Treball de fi de grau

Game Desing i propototip jugable d'un videojoc amb estil pixel-art

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Alumne: Xavier Serra García

Director: Oriol Boira Ricart

Convocatòria: juliol de 2015

Índex

1 INTRODUCCIÓ

La finalitat d'aquest treball de fi de grau és la realització del prototip d'un videojoc en **Unity**, basant el seu desenvolupament en tots els coneixements apresos al grau multimèdia, i d'aquesta manera ser capaç d'observar les dificultats i obstacles que poden aparèixer en un projecte d'aquestes característiques.

El videojoc a desenvolupar es va realitzar amb una gràfica **pixel art** i està orientat a un ordinador de sobretaula, encara que, com observarem més endavant, el mercat ha variat en gran mesura durant el desenvolupament del joc i s'exposaran altres alternatives de mercat.

Per començar el desenvolupament d'un producte d'aquestes característiques, prèviament es va realitzar un estudi sobre el mercat i les tecnologies actuals, situant un context òptim on poder començar a treballar i prendre decisions correctes sobre la idea, producció i postproducció del joc. Aquest pas és molt important ja que s'ha de posicionar bé el producte i fer-lo destacar entre milers i milers de jocs nous que apareixen cada dia.

No només es van estudiar tècniques útils per a la creació de videojocs, sinó que es van afegir casos d'èxit, fracassos, tecnologies actuals, i per què utilitzar una o una altra metodologies de treball i de gestió àgils.

1.1 Objectius generals

- Estudiar el mercat de videojocs actual.
- Comparar les tecnologies més actuals per al desenvolupament de videojocs.
- Realitzar prototip jugable d'un videojoc de plataformes orientat a ordinador de sobretaula en Unity.

1.2 Objectius secundaris

- Estudi del mercat actual de videojocs.
- Estudi i anàlisi de videojocs semblants al projecte.
- Estudi i anàlisi de tecnologies actuals de desenvolupament de videojocs.
- Realitzar planificació del projecte.
- Aplicar SCRUM al projecte.
- Realització del prototip jugable.

1.3 Objectius personals

- Ser capaç de realitzar un videojoc des de 0 fins al final, posant a prova al màxim les meves habilitats de programació.
- Intentar aconseguir un producte viable per a una publicació jugable online.

2 TECNOLOGIES ACTUALS

2.1 Desenvolupament

Les tecnologies actuals han avançat moltíssim els darrers anys en relació al món dels videojocs. Més enllà de la potència que poden oferir de cara a l'usuari, també s'han dedicat molts esforços per fer el desenvolupament més accessible i ràpid de cara als desenvolupadors.

Això son bones i males notícies. Per una banda el desenvolupament de videojocs és més accessible i ràpid, donant la possibilitat de desenvolupar jocs en setmanes, inclús dies. Però d'altra banda, al facilitar tant el desenvolupament, la quantitat de jocs desenvolupats a augmentat de manera exponencial, el que dificulta moltíssim destacar en un mercat tant massificat.

2.2 Mercat i plataformes

Inclús amb la massificació de videojocs que pateix el mercat actualment, el nombre d'usuaris que poden accedir a comprar o jugar un videojoc a augmentat de manera exponencial. El mercat s'està obrint a usuaris que mai abans havien provat un videojoc. Dones de mitjana edat, nens menors, de menys de 6 anys, i inclús persones majors de 65 anys que estan començant a descobrir aquest món. Tot això es gràcies als mòbils. Milions i milions d'usuaris diaris amb una facilitat enorme d'accedir al videojoc.

Amb jocs com Clash of Clans o Candy Crush generant més d'un bilió de dòlars al dia, i amb uns DAU (Daily Active Users) de 5 milions d'usuaris, tenint en compte que són jocs freemium, queda molt clar qui té el control del mercat actualment.

No obstant, els jocs per a consola segueixen sent rentables. Considerem Assassin's Creed, on la darrera llicència d'aquesta franquícia ha venut més de 70 milions d'unitats, o GTA V que va recaptar més d'un bilió de dòlars durant els primers tres dies després de sortir al mercat.

Després de les consoles i els mòbils queda una altra opció, l'ordinador. Actualment un joc per ordinador de retail (compra física) ja no es 100% rentable, ni la millor aposta de negoci, però han aparegut plataformes que ajuden a la distribució d'aquests videojocs. Una d'aquestes plataformes, la més exitosa, es Steam que compta amb més de 60 milions d'usuaris i més de 6 milions d'usuaris actius. L'entrada a Steam d'un videojoc, però, es complexa ja que s'ha de passar un filtre molt agressiu.

Tenint tots aquests factors en compte, el més profitós és anar a Mobile. Però amb la tecnologia actual és possible realitzar projectes 'cross platform', és a dir, jocs que funcionen en totes les plataformes. Més endavant s'explicarà com i per què realitzar un joc 'cross platform'.

2.3 Tecnologies

La tecnologia utilitzada per al projecte depenia de varis factors. Primer de tot, i el més important, és saber per a quina plataforma anirà destinat el projecte. Depenent de la plataforma, serà necessari realitzar el projecte amb un llenguatge de programació o un altre. A continuació s'exposen els llenguatges més importants actualment:

- **Objective-C** o **Swift** son llenguatges orientats solament a IOS, tancant el mercat del joc a una sola plataforma.
- **Java** es un llenguatge que es podria utilitzar per a PC, Android o Windows Phone.
- **HTML5** (javascript) es un llenguatge 'cross platform', es podria utilitzar per a qualsevol dispositiu amb un navegador, o inclús compilar per a plataformes natives com Android o IOS utilitzant phoneGap, cordova, o altres compiladors.
- **CSharp** o **Javascript** amb Unity. Gràcies a Unity es pot compilar a quasi totes les plataformes possibles, incloent smart TV's o consoles.
- **C++** amb Unreal Engine, també 'cross platform', per a diverses consoles, PC i mòbils.

De totes aquestes tecnologies, les més òptimes per a un equip petit / mitjà de persones, entre 1 i 7 aproximadament, serien Unity, Unreal Engine i HTML5.

2.4 Unity vs. Unreal Engine vs. HTML5

Deixant de banda els llenguatges o les tecnologies mono-plataforma. Quin llenguatge s'ha d'utilitzar per a un videojoc avui en dia?

Unity, Unreal Engine i HTML5 ofereixen un producte multi-plataforma de gran qualitat, però hi ha unes petites diferències que han de ser estudiades abans d'escollir una de les tres tecnologies.

2.4.1 Unity

Unity es un IDE de desenvolupament de videojocs centrat al 3D, encara que des de la versió 4.3 ha millorat molt el desenvolupament de videojocs 2D. El llenguatge utilitzat per programar amb Unity es C# (c sharp) o javascript.

Unity es perfecte per al desenvolupament de tot tipus de jocs, encara que està molt més enfocat a jocs d'una grandària petita/mitjana en 3D. Gràcies a les seves integracions amb la majoria de consoles i dispositius, on es possible exportar un joc quasi sense canviar el codi base (normalment només s'han de retocar dimensions i inputs), l'àmplia documentació, tutorials, i a la interfície gràfica que ofereix, es va considerar Unity com una opció més que vàlida per aquest projecte.

2.4.2 HTML5

HTML5 és una tecnologia bastant nova, que està creixent moltíssim els darrers anys. El seu rendiment està millorant molt, i a l'aparèixer el canvas s'ha convertit en una de les eines més òptimes per a realitzar jocs de navegador, ja que es poden reproduir en quasi qualsevol dispositiu.

Al ser una tecnologia poc madura, hi ha navegadors i dispositius que encara no soporten al 100% el canvas, però tot apunta a que serà una de les tecnologies per a videojocs més importants dels pròxims anys. Treballar amb canvas directament és bastant complex, però s'han creat llibreries i frameworks que permeten un desenvolupament més ràpid i accessible per als programadors.

PhaserJS o **CreateJS** son els frameworks més utilitzats, proporcionen una API amb moltíssimes aplicacions i ajudes orientades al desenvolupament de videojocs, com animacions, transicions, gestió i càrrega d'imatges, etc.

