtoBLACKBOARD_Player>().activeShopColorSkins = false;
        player.GetComponent<ProtoBLACKBOARD_Player>().activeShopDrawSkins = false;
        player.GetComponent<ProtoBLACKBOARD_Player>().activeHardMissions = false;
        player.GetComponent<ProtoBLACKBOARD_Player>().activeLargeMissions = false;
    }
    else
    {
        if(xPos > 5 && xPos <= 25 || yPos > 5 && yPos <= 25) //xPos entre 5 i 25 i yPos entre 5 i 25
        {
            Debug.Log("ACHIEVMENT LV1");
            player.GetComponent<ProtoBLACKBOARD_Player>().activeShopColorSkins = true;
            player.GetComponent<ProtoBLACKBOARD_Player>().activeShopDrawSkins = true;
            player.GetComponent<ProtoBLACKBOARD_Player>().activeHardMissions = true;
            player.GetComponent<ProtoBLACKBOARD_Player>().activeLargeMissions = false;
        }
        else
        {
            Debug.Log("ACHIEVEMENT -no esta entre 5 i 25 o 5 i 25-");
            achievementProfile = false;
            player.GetComponent<ProtoBLACKBOARD_Player>().activeShopColorSkins = false;
            player.GetComponent<ProtoBLACKBOARD_Player>().activeShopDrawSkins = false;
            player.GetComponent<ProtoBLACKBOARD_Player>().activeHardMissions = false;
            player.GetComponent<ProtoBLACKBOARD_Player>().activeLargeMissions = false;
        }
    }
}
```

Figura 64: Captura del codi MasterBrainScript() amb part de la funció AchievementProfile(). Font pròpia.

Un cop el jugador arriba al nivell 10 (final del joc), mitjançant els 4 valors que s'han guardat cada dos nivells (nivell 2, 4, 6 i 8), es fa una mitjana que determina quin ha estat la posició final del jugador, i per tant, el seu perfil resultant durant el transcurs del joc.

```
void FinalPos()
{
    finalXpos = (posXLvl2 + posXLvl4 + posXLvl6 + posXLvl8) / 4;
    finalYpos = (posYLvl2+ posYLvl4 + posYLvl8 + posYLvl6) / 4;
}
```

Figura 66: Captura del codi MasterBrainScript() amb la funció FinalPos(). Font pròpia.

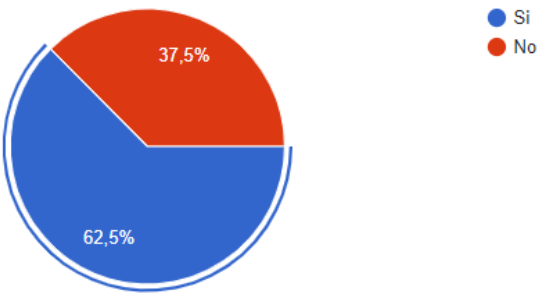
Aquesta és reflexa en el menú final, que de manera dinàmica, mostra al jugador el seu perfil d'usuari i una petita definició sobre aquest.

RESULTATS

Un cop el joc és funcional, es poleix diferents aspectes afegint sistemes de partícules, sons, música de fons, menú principal, menú introductori i d'altres elements. Amb això, es vol aconseguir una experiència de joc més amena i divertida.

S'efectua la *build* 1.0 de *The EYE,* que es proporciona a un grup reduït de població. Aquest, durant el període d'una setmana, tracta de superar-lo.

Finalitzada la setmana de prova es passa una enquesta a aquelles persones que han provat el joc amb el fi d'aconseguir informació sobre algunes qüestions del títol. L'enquesta consta de 6 preguntes que han aconseguit els següents resultats:

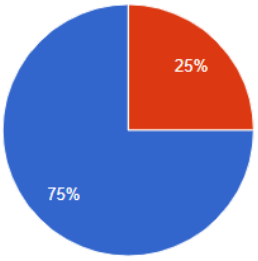


Gràfic 2: Percentatge de persones que han superat el joc més la seva respectiva llegenda. Font: formularis de google.

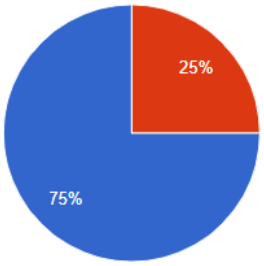
Gràfic 3: Percentatge de persones que han notat una millora en el joc. Font: formularis de google.