El desenvolupament amb HTML5, d'altra banda, es més costós que **Unity** ja que encara que es pugui crear un projecte de zero amb força rapidesa, gràcies als frameworks esmentats, generar un entorn de treball amb automatització de tasques, muntar el servidor on poder llençar l'aplicació i exportar-ho pot resultar en una feina molt costosa per a algú amb poca experiència en coneixements avançats de javascript des de zero.

El **rendiment**, al ser utilitzar directament en navegador, és el més **precari** de les tres opcions. Per tant es recomana per a projectes petits.

2.4.3 Unreal Engine

Unreal Engine es semblant a **Unity**, proporciona un IDE i una API preparada per al desenvolupament de videojocs i amb més potencia que Unity a l'utilitzar per sota el llenguatge **C++** que dona molta més llibertat per alliberar memòria i processos de manera manual. Per tant, tindria que ser la eina per excel·lència, tot i això, el ser més potent també incrementa la **complexitat** i el **temps de desenvolupament** dels projectes.

És una opció a considerar ja que en les últimes actualitzacions s'ha orientat molt l'eina a desenvolupament per a dispositius mòbils, s'ha tornat una eina gratuïta i a afegit una store amb assets i snippets que ajuden al desenvolupament.

2.4.4 Tecnologia utilitzada

Entenem que **Unreal Engine** és una eina per a projectes una mica més grans del que s'ha plantejat aquest, inclús arribant a ser òptima per a projectes AAA, ja que necessita un equip de persones molt més gran i amb molta experiència. **HTML5** és una tecnologia vàlida i accessible, però per a projectes que requereixen de varis escenaris, molts gràfics, animacions, físiques, etc, es queda una mica curt. A més la poca experiència amb la tecnologia faria el procés molt lent, costós i poc rendible. Per tant, com a opció més factible, es va escollir **Unity**, ja que oferia les característiques perfectes per al desenvolupament del joc; rendiment necessari per a la execució de varies escenes i molts gràfics de gran qualitat.

Plugins necessaris per a certs elements del joc i tot això sense afegir l'excessiva complexitat que afegia Unreal Engine. Es va orientar el joc a PC, gràcies al sistema d'exportació de Unity, i seguidament es va intentar exportar el joc a mòbil.

2.4.5 Altres tecnologies necessàries / Git

Git es una eina d'emmagatzematge de codi que dona moltíssimes millores per al desenvolupador.

Git ofereix la possibilitat de dividir la feina en "commits", així si un commit fa que es trenqui alguna part de l'aplicació, el desenvolupador ràpidament pot mirar què ha passat comparant el codi amb una versió anterior, o inclús eliminant la última versió.

La part més bona de Git es que cada cop que es puja a internet una versió, només es pugen els canvis a les línies de codi i no el fitxer sencer, per tant és molt ràpid i segur.

3 METODOLOGIES DE TREBALL

3.1 Scrum

Scrum és una **filosofia** o metogologia de treball **'àgil'** definit per Ikujiro Nonaka i Hirotaka Takeuchi a principis dels anys 80.

Avui en dia Scrum es utilitzat per moltes empreses de desenvolupament de software, degut a la seva **flexibilitat** i **rapidesa** d'execució. Per a aquest projecte es va utilitzar el mètode Scrum simplificat, s'explicarà a continuació com es va implementar aquesta metodologia al projecte.

Scrum defineix un conjunt de pràctiques i rols en un equip multidisciplinar, que es poden prendre com a punt de partida per a definir un procés de desenvolupament que s'executarà durant el projecte. Gràcies a aquest conjunt de bones pràctiques Scrum permet la creació d'equips **autogestionats**.

Els rols principals en Scrum son: el Scrum Master, el Product Owner i l'equip.

El **Scrum Master** s'encarrega de facilitar l'aplicació de Scrum en el projecte i d'eliminar qualsevol obstacle que impedeixi que l'equip arribi al objectiu del sprint. El **Product Owner** és la veu del client i l'equip s'encarrega del desenvolupament del projecte.

3.2 Introducció

Les tasques es divideixen en Sprints, normalment d'una a quatre setmanes, intentant sempre que siguin el més curts possibles.

En aquest projecte no s'han pogut aplicar rols, ja que no hi ha un equip sinó una persona. Els sprints s'han realitzat en 4 setmanes.

Al començar el projecte es separen les tasques en històrics i un cop definits els requisits d'acceptació de cada històric es divideixen en petites tasques.

Es fa una petita aproximació de la dificultat de cada tasca i un cop acabada aquesta aproximació es van seleccionant tasques fins a sumar una quantitat de punts de dificultat assignada com a màxima per als sprints. A partir d'aquí no es poden assignar més tasques per al sprint actual.

Les tasques que queden pendents fora del sprint es queden en un apartat anomenat backlog, d'on s'agafaran en el sprint següent.

3.3 Gestió de tasques

Durant el sprint es col·loquen les tasques en Post-its o s'utilitzen eines online, com Trello o Jira, per a gestionar les tasques.

Es divideix la feina en tres apartats:

- To Do: tasques que estan per fer.
- In Progress: tasques que s'estan realitzant en aquest moment.
- Done: tasques acabades.

Cada dia s'ha de realitzar una "daily", una reunió de 15 minuts, on els membres de l'equip estan en peus i cadascú explica que va fer el dia anterior, que farà aquell dia i si hi ha algun obstacle que impedeixi finalitzar les seves tasques. En aquest projecte es van ometre les "daily" ja que l'equip era d'una persona.

3.4 Post Sprint

Al final del Sprint es realitza el "Sprint Review Meeting", on s'analitza si s'ha complert l'objectiu, que ha fallat, etc.

També s'executa una "Sprint Retrospective", on els membres de l'equip donen una visió subjectiva de les seves impressions sobre el sprint. Amb això s'espera una millora per al sprint següent.

4 ESTUDI DE MERCAT

Actualment fer un estudi de mercat sobre jocs és molt complicat si no s'acota l'estudi a una tipologia de joc específica. La gran diversitat de tipus de joc, el gran contrast de públic objectiu i la plataforma de joc són factors que afecten molt al mercat del joc en concret, podríem dir que fins i tot son mercats totalment diferents.

S'intentarà extreure els trets comuns que tenen els jocs per fer-los servir com un estàndard. A més, també es poden veure coses que no han funcionat o veure coses que no s'havien pensat en un primer moment però que han resultat ser una bona idea.

Tenint en compte que la idea era realitzar un joc per a ordinador, de plataformes, i amb referències als primers jocs de plataformes, es va fer una recerca de jocs amb característiques semblants.

Els jocs estudiats van ser:

- Limbo
- Spelunky
- Terraria

4.1 Limbo

Limbo és un videojoc del tipus **puzzle-plataformes** desenvolupat per la companyia independent **PlayDead Studios**. Aquest és el primer títol de la companyia danesa, i va ser llençat el 21 de juliol de 2010 en Xbox Live Arcade. El joc és un "Scroll parallax" en 2D, amb un sistema de físiques que aplica tant al personatge com als elements de l'entorn, el jugador guia a un nen sense nom a través de perillosos escenaris intentant evitar la multitud de paranys amagats en la foscor.

És un joc bastant lent, però que gràcies al so ambiental, al disseny tètric i al level design, ofereix una experiència **única** i **molt original**. Es força un guió molt psicològic, sense textos, sense converses, simplement un noi que busca una noia. Ja que la idea principal del projecte era molt semblant a la de Limbo, d'anar a buscar a una noia, i de la **dificultat extrema** d'arribar a trobar-la, es va decidir prendre'l com a referència.

El sistema de joc va ser dissenyat pensant evitar els errors d'altres jocs on es repeteix la mateixa mecànica de joc constantment.

4.2 Spelunky

Spelunky és un videojoc indie d'acció i aventura creat per **Derek Yu** i inicialment llençat com a freeware per a Microsoft Windows. Després va ser llençat per a Xbox 360, PlayStation 3 i Playstation Vita.

El jugador controla a un espeòleg que explora una sèrie de cavernes col·leccionant tresors, salvant damisel·les i evadint trampes. La primera versió del joc va ser llençada el 21 de desembre de 2008.

Joc molt ràpid, amb falta de guió i immersió, però molt divertit i additiu.

La seva velocitat no era un punt a seguir, però la manera d'implementar els nivells 2D, i la **fluïdesa del moviment** i dels salts del personatge van servir de referència directa per al projecte.

4.3 Terraria

Terraria és un videojoc d'acció i aventura de món obert produït de forma independent per l'estudi **Re-Logic**. Té característiques tals com l'exploració, l'artesania, la construcció d'estructures i combatre diversos monstres. El 16 de maig de 2011 es va llançar. S'estima que el joc va vendre al voltant de 50.000 còpies el dia del llançament, amb més de 17.000 jugadors en línia al mateix temps. En associació amb la productora 505 Games, el joc va ser llançat per a les plataformes XBOX 360, PS3 i PSvita on les seves vendes totals s'estimen al voltant d'un **milió de còpies**. El joc també va ser llançat per als sistemes Android i IOS aconseguint en aquestes plataformes 1.3 milions de còpies descarregades.

Un joc lent, es necessita molt de temps per aconseguir materials, crear objectes i avançar. Tot i així es un referent molt gran del joc que es va crear ja que l'exploració, les mecàniques, el disseny i el so són molt semblants als que es volien per al projecte.

Es va seguir també la manera de **publicitar** el joc de l'estudi Re-Logic, creant un **blog** i fent updates amb imatges del joc i petites explicacions del que s'anava afegint.

5 GAME DESIGN

A continuació s'explicarà el game design del joc realitzat. S'ha de tenir en compte que el prototip no conté totes les característiques que es descriuen a continuació, però sí les mecàniques bàsiques de joc, així com el nivell tutorial i la base del primer nivell.

5.1 Tipologia de joc

Aventura / Plataformer on es posarà a prova l'habilitat mental i els reflexos de l'usuari mentre s'encarrega de què el protagonista eviti tots els obstacles i resolgui puzles amb l'objectiu d'arribar al final de cada nivell. Tot això estarà dins d'una història profunda amb cert caire crític.

El narrador explicarà tots els successos de la història com si fos un conte. Els personatges parlaran de manera incomprensible a l'usuari. Es busca captar l'atenció amb canvis de guió inesperats, sorpreses en forma de diàlegs o aparicions espontànies i amb diferents tipus de paranys amagats que acabaran amb la paciència del jugador.

El fet que els paranys siguin tan complicats (sempre respectant una corba de dificultat dins els mínims acceptables), fan que la dificultat del joc s'incrementi molt i el jugador rebi una alta recompensa intrínseca en finalitzar un nivell.

És un joc dedicat a jugadors que busquen un repte, un joc que els faci arribar als límits de les seves habilitats en jocs de plataformes, però alhora aportant un guió fort i amb un missatge clar.

5.2 Vista de l'usuari

2D amb scroll horitzontal i vertical, a més de comptar amb diferents escenes. Cada vegada que el personatge desapareix de la pantalla, apareix a una escena nova. La càmera segueix al protagonista en tot moment. Amb aquest efecte es busca dinamisme, separació d'entorns i sensació d'escenaris més grans.

Amb el canvi d'escena s'aconsegueix separar totalment una sala d'una altra, podent canviar l'escala de la càmera, la posició d'aquesta, l'ambient, la il·luminació, també es poden afegir events que apareguin a l'entrar a una sala, salvar la partida, etc.

5.3 Mode de joc

El mode de joc serà en 3era persona, ja que és un joc amb scroll i de plataformes en 2D, és el mode de joc òptim, generant a més a més una empatia de l'usuari amb el personatge principal al veure com pateix durant tots els nivells. Aquest mode de joc permet una interacció amb l'entorn molt més usable i ràpida.

5.4 Objectius

- Arribar al final del nivell evitant els paranys i realitzant els puzles.
- Intentar recuperar a la noia, sense caure a la frustració nivell rere nivell, veient que es completament impossible.
- Trobar tresors i descobrir pistes per a resoldre puzles posteriors.
- Recollir relíquies per aconseguir més vides.

5.4 Estil de grafisme

Es va decidir realitzar el joc amb un estil pixelart, molt treballat, amb molts detalls, per donar molta vida i color a l'escenari sense caure a la vall de la incertesa.

Exemple grafisme

5.5 Marc MDA

5.5.1 Mecàniques

Les mecàniques són saltar, córrer, escalar i interactuar amb l'entorn. Això dona l' oportunitat a l'usuari de moure's per tot l'escenari amb llibertat i podent interactuar amb elements que prèviament s'han mostrat com a interactables (cofres, escales, palanques). Amb això i res més l'usuari ha de ser capaç de sobrepassar tots els reptes de l'escenari, no hi ha power ups, ni doble salt, ni res que pugui facilitar la resolució d'un nivell.

5.5.2 Dinàmiques

Tenint en compte les mecàniques esmentades abans, això permet a l'usuari explorar el mapa, on realment el que busca és el necessari per poder passar de nivell, ja sigui una pista, una palanca o un cofre on pot trobar relíquies que augmentaran la seva puntuació final i sumarà un vida al jugador.

El sistema de vides farà que l'usuari hagi de prioritzar si anar cap a la sala final del boss o buscar les relíquies per aconseguir més vides. Un cop s'entra a la sala del boss, si l'usuari es queda sense vides tornarà al principi del nivell. Si entra a l'escenari del boss amb més vides tindrà més possibilitats de derrotar-lo. Tot i així, aconseguir relíquies a vegades pot ser molt difícil i pot provocar la mort del jugador. S'obre així la dinàmica de prioritzar tasques, afegint un punt d'estratègia i trencar amb la linearitat de la partida.

5.5.3 Estètica

Al no poder mostrar emocions amb les cares, les veus, o la gesticulació dels personatges, es va trobar la manera de mostrar intencionalitat amb el color de l'ambient i la música.

Els tons de colors varien segons el moment de la partida, al començament, quan l'usuari encara no ha mort ni una vegada, tot està ple de colors vius, alegres, sembla que serà una aventura fàcil, s'intenta relaxar a l'usuari, ja que després es trobarà un repte molt gran, després de cada nivell de dificultat extrema es realitzarà un nivell de descans de transició, on la dificultat no serà només molt més fàcil sinó que els colors tornaran a ser vius i alegres, per donar una sensació relaxant intentant treure una mica de la frustració que pot haver acumulat l'usuari al darrer nivell.

Els colors utilitzats als nivells que representen un repte per al jugador s'utilitzen colors foscos y poc saturats, ja que així fan més difícil que es vegin totes les pistes, recursos, paranys i secrets que amaguen els nivells. El que en un joc normal una plataforma no es distingeixi

molt d'una altra pot semblar un error d'usabilitat, en aquest joc es va buscar aquest efecte per incrementar la dificultat.

Finalment es van utilitzar un tercer tipus de colors encara que no estan implementats al prototip entregable, els escenaris dels enemics finals de cada nivell. Es va buscar una il·luminació molt fosca, on els colors que destaquen son els de l'enemic, sempre amb colors brillants, com un animal tòxic de la selva.

Deixant de banda el color, l'estil gràfic s'adapta a un entorn natural, amb roques, arbres, matolls, etc. Tenint en compte que una civilització d'aborígens viu en aquest entorn, per la qual cosa els elements construïts sempre seran artesans, de fusta o pedra, rústics i exòtics.

La música utilitzada es royalty free. Al menú s'ha utilitzat una musica relaxant, tranquil·la, acompanyant el paisatge que es veu, la situació del personatge abans de l'aventura, quan està a la vora del foc amb la seva dona.

A la introducció no s'ha col·locat cap mena de música per donar èmfasi a la història que s'explica al començament. El que si que s'ha introduït es un so per a que no es quedi mai en silenci, ja que es angoixant per a l'usuari, sembla que alguna cosa no funcioni bé.

La música segueix les mateixes característiques que els colors, a les zones de descans s'utilitza una música alegre i calmada, a les zones que ofereixen un repte s'utilitza una música més trista, fosca i cavernosa i, per últim, als enemics finals s'utilitza una música enèrgica i elèctrica.

5.6 Jugabilitat

5.6.1 Breu definició de la història

Una parella d'exploradors acampa en el bosc, discuteixen molt, van a dormir i quan es desperta el protagonista la noia ha desaparegut. El rastre el condueix a l'entrada d'una cova, on succeirà tota la historia.

El protagonista lluitarà per trobar a la seva esposa dins de l'enorme cova, lluitant contra els paranys i puzles que els natius li han preparat. Lluitarà fins al final, la intenció es donar a entendre a l'usuari que aconseguir qualsevol cosa requereix esforç, per tant al final de cada nivell semblarà que s'ha salvat a la noia, fins que se la tornaran a emportar.