En primer lloc s'obté el nombre de persones que han superat el joc que representen un 62.5%. Aquest fet pot indicar que el 37.5% s'han sentit frustrats per un mal balanç de la dificultat que els ha portat a renunciar. En contraposició, s'obté que un 100% dels usuaris han millorat en el seu estil de joc en cada *run* jugada. Per tant, es pot afirmar que la corba de dificultat que presenta *The EYE* funciona.

Seguint amb l'enquesta, es recopila el nombre de jugadors que han trobat errors. Aquests es troben tant en el disseny, amb un 25% dels usuaris, a causa de mecàniques o enemics mal plantejats. Per altra banda quan parlem d'erros de programació es sol utilitzar el terme *bug.* Aquests causen situacions inesperades que dificulten, faciliten o impedeixen jugar amb normalitat. Aquests han estat els més presents amb un 75% de persones que n'han trobat algun. Donades aquestes dades es parla amb les persones que han jugat amb l'objectiu d'identificar més detalladament aquests errors.

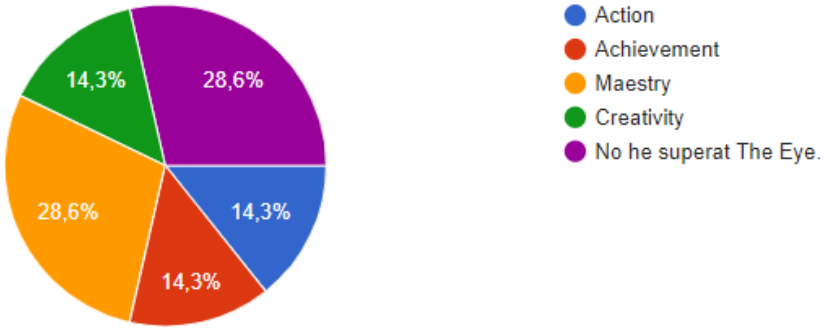


Gràfic 6: Percentatge de persones que han trobat errors de cosi en el joc. Font: formularis de google.



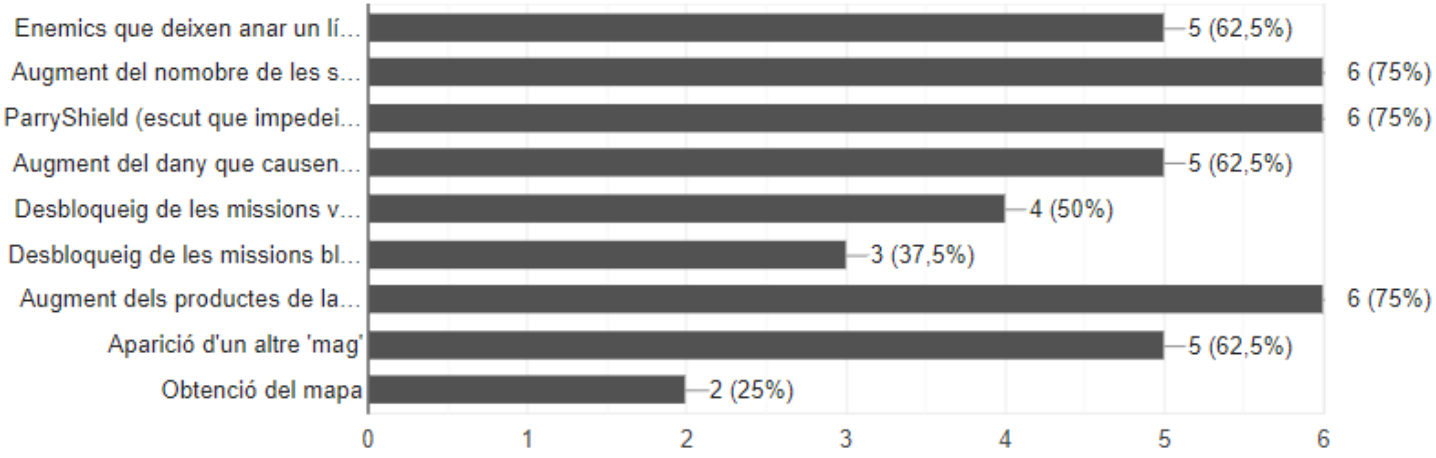
Gràfic 5: Percentatge de persones han entès tots els sistemes del joc. Font: formularis de google.

Per últim s'ha preguntat quines mecàniques han aparegut durant el transcurs del joc i, en cas d'haver superat *The EYE,* quin havia estat el perfil resultant. El perfil que més apareix és *Maestry* seguit dels altres 3 empatats.