La noia anirà apareixen durant la història, donant pistes del final del joc o de com passar certs nivells. Al final de la història s'hi enfrontarà, ja que consumida pels aborígens intentarà lluitar amb el protagonista amb màgia negra. Un cop derrotada, el dimoni que la posseïa marxarà i el jugador haurà salvat a la noia i haurà acabat el joc.

5.6.2 Definició de les regles i reptes

L'usuari pot moure's per l'escenari lliurement, interactuant amb els elements de l'entorn (cofres, palanques, explorar la cova i resoldre els puzles plantejats en cada nivell).

El joc estarà dividit en nivells. A cada nivell hi ha un puzle que ha de ser resolt per poder passar al nivell següent. Per resoldre aquest puzle s'ha de recórrer l'escenari, esquivant els paranys i descobrint les pistes que ajudaran a resoldre'l.

El repte és que un cop mori el personatge apareix al començament del nivell, perdent tot el que havia avançat. Es busca una dificultat de joc alta, i això la incrementarà, encara que un cop l'usuari hagi mort moltes vegades ja tindrà l'habilitat necessària per no morir durant el reconeixement de la cova. Aquesta recompensa, que l'usuari veurà guanyada quan comenci a moure's pel nivell amb molta més agilitat que al principi.

5.6.3 Definició d'enemics

Els enemics son uns aborígens de la cova. Només s'enfronta contra els enemics al final de cada nivell contra un "boss" i és en format puzle, això vol dir que mai atacarà directament a l'enemic, sinó que l'usuari ha de buscar altres formes d'acabar amb ell, buscant les seves debilitats i aprofitant l'entorn al seu favor. S'explicarà més endavant els dos enemics que es van dissenyar per a aquest projecte.

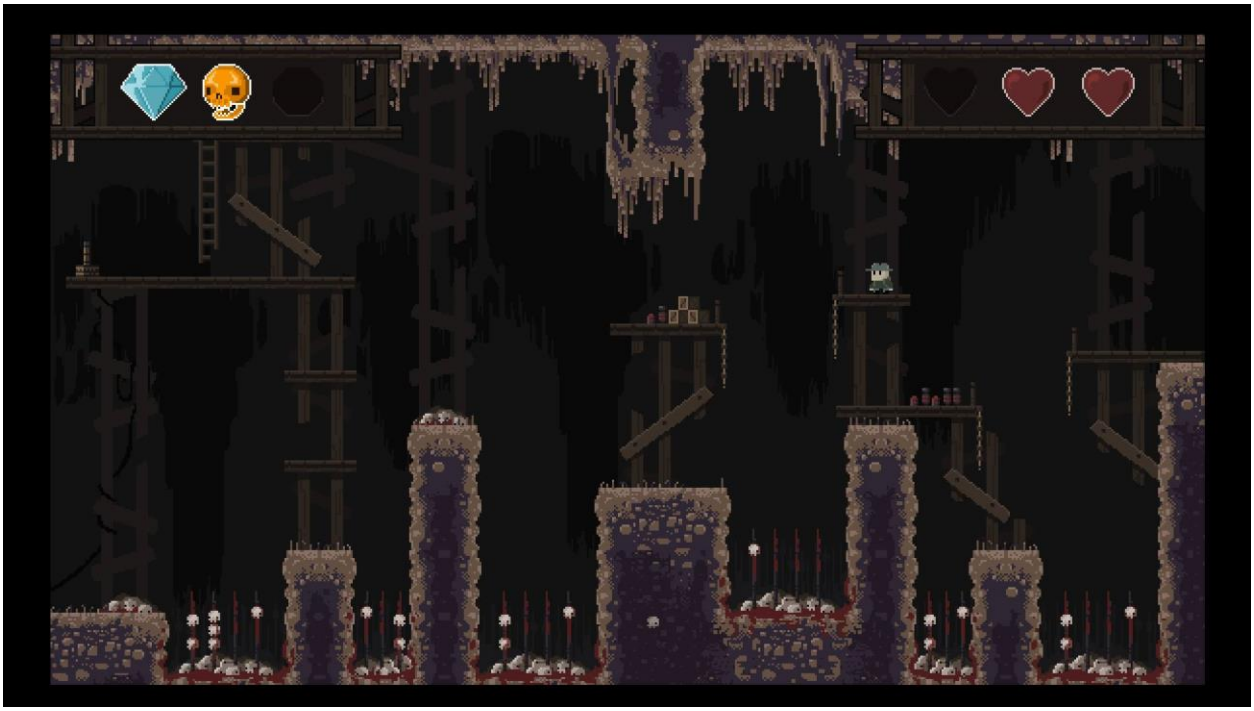
5.6.3 Unitats de contingut

El joc es divideix en tres unitats de contingut, la primera és el joc base, on l'exploració del mapa és important per revelar pistes i trobar la sortida; és on succeeix tota la història. L'altra unitat de contingut són els puzles, que van apareixent a cada nivell per generar un nivell de dificultat afegit i trencar la linealitat del gameplay.

5.7 Interfície

5.7.1 Joc

La interfície que hi ha durant la partida és la que mostra les 3 relíquies del nivell. Hi han 3 relíquies per nivell, amagades en cofres. A més a més mostra la vida actual de l'usuari, que es mostra en forma de cors, això no vol dir la quantitat de vida fins a morir, sinó el número d'intents que té el jugador de passar el nivell sense haver de repetir totes les sales un altre cop.



Interfície de joc

5.7.2 Menú

El menú constarà d'una animació de fons, un so de fons en loop i tres botons, el de començar una partida, el de crèdits i el de continuar partida.

Les estrelles estan animades y el foc també, per donar un efecte de què tot està funcionant correctament i de que el joc no s'ha encallat.

Menú

5.7 Level design

Els nivells són bastant grans, per tenir l'oportunitat d'explorar-los. Hi ha diferents camins per arribar al final, encara que s'haurà de recórrer la cova de la manera dissenyada per ser capaç de passar de nivell, ja que els puzles estaran col·locats correctament per aconseguir aquest propòsit.

Com es pot observar hi han tres tipus de trampes i ídols col·leccionables. Aquesta és la versió de level design del primer nivell.

5.8 Puzles

5.8.1 Primer nivell

El puzle del primer nivell consta de dues parts, la primera part consisteix en buscar els tres símbols amagats pel nivell, evidentment de primera passada l'usuari no acabarà de donar-se compte de que aquests símbols són importants, això sí, un cop arribat a la sala final l'usuari veurà que hi han 3 rodes i 3 palanques, cada una mou una roda i s'han de col·locar els tres símbols per ordre d'aparició. Aquest ordre es pot saber gràcies a la pista que dona la noia en una aparició.

La dificultat d'aquest puzle és tornar a fer el camí enrere, que encara és una mica més difícil, per tornar a veure els símbols, i tornar per últim cop a la sala del puzle. Durant aquest puzle es pot veure com es posa a prova la memòria i l'atenció de l'usuari.

5.8.2 Segon nivell

Durant el puzle del segon nivell es posa a prova l'habilitat del jugador. El nivell conté tres palanques. Les tres s'han d'activar en un cert temps per a que s'obri la porta i un cop oberta l'usuari disposa d'un temps limitat per entrar-hi, de no ser així la porta es tanca i ha de tornar a començar. Tot això evidentment rodejat de paranys i forats per on caure.

El primer repte es descobrir on estan les palanques i per a què serveixen. Seguidament s'ha de pensar en l'ordre exacte d'activació de palanques per a que doni temps a entrar a la porta i per últim s'ha de realitzar l'acció sense morir i molt ràpidament.

5.8.2 Tercer nivell

El puzle del tercer nivell és un dels més frustrants i complicats. Després d'una animació d'entrada, s'afegirà al jugador una nova mecànica; activar una llanterna per il·luminar el nivell. Un cop realitzat el tutorial, el nivell queda a les fosques i el jugador ha de ser capaç de travessar tot el nivell ple de paranys utilitzant la llanterna, però té una duració limitada.

Si la llanterna es gasta, el nivell es queda a les fosques i la probabilitat de superar-lo es molt baixa.