Gràfic 4: Percentatge dels diferents perfils que han obtingut els usuaris en el joc. Font: formularis de google.

Aquesta dada es confirmada amb les obtingudes en la gràfica de les mecàniques . Tant el *Parry Shield* com l'augment del dany dels enemics són dues de les més vistes.



Gràfic 7: Percentatge de mecàniques que han desbloquejat en el joc. Font: formularis de google.

Per altra banda s'observa que un dels perfils més difícils d'aconseguir és *Creativity,* ja que molt poca gent ha desbloquejat el mapa. Aquest fet podria també podria venir a causa d'un mal disseny en el moment de presentar-lo causant que moltes persones no reparin en la seva existència.

Seguint amb les menys vistes, es troben les missions blaves, és a dir, el nivell 2 del perfil *Achievement*. Ambdós perfils es troben en la zona de l'eix que tendeix cap a X positiva. Es pot dir per tant, que la majoria de jugadors han prioritzat el *gameplay* al món. Donada la falta d'art, es una dada lògica.

Un cop processades les diferents opinions sobre *The EYE*, es pot organitzar els punts positius i negatius en dos blocs:

- Disseny.
- Codi (només errades).

Disseny

Problemes

El primer problema que s'identifica és la presentació del *parry*. La majoria d'usuaris són jugadors casuals, cosa que porta a una confusió respecte com utilitzar aquesta mecànica. A falta d'un tutorial o explicació, els jugadors no entenen ni la seva utilitat en el joc ni la relació amb les bales especials que disparen els enemics. Aquest fet porta a no utilitzar-lo i per tant, impedeix resoldre certes missions.

Per solucionar aquest error es pot presentar un petit tutorial opcional. Aquest apareixeria com una nova opció en el menú inicial. Permetria als nous jugadors entendre i provar les mecàniques abans d'intentar superar el videojoc.

Per altra banda es troba un problema amb les missions. Donat que aquestes s'activen automàticament, molts jugadors les porten a terme sense saber-ho. Aquest fet provoca el desconcert d'alguns usuaris a la vegada que els col·loca en la taxonomia *Achievement* sense entendre molt bé que estan fent.

Per arreglar-ho, es pot implementar un sistema que permeti al jugador acceptar les missions i per tant, ser conscient de l'existència d'aquestes. Es pot portar a terme el sistema en el menú de pausa per facilitar a tots els usuaris accedir-hi.

Encerts

Tant les mecàniques de combat com els diferents enemics, resulten entretinguts i fàcils d'entendre. Encara que no presentin cap innovació en el gènere, si compleixen la seva funció de presentar un repte al jugador que en cap moment es sent ni massa fàcil, ni injust.

Els jugadors, per altra banda, presenten una millora considerable després de jugar una bona estona, cosa que porta a superar amb més facilitat els diferents nivells. Aquest fet fa frustrar poc l'usuari i el motiva a seguir intentant-ho. Tals característiques són un punt positiu en un videojoc *roguelike* que pot pecar d'impossible o frustrant si no es troba ben balancejat.

Codi

Problemes

Donada la gravetat d'alguns, i amb el fi d'aconseguir una millor experiència de joc, molts dels problemes que es presenten a continuació s'han arreglat de cara a la versió final de *The EYE.*

El primer que tothom assenyala i el més important es relaciona amb la creació del mapa. En algunes ocasions, les sales contigües a la *EntryRoom* apareixen, creen tot allò necessari per al mapa i desapareixen. Al no tenir component *CameraPoint()* el comportament dels enemics i el de la càmera falla. No s'ha pogut identificar la raó del problema, i per tant, no s'ha pogut solucionar. Aquest error apareix molt de tant en tant i pot arribar a arruïnar l'experiència de joc.

Per altra banda es troba que els enemics poden fer-te fóra d'una sala tancada, i que aquesta no torna a obrir les portes. En cas que formés part del camí principal, impediria al jugador superar el nivell. Per solucionar l'error, en el codi *EnemyRoomManager(),* s'afegeix la funció *OnTriggerExit2D()* que detecta si quelcom no està en contacte amb l'objecte. D'aquesta manera, en cas que el jugador es trobi fora de la sala, les portes tornen a obrir-se.