Utilitzar poc la llanterna serà beneficiós per al jugador al monstre final.

5.9 Monstres finals

5.9.1 Primer nivell

El monstre final del primer nivell es un golem, una estàtua de pedra gegant, que tirarà roques al jugador. Aquest ha d'esquivar-les i esperar que el golem descansi per poder arribar a unes escales que hi ha just davant de l'enemic, escalar-les i activar un parany que ferirà l'enemic.

S'haurà de repetir aquesta acció tres vegades, tenint en compte que un cop s'activa la palanca el jugador cau de nou a l'abast de l'enemic. La dificultat s'incrementa després de cada atac.

5.9.2 Segon nivell

En aquest cas l'enemic és un mag que activa trampes de foc als peus del jugador constantment. El jugador ha de pensar un ordre correcte de moviments sinó les flames el cremaran. Un cop esquivat l'onada d'atacs de l'enemic, aquest es posarà a sobrevolar el

jugador. Si el jugador aconsegueix que una de les columnes de foc aparegui a sota de l'enemic el ferirà, així fins a tres vegades, igual que el primer boss, incrementant la dificultat després de cada atac.

5.9.2 Tercer nivell

En aquest cas l'enemic es un monstre fosc, un fantasma. El puzle consisteix a acabar amb el boss abans que s'acabi el temps.

El temps és una llanterna que es va gastant, quant menys hagi utilitzat la llanterna el jugador durant el nivell, més dura durant la escena del boss.

Per eliminar l'enemic, el jugador ha de saltar de plataforma en plataforma, esquivant els atacs. Les plataformes es mouen aleatòriament pel mapa i el jugador ha d'utilitzar la seva habilitat per aconseguir arribar a una palanca saltant de plataforma en plataforma. Un cop accionada, una bola de foc caurà sobre l'enemic.

S'ha de repetir aquest procés tres vegades abans que s'acabi el temps sinó el nivell quedarà a les fosques.

5.9.3 Mecàniques comunes

Com es pot observar els enemics conserven un mateix estil de joc, això dona una sensació de progrés a l'usuari ja que cada cop dominarà més com eliminar els enemics, però al canviar la mecànica de l'atac i la manera d'aconseguir ferir a l'enemic s'aconsegueix que la dificultat del boss s'incrementi i el jugador no caigui en l'avorriment.

5.10 Intencions no implementades

Com en tot projecte, sempre s'han d'acabar retallant coses si es vol finalitzar el projecte amb un nivell de qualitat mínim.

El joc que es va realitzar també incloïa una sèrie de mecàniques que no s'han pogut implementar i que no s'implementaran en un futur, ja que no aporten el que es desitjava.

5.10.1 Enemics

Es van arribar a crear fins i tot dos tipus d'enemics, que van acabar sent eliminats. Durant la creació del joc es va veure que els enemics no encaixaven amb l'estètica, no encaixaven amb la història i no encaixaven amb el flux de partida que es volia aconseguir. Eren molt complicats de derrotar, feien més lenta la partida, la intel·ligència artificial era rudimentària degut a la falta de coneixement per crear-la, etc.

Com que el joc es basava més en l'habilitat d'esquivar paranys i forats i la capacitat de resoldre puzles, els enemics eren un més a més que no aportava res i que trencava molt amb el 'feeling' que intentava crear el joc.

5.10.2 Tenda

Al començament de la creació del joc es volia implementar una tenda, on el jugador pogués millorar la seva armadura i arma. A l'eliminar enemics, la tenda va quedar obsoleta, comprar vides era massa simple i la idea de crear una tenda amistosa dins una cova on teòricament vivien els enemics del protagonista no era massa realista. Per tant es va decidir eliminar-la.

6 ORGANITZACIÓ

Abans de començar el projecte es va preparar una llista de tasques per a poder efectuar el mètode SCRUM correctament.

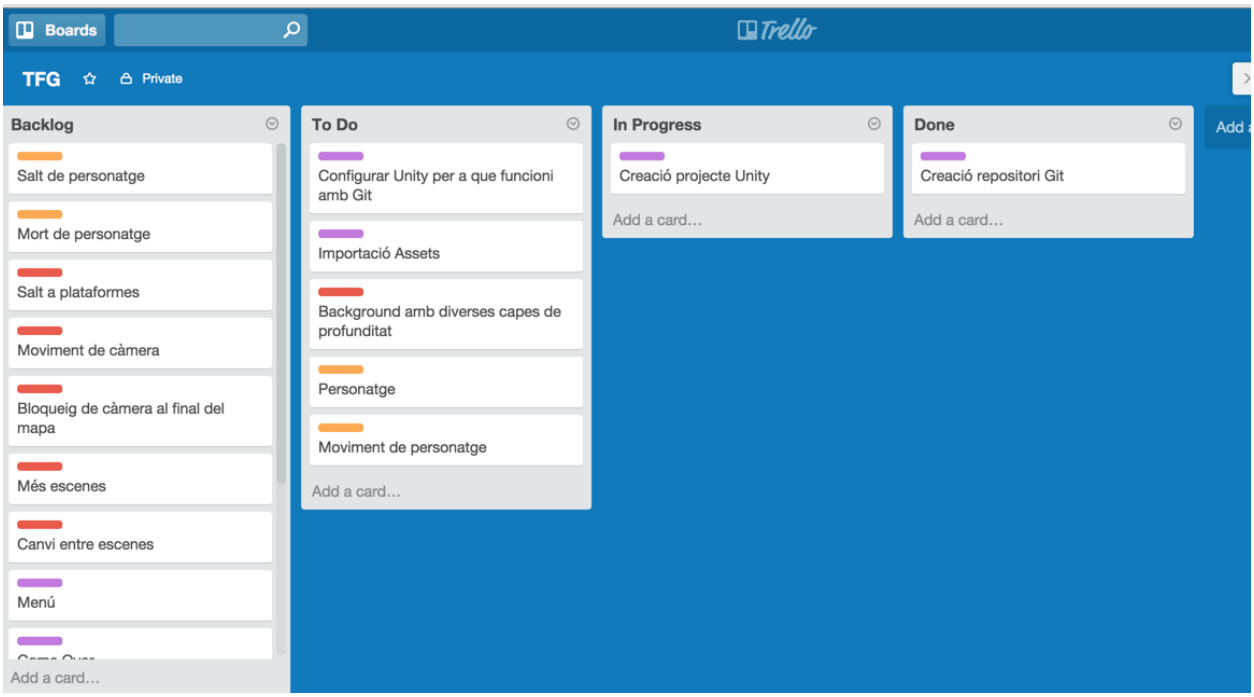
6.1 Tasques

Les tasques es van preparar a nivell tècnic, deixant de banda les tasques de disseny, ja que no entraven com a objectiu del projecte.

- Creació projecte Unity
- Creació repositori Git
- Configurar Unity per a que funcioni amb Git
- Importació assets
- Background amb diverses capes de profunditat
- Personatge
- Moviment de personatge
- Salt de personatge
- Mort de personatge
- Salt a plataformes
- Moviment de càmera
- Bloqueig de càmera al final del mapa
- Més escenes
- Canvi entre escenes
- Menú
- Game Over
- Paranys
- Cofres
- Plataformes mòbils
- 2 tipus de velocitat de moviment
- Diàlegs
- Narrador
- Escales
- Fade in a negre al canvi d'escena

6.2 Trello

Amb aquestes tasques el prototip quedaria preparat per a una demo. Un cop preparades les tasques es va preparar el primer sprint al Trello.



Com es pot observar es van etiquetar les tasques per diferenciar si eren de:

ESCENARI

PERSONATGE

PROJECTE

Per al primer sprint es van puntuar les tasques fins a complir les hores possibles en aquell mes i es van posar les tasques a To Do. Es van realitzar quatre sprints en total per a completar el prototip.

7 PROJECTE UNITY

7.1 Estructura del projecte

Un dels grans problemes d'Unity es la des estructuració completa del projecte, a més de tots els fitxers temporals que crea per a poder fer funcionar l'editor correctament, això implica que s'han de guardar en la carpeta del projecte diferents elements com els prefabs, animacions, controladors d'animacions, scripts, gràfics, elements de UI, escenes, plugins, etc.