```
void OnTriggerExit2D(Collider2D other)
{
    if(other.gameObject.tag == "Player")
    {
        playerIsHere = false;
    }
```

Figura 65: Captura del codi CameraPoint () amb la funcó OnTriggerExit2D(). Font pròpia.

També es troba que els enemics poden expulsar-te de la sala i acabar fora del mapa, impedint per tant, superar el nivell. Per arreglar-ho, s'implementa un comptador de temps que va augmentant només si el jugador no es troba en contacte amb l'objecte *CameraPoint()*. Si aquest temps supera els 3 segons, automàticament transporta l'usuari a la sala principal d'aquell nivell.

```
void ReturnToYourInitalPos()
{
    if(timerNoContactWithCameraPoint < 3)
    {
        if(!contactWithCameraPoint)
        {
            timerNoContactWithCameraPoint += 1*Time.deltaTime;
        }
        else
        {
            timerNoContactWithCameraPoint = 0;
        }
    }
    else
    {
        this.transform.position = rBrain.GetComponent<RoomTemplates>().entryRoom.transform.position;
        timerNoContactWithCameraPoint = 0;
    }
}
```

Figura 67: Captura del codi CameraPoint() amb la funcó ReturnToYourInitialPos(). Font pròpia.

En el menú final, s'identifica que el botó que desbloqueja el perfil del jugador dóna errors i impedeix mostrar-lo. Per solucionar-ho, es canvia el botó 'K' per un comptador de temps. Aquest, al cap de 4 segons, activa automàticament el perfil final i elimina el problema.

Per últim, el sistema de llums que guia al jugador es confon a vegades al mostrar el color de la sala. Aquest error es dóna quan una sala és ocupada per una *Enemy-Room()* i a la vegada per un altre amb preferència (botiga, escales o sala especial). Encara que una elimina l'altre, la llum decideix una de les dues i no torna a variar. Per evitar-ho, es col·loca un delay de 1.5 segons al crear les llums per assegurar-se que al fer el càlcul del color, la sala estigui ocupada per la seu únic complement.

CONCLUSIONS

En primer lloc, dissenyar un videojoc de gènere *roguelike* és un repte. Són molts els casos que la dificultat i la *permadeath* poden arruïnar una experiència de joc polida i ben treballada. Encara la senzillesa del projecte presentat, són molts els detalls col·locats per ajudar l'usuari a tenir una experiència divertida. En tot moment es busca evitar la frustració del jugador mitjançant diferents elements: un *fast failure loop* ràpid, un increment gradual de la dificultat, un *short time horizon*, etc... Amb tot això, s'ha tractat de presentar la mort com una oportunitat d'aprenentatge, ja que en el pròxim intent, l'usuari es troba més preparat i pot arribar més lluny.

Sobre l'art, és el punt més negatiu del projecte. Donat que aquest treball es centra en tant en l'apartat de disseny com en la implementació d'un sistema de narrativa procedimental, l'aspecte visual no ha estat una preferència. Encara així, el videojoc ha de ser clar i jugable, i per tant, s'ha d'entendre tot allò que passa per altres vies. Per aquesta raó, ha estat molt important *feedback* que rep el jugador. D'entrada es tracta de deixar clar tot allò que passa mitjançant els menús i el *Hud*. El menú de *Pausa* mostra les missions i les estadístiques de manera breu i concisa, per no saturar l'usuari amb un *Hud* amb massa informació. També apareixen diferents panells informatius cada cop que el jugador interactua amb elements importants del joc (habilitats actives, especials, passives, cristalls...) amb l'objectiu d'aclarir que és allò que acaben de trobar. Les missions també compten amb els panells cada cop que es compleix o es falla una d'elles.

En segon lloc, existeixen altres elements que ajuden al jugador a saber més sobre el videojoc. El sistema de colors que s'ha utilitzat tractar d'aclarir, a falta d'art, les diferencies entre elements semblants. Tant els enemics, les habilitats o les llums de les sales, presenten un patró de colors que tracta de ser intuïtiu per l'usuari. D'aquesta manera, un cop s'identifica la raó de cadascun, el jugador pot entendre en major mesura tot allò que conté el mon.

Per altra banda les partícules són també un element important. No només tenen una funció estètica quan mor un enemic sinó que poden adoptar diferents funcions. En aquest cas serveixen tant per recalcar objectes importants com per complementar l'acció de *parry* entre d'altres.

Per últim l'apartat sonor acompanya tot el prèviament explicat. Aquest s'utilitza en totes les accions del joc i ajuda a alleujar l'experiència a la vegada que informa sobre les diferents situacions.

Un altre punt important del projecte són els elements procedimentals i aleatoris. La creació del mapa s'ha polit perquè seguís les especificacions marcades per aconseguir-ho, són molts els elements que s'han hagut de desenvolupar. Encara així, gràcies al sistema que lidera el script *RoomTemplates(),* s'ha creat un mon amb diferents característiques que no es veu monòton. Encara estar molt lluny de gegants com *The binding of Issac* o *Hades*, sí que compleix la funció esperada a la vegada que presenta molta facilitat alhora de millorar-lo afegint nous elements.

Ja per acabar, el sistema de narrativa procedimental basat en les taxonomies dels usuaris es compon del desafiament més costos de tot el projecte.

En primer lloc, l'algoritme només contempla models de jugadors organitzats en quatre regions. Aquest fet impedeix extrapolar a autors com *Andrzej Marczewski* o *Chris Bateman*. Encara així, qualsevol segmentació organitzada en un eix de coordenades pot adaptar-se al codi presentat. Només és necessari redissenyar els indicadors que dependran dels nous eixos i la classe de joc.

També permet que el sistema pugui ser més específic del que s'ha mostrat. En aquest treball només s'han contemplat 5 indicadors per cada eix, però en cas de voler treballar amb un nombre major, l'algoritme es tornaria més i més precís. D'aquesta manera es pot aconseguir amb més exactitud un joc que mostri el perfil de jugador. Donat que aquest treball també s'enfocava en els diferents canvis en el joc a causa del jugador, s'ha considerat que 5 indicadors era el nombre necessari per aconseguir tal objectiu.

Per últim, s'ha comprovat que els jugadors poden variar de perfil segons el joc, o l'estat en què es trobin dins d'aquest. En molts casos, alguns usuaris podien canviar radicalment de taxonomia en ple *gameplay* després de perdre vida o aconseguir una habilitat especial. Per tant, es pot finalitzar aquest treball dient que molts dels jugadors no tenen una taxonomia marcada i que és el videojoc al final, el que els pot portar a variar el seu estil de joc.

Annexes

Annex 1

Build del videojoc del projecte (versió 1.0)

A continuació, es facilita el la *build* del projecte que s'ha desenvolupat en aquest treball. Aquesta versió ha estat la facilitada a diferents persones per el seu posterior anàlisis.

https://drive.google.com/file/d/15Tkzdimb5hSL8_RyJXdb9OmCBbfbWmkU/view?usp=sharing

Build del videojoc del projecte (versió 2.0)

També es concedeix la versió 2.0 de *The Eye* amb tots els canvis en el codi que s'han realitzat després de l'enquesta.

https://drive.google.com/file/d/1d9AKhPHBBx7fwUP6BDQn61o8zVz9FRjJ/view?usp=sharing

Annex 2

Projecte de Unity (github)

Mitjançant un link al repositori utilitzat durant la realització del treball, es permet l'accés al projecte per el seu anàlisis o correcció.

https://github.com/AlvaroOlivetGimeno/TFG-ALVARO_OLIVET_GIMENO-v_1.0.git

Projecte de Unity (carpeta al drive)

Per assegurar que tothom pugui aconseguir l'arxiu de Unity, també es facilita el projecte per mitja d'una carpeta pública al drive.

https://drive.google.com/file/d/14yqtlitId33iagjEM-4qxEUUJSQPHmdA/view?usp=sharing

Annex 3

Enquesta

També es presenta les diferents preguntes que es van facilitar a les persones que van provar la versió 1.0 del videojoc.

Pregunta 1:

Has aconseguit superar The EYE?

- Si

- No

Pregunta 2:

Has entés tot allò que el joc presenta? (controls, parry, enemics, habilitats, missions, sistema econòmic...)

- Si
- No

Pregunta 3:

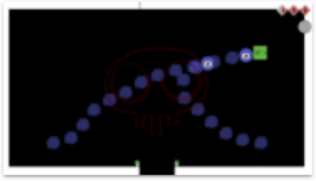
Has trobat erros (bugs, problemes, sistemes injustos...) durant el joc?