Per a no crear un caos absolut, per a aquest projecte es va seguir la estructura de carpetes següent:

Assets /

- Animations /
  - Characters
  - Enemies
  - General
- Controllers /
- Plugins /
- Levels /
- Prefabs /
- Scripts /
- Sounds /
- Sprites /

7.2 Prefabs

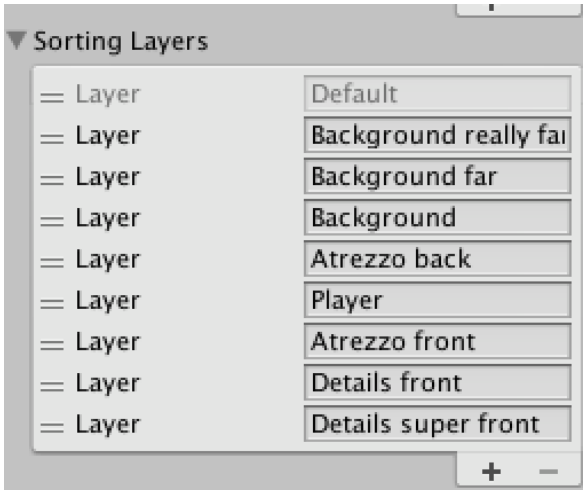
Els prefabs són elements que es poden clonar i utilitzar en diferents parts de l'aplicació. Unity funciona molt acoblat als prefabs, ja que cada objecte que es crea s'ha de convertir en prefab si es volen guardar les seves propietats. S'ha d'anar amb compte si es vol un objecte semblant però amb alguna característica diferent s'ha de crear un altre prefab a partir del primer, ja que sinó editant el prefab s'editarien tots els objectes del joc existents.

En aquest projecte s'han creat diversos prefabs, alguns simplement com a contenidor d'objectes per tenir accés ràpid des del visor del projecte, i d'altres per ser repetits. Els prefabs també es poden utilitzar per a guardar molts elements com a un sol. Així es poden clonar varis elements. Per exemple, en aquest projecte les trampes son prefabs que contenen el sprite i el collider que activa la trampa.

7.3 Creació d'escenaris

A continuació s'explicarà el procés seguit per crear els escenaris a Unity, seguint pas per pas el procés des del començament fins al final.

Al treballar amb Unity 2D no era necessari utilitar l'eix Z per la superposició de capes, ja que ja porta un sistema de capes implementat internament. Primerament es van crear totes les capes necessàries per a la creació d'un nivell.



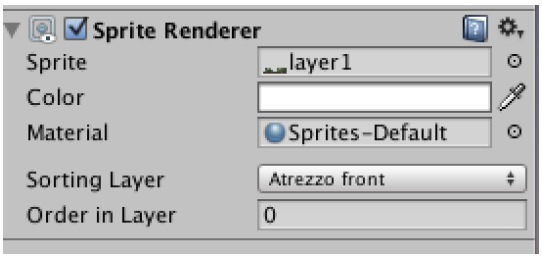
Per a crear les capes s'ha de fer des de l'editor de Tags & Layers, accessible des de, Edit -> Project Settings -> Tags and Layers.

Aquest editor conté totes les capes que renderitzarà l'editor, hi ha unes quantes capes reservades, les altres son lliures a edició.

Es van crear 8 capes, les tres primeres dedicades al background, dividides en really far, far i normal. Després tres més dedicades a la zona del personatge, la primera per al attrezzo que hi ha darrere del personatge, la següent es la capa on es posiciona el personatge, i tot seguit la que s'utilitza per al attrezzo que passa per davant del personatge. Per finalitzar es van afegir dues més per detalls que estiguessin molt endavant, com per exemple, textures i llums.

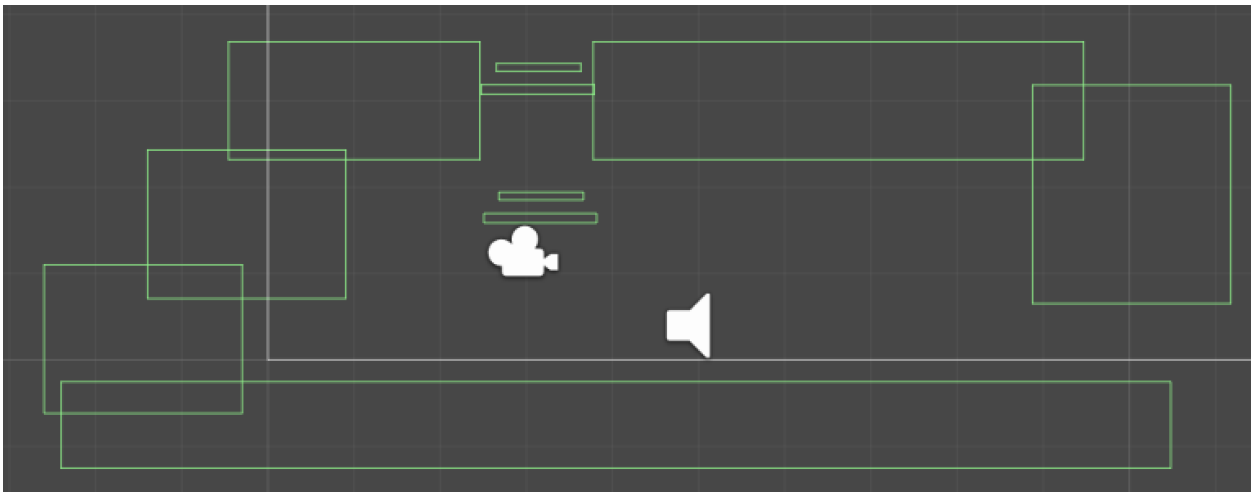
Un cop amb les capes creades, es va començar amb la col·locació d'assets al món. Cada escena es un grup, a dintre d'aquest grup s'hi troben diferents elements. Al començament es crea un grup anomenat layers, aquest contindrà totes les capes gràfiques del nivell.

A dins del grup de layers s'afegeixen Game Objects nous amb un sprite renderer on col·loquem el sprite que es renderitzarà en aquell objecte. Tot seguit li apliquem la capa desitjada seleccionant-la a l'apartat Sorting Layer. També es poden canviar ordres de capes en una mateixa capa amb la opció Order Layer.



Sprite Renderer d'una layer

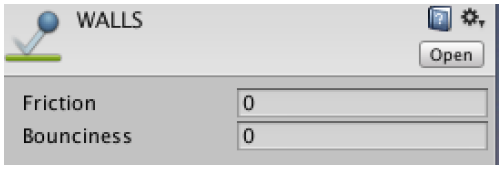
Tot seguit es creen els objectes col·lisionables (terra, parets, pedres, etc). S'han de diferenciar dos tipus d'objectes col·lisionadors:



Col·lisions d'una escena

- Floor, aquest tipus de col·lisionador té una certa fricció (0.4), i provoca que el jugador pugui moure's per sobre sense quedar-se atrapat.

- Wall, aquest tipus de col·lisionador es va crear ja que si s'utilitzava el mateix col·lisionador que el Floor, el jugador podia enganxar-se a les parets i escalar-les. Per evitar això es va crear un objecte Physics2d Material, que permet canviar la fricció i el rebot d'un objecte 2D. Com es pot observar, es va treure la fricció de les parets.



Un cop acabades les col·lisions del personatge, es van col·locar les col·lisions de la càmera, ja que es necessiten per fer l'efecte de que s'ha acabat la escena quan la càmera ja no es mou més.

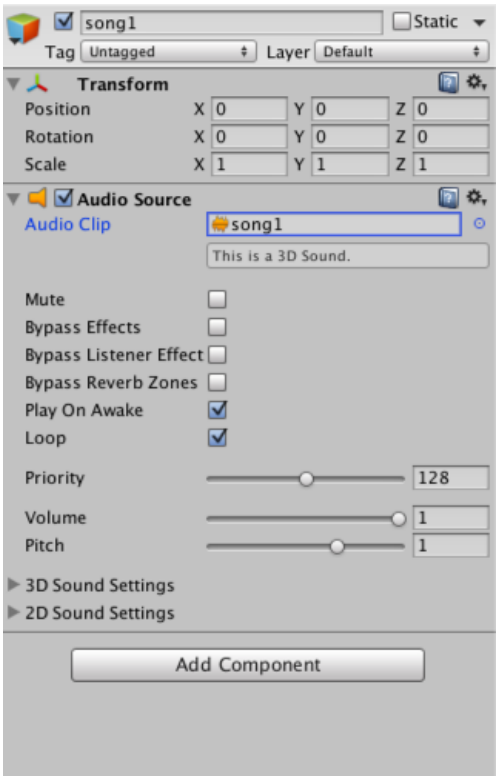
Per acabar de decorar un nivell es van afegir els detalls, ja siguin elements d'atrezzo o trampes, aquests elements s'afegeixen a un grup anomenat Details.

Finalment, com que cada escena està relacionada amb unes altres, es van d'afegir els Gateways o portes, que mouen al personatge i a la càmera a una altra sala, donant la sensació que simplement la travessa. Els Gateways son formats per dos elements, el primer es l'anomenat NextScene, un collider que quan col·lisiona amb el personatge el mou a un lloc predefinit.

L'altre element es l'EntryRight, que serveix de punt de referència del NextScene, conté dos punts per marcar on apareixerà la càmera i el personatge al entrar a l'escena.

7.4 Sons

Per aplicar sons al joc primer de tot s'importa el so al projecte, seguidament s'afegeix un objecte buit a l'escena amb el component "audio source" on afegim el clip de so.



Es configuren els efectes pertinents, com per exemple si es vol que el so s'executi al començar la escena, si es mantindrà en loop o no, el volum, etc. Es pot accedir a aquests paràmetres més tard mitjançant codi.

7.5 Scripts

A continuació s'exposen les funcionalitats més importants, destacades o complexes del joc i les parts dels scripts relacionats, si es vol fer una consulta del codi sencera es pot trobar als annexos.

7.5.1 Càmera

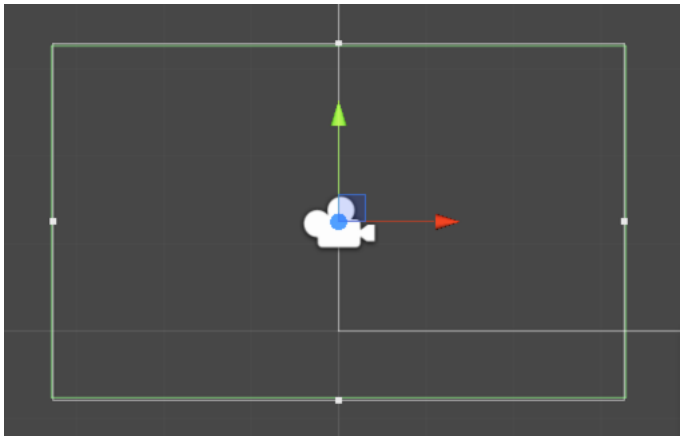
La càmera funciona de la següent manera: segueix al personatge en tot moment centrat a la pantalla, tot i això, si s'arriba al final del mapa, la càmera para de moure's per què no hi ha més nivell, un cop el personatge torna a moure's cap a una zona on hi hagi més nivell la càmera el torna a seguir, com que l'efecte a aconseguir es el d'un joc "retro" no es va aplicar ningun efecte de fade in o out al moviment per fer-lo més suau.

Per a aconseguir això, es va crear un script que aniria amb la càmera, on al seu update, la y i la x de la càmera s'igualarien sempre a la x i la y del personatge, donant l'efecte de seguiment.

```
void Update ()
{
    if (!block) {
        if (canMoveX == true && canMoveY == true) {
                transform.position = new Vector3 (character.position.x, character.position.y + offsetY, transform.position.z);
        }
        if (canMoveX == true && canMoveY == false) {
                transform.position = new Vector3 (character.position.x, transform.position.y, transform.position.z);
        }
        if (canMoveX == false && canMoveY == true) {
                transform.position = new Vector3 (transform.position.x, character.position.y + offsetY, transform.position.z);
        }
    }
}
```

L'update es crida a cada refresc de pantalla, per tant mourà la càmera sempre on estigui el personatge, tenint en compte que no estigui a una banda de la pantalla, on llavors es compta amb una sèrie de booleans que indiquen si es pot moure o no. El boolean "block" el comentarem al següent script.

Aquests booleans que indiquen si la càmera esta al final del nivell o no, s'activen quan la càmera col·lisiona amb uns triggers que hi ha a cada nivell.



Hi ha un per a cada banda del nivell (left, right, top, bottom). Per detectar la col·lisió de la càmera, com el que volem es la col·lisió del punt més llunyà que capta la càmera i no del objecte càmera en si, la càmera conté també quatre triggers, un a cada banda del seu visor.

```
void OnTriggerStay2D (Collider2D other)
{
    if (other.tag == "CameraLeft" && transform.position.x > other.transform.position.x)
    {
        canMoveX = false;
        if (character.position.x >= transform.position.x){
            canMoveX = true;
        }
    }
    if (other.tag == "CameraRight" && transform.position.x < other.transform.position.x)
    {
        canMoveX = false;
        if (character.position.x <= transform.position.x){
            canMoveX = true;
        }
    }
    if (other.tag == "CameraTop")
    {
        canMoveY = false;
        if (character.position.y <= transform.position.y - offsetY){
            canMoveY = true;
        }
    }
    if (other.tag == "CameraBottom")
    {
        canMoveY = false;
        if (character.position.y >= transform.position.y + offsetY){
            canMoveY = true;
        }
    }
}
```

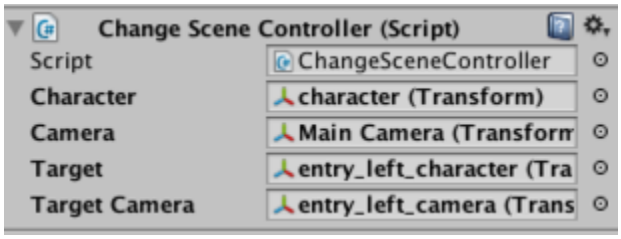
Com es pot observar en el codi anterior, cada cop que col·lisiona amb un dels triggers de final de nivell, es comprova el tag del trigger per saber de quin es tracta i després es comprova la posició de la càmera respecte al personatge, depenent de l'estat la càmera seguirà al personatge o no.

7.5.2 Controlador de canvi d'escena

Es va decidir que el joc no canviaria de escena tal i com presenta Unity el seu sistema d'escenes, és a dir, en aquest cas un nivell podria tenir moltes escenes i només es canviaria d'escena de Unity al canviar de nivell, això es va realitzar per evitar el "flickering" o parpadeig que podia aparèixer al canviar d'escena, ja que al no fer fade outs el canvi havia de ser molt ràpid, i Unity quan canvia d'escena carrega tots els elements i entre escena i escena podria haver un moment que no es veiés res, per tant en ordinadors poc potents podrien ser un problema.

Tenint en compte que el canvi d'escenes es faria manual, es va plantejar de la següent manera:

El canvi d'escena sempre seria al final del nivell, amb un trigger que accionaria el canvi d'escena un cop el personatge el toqués. El canvi d'escena implica que el personatge es tele transporti a la escena següent i que la càmera el segueixi, estigui o no bloquejada per estar al final d'un nivell. Quan el personatge col·lisiona amb un trigger de canvi d'escena (el qual té la escena a la que ha d'anar linkejada amb l'editor) el tele transporta a un punt de l'escena següent controlat amb un objecte buit (mirar imatge del editor).



Com es pot observar, cada "Gateway" o porta a la següent escena conté un script de canvi d'escena amb els objectes següents: el personatge i la càmera, per poder moure'ls des del codi, i dos punts on es mouran els objectes.

```
private void Change() {
    camera.GetComponent<CameraController> ().setBlock (true);
    character.position = new Vector3 (target.position.x, character.position.y, character.position.z);
    camera.position = new Vector3 (targetCamera.position.x, camera.position.y, camera.position.z);
    camera.GetComponent<CameraController> ().setBlock (false);
}
```

El codi és bastant simple. Un cop accionat el canvi d'escena, es bloqueja la càmera (per evitar errors de moviments inesperats de càmera durant la teleportació), es mou el personatge i la càmera i seguidament es torna a desbloquejar.