- Si
- No

Pregunta 4:

Quines mecàniques, d'entre les següents, has trobat en el joc:

- [ ] Enemics que deixen anar un líquid enganxos



- [ ] Augment del nomobre de les sales d'enemics
- [ ] ParryShield (escut que impedeix causar dolor als enemics)

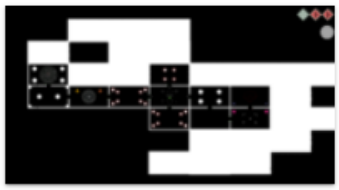
- [ ] Augment del dany que causen els enemics cap al jugador (de mitja vida a una de sencera)
- [ ] Desbloqueig de les missions vermelles

- [ ] Desbloqueig de les missions blaves

- [ ] Augment dels productes de la botiga (de 3 a 5)

- [ ] Aparició d'un altre 'mag'

- [ ] Obtenció del mapa



Pregunta 5:

Consideres que has millorat gradualment mentre jugabes?

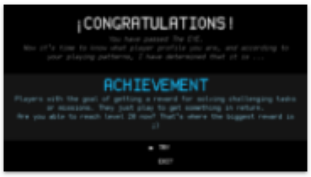
- Si
- No

Pregunta 6:

En cas d'haver superat el nivell, quin perfil ha estat el resultant?

- Action

- Achievement



- Maestry

- Creativity

- No he superat The Eye.

Annex 4

Video del gameplay de *The EYE*

Per últim deixa un link a *Youtube* del *gameplay* complet del videojoc:

https://www.youtube.com/watch?v=pmyrbBvAvDk

Bibliografia

1. Aversa, A. [Roguelike Celebration Youtube Channel]. (2020, 15 octubre). *The end of permadeath* [Vídeo]. YouTube. https://www.youtube.com/watch?v=kn2QkH5lByw&list=PLi7jNGNQhwdhKzh2I7NNJTxHjQEVejLxm&index=4
2. Ballester Hernández, B. (2016, septiembre). *Diseño de una interfaz inteligente y adaptativa para un LMS*. Universidad de la Laguna. https://riull.ull.es/xmlui/bitstream/handle/915/3080/Diseno%20de%20una%20interfaz%20inteligente%20y%20adaptativa%20para%20un%20LMS.pdf?sequence=1
3. Bartle, R. (1990). *Hearts, Clubs, Diamonds, Spades: Players who suit MUDs*. https://drive.google.com/file/d/1eaTt-wLXqcQiwbgewpo1VvxV7TGOsuTCK/view?usp=sharing
4. Bateman, C., Nacke, L., & Mandryk, R. (2014). *BrainHex: A neurobiological gamer typology survey*. ELSEVIER. https://dropsdejogos.uai.com.br/wp-content/uploads/sites/10/2016/05/https%3A__hcigames.com_wp-content_uploads_2015_01_BrainHex-A-Neurobiological-Gamer-Typology-Survey.pdf
5. Brown, L. [Roguelike Celebration Youtube Channel]. (2020, 15 octubre). *Why do I even like roguelikes? An exploration of player motivation* [Vídeo]. YouTube. https://www.youtube.com/watch?v=BjqZdMW8U9Q&list=PLi7jNGNQhwdhKzh2I7NNJTxHjQEVejLxm&index=2
6. Cortesi, D. (2018). *REINFORCEMENT LEARNING IN ROGUE*. https://core.ac.uk/download/pdf/211575307.pdf

7. Edmun McMillen (2011). The Binding of Isaac (PC) [Videojoc]. Edmund McMillen and Florian Himsl.

8. Game Developers Conference [GDC Youtube Channel]. (2017, 12 mayo). *Best Practices for Procedural Narrative Generation* [Vídeo]. YouTube. https://www.youtube.com/watch?v=k2rgzZ2WXKo

9. Jo Kim, A. (2014). *AmyJoKim*. https://amyjokim.com/blog/2014/02/28/beyond-player-types-kims-social-action-matrix/

10. Marczewski, A. (2012). *Marczewski's Player and User Types Hexad*. Gamified UK. https://www.gamified.uk/user-types/

11. *StackOverflow*. (s. d.). StackOverflow. https://stackoverflow.com/

12. Stewart, B. (s. d.). *Personality And Play Styles: A Unified Model*. Gamasutra. https://www.gamasutra.com/view/feature/6474/personality_and_play_styles_a_.php?print=1

13. Unity. (s. f.). *Unity Application*. Unity Scripting API. https://docs.unity3d.com/ScriptReference/Application.html

14. Yee, N. & Ducheneaut, N. (2005). *Quantic Foundry*. https://quanticfoundry.com/#motivation-model