7.5.3 Controlador del personatge

El personatge és dels scripts més complexos que hi ha al joc, bàsicament controla tot l'input d'usuari. Com que es volia un comportament molt fluït es va insistir molt en millorar aquesta part del codi. Per començar l'input d'usuari es controla al update, per tant a cada refresc del gameLoop es controla si hi ha alguna tecla polsada, per fer-ho més net en comptes de posar tot el codi a la funció update es crida a un mètode anomenat HandleIO. També es compta amb una variable move que conté si s'estan polsant els botons de moviment horitzontal (left, right).

```
//Update function is called every frame, we use this because of the inputs.
void Update() {
    HandleIO();
    move = Input.GetAxis ("Horizontal");
}
```

Aquesta funció separa els inputs i acciona les funcions de córrer, saltar o caminar.

El mètode Run mou el personatge utilitzant el resultat guardat anteriorment a la variable move multiplicat per la velocitat (configurable des de l'editor), i després activa l'animació de córrer.

```
public void Run() {
    rigidbody2D.velocity = new Vector2 (move * runningMaxSpeed, rigidbody2D.velocity.y);
    //If to control the fast running animation.
    if (move != 0) {
        _animCharacter.SetBool ("isRunningFast", true);
    } else {
        _animCharacter.SetBool ("isRunningFast", false);
    };
}
```

El mètode Jump tracta de manera diferent el moviment que el de córrer, en aquest cas s'aplica un vector de força a la y del objecte rigidbody2D que conté el personatge, aquest objecte és el que interactua amb la gravetat, seguidament s'activa l'animació de salt.

```
public void Jump() {
    _animCharacter.SetBool ("isGrounded", false);
    rigidbody2D.AddForce (new Vector2 (0, jumpForce));
}
```

La separació d'utilitzar forces per al salt i moviment estàtic per al moviment horitzontal del personatge és vital per a la fluïdesa i usabilitat del joc. Utilitzar forces per al moviment horitzontal és un dels principals errors dels programadors de videojocs principiants, si s'apliquen forces al moviment fa que el personatge respongui de manera errònia i lenta al input de l'usuari, cosa que converteix el gameplay en una experiència farragosa, lenta, costosa i poc agradable.

Per optimitzar recursos, en comptes de seguir el standard de fer una animació per a cada banda, és a dir, el personatge corrent cap a la dreta i cap a l'esquerra, el que es va realitzar va ser un mètode que rota els sprites al eix z, per tant els inverteix. Es guarda una variable booleana per saber a quina direcció està girat actualment el sprite.

```
//Function to flip the sprite
private void Flip() {
    _facingRight = !_facingRight;
    Vector3 theScale = transform.localScale;
    theScale.x *= -1;
    transform.localScale = theScale;
}
```

La mort del personatge també es controla des d'aquest script, l'únic que executa per al prototip d'aquest projecte és carregar el nivell de nou, més endavant podria carregar un text o una animació per donar més feedback al usuari.

```
public void Die() {
    Application.LoadLevel(Application.loadedLevel);
}
```

7.5.4 Controlador de cofres

El cofre només té dues funcionalitats en aquest prototip, activar un popup indicant que es pot obrir i activar l'animació de obertura en cas de que l'usuari premi la tecla d'acció quan està dins d'un radi d'acció. Per activar o desactivar el popup d'acció es crida a una funció passant l'alpha al qual es vol transformar.

```
void SetAlpha (float value)
{
    infoActionColor = infoActionRenderer.color;
    infoActionColor.a = value;
    infoActionRenderer.color =  infoActionColor;
}
```

Hi ha dos mètodes que controlen si el personatge es aprop o no, OnTriggerEnter2D i OnTriggerExit2D, en aquest mètodes s'actualitza la variable nearChest, tenint això en compte al mètode update, seguint la estructura de codi standard del projecte, es comproven els inputs d'usuari i si s'ha clicat el botó d'acció, s'anima el cofre i s'actualitza una variable que marca el cofre com a obert, per a no poder obrir-lo un altre cop.

```
// Update is called once per frame
void Update ()
{
    if (nearChest && Input.GetKeyDown (KeyCode.E))
    {
        animChest.SetBool ("isOpen", true);
        isOpen = true;
    }
    if (isOpen) {
        SetAlpha (0f);
    }
}
```

7.5.5 Controlador de joc

El controlador de joc només té la funció de canviar de nivells. En aquest cas només carrega el segon nivell, ja que no n'hi ha més, però més endavant s'hauria de fer de manera dinàmica.

```
void changeLevel() {
    Application.LoadLevel (2);
}

public void nextLevel() {
    changeLevel ();
}
```

7.5.6 Controlador de nivells

Aquest script va linkejat a tots els triggers que canvien de nivell, en cas que col·lisioni amb el personatge es crida al mètode de canvi de nivell que conté el controlador de personatge.

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.tag == "Player")
    {
        gameController.nextLevel();
    }
}
```

7.5.7 Controlador del menú

Aquest script té tres funcionalitats principals, tots els mètodes estan linkejats als botons del menú, des de l'editor. Les dues primeres son ocultar i mostrar els crèdits, però com que utilitzen un plugin s'esmentarà el seu funcionament a l'apartat de plugins. El moviment del cel també es realitza amb el mateix plugin.

L'altre mètode és el de començar partida, accionat quan es fa click al botó play. El que fa és amagar els crèdits si estaven mostrats i aplicant una funció creada al script Utils, de la que es parlarà més endavant, oculta tota la pantalla sota una gràfic negre mitjançant un alpha, seguidament carrega el primer nivell.

```
public void startGame () {
    hideCredits ();
    blackBg.GetComponent<Image>().enabled = true;
    Utils.changeColor (black0, black100, blackBg, 3f).setOnComplete(() => {
        Application.LoadLevel (1);
    });
}
```

7.5.8 Controlador de palanques

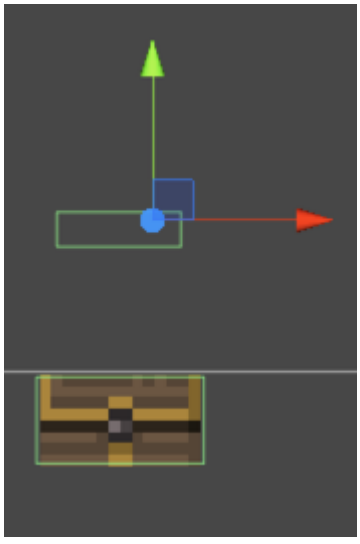
Aquest controlador s'aplica a les palanques que hagin d'activar una plataforma mòbil. Segueix pràcticament el mateix comportament que un cofre per detectar si l'usuari està a prop o no, però a l'activar la palanca canvia de frame per donar feedback a l'usuari de que està activada i activa el moviment de la plataforma associada.

El mètode movePlatform utilitza el plugin LeanTween per moure la plataforma. En cas de que ja s'estigui movent es pot parar el moviment desactivant la palanca.

```
public void SwitchPlatform () {
    if(!isMoving) {
        animSwitch.SetTrigger ("goActive");
        animPlatform1.SetBool("Active", true);
        animPlatform2.SetBool("Active", true);
        isMoving = true;
        MovePlatform();
    } else {
        animSwitch.SetTrigger ("goOff");
        animPlatform1.SetBool("Active", false);
        animPlatform2.SetBool("Active", false);
        isMoving = false;
        LeanTween.cancel (platform);
    }
}
```

7.5.9 Plataformes d'un sentit

Aquest script controla totes aquelles plataformes les quals es pugin travessar per sota però un cop a dalt el personatge col·lisioni i no caigui, a no ser que premi la tecla S o la fletxa inferior per baixar. Aquest comportament és el més usual en aquest estil de jocs. Això permet el moviment vertical d'una manera més fluida i còmode per a l'usuari.



Cofre amb trigger superior

Qualsevol element pot utilitzar aquest script, només fa falta aplicar-li un trigger a la part superior (per exemple els cofres el tenen).

El codi que executa aquesta funcionalitat és bastant simple, l'objecte que es vol utilitzar com a plataforma d'un sentit conté un collider de les mateixes dimensions que el sprite. Aquest collider està desactivat, però en el moment que el personatge col·lisiona amb el trigger superior, el collider de l'objecte s'activa, per tant el personatge no cau.

```
// When the player enter on the top trigger it actives the platform collider
void OnTriggerEnter2D(Collider2D player)
{
    //If the player enter the top trigger and is not falling, we enable the collider of the platform
    if (player.isTrigger == false)
    {
        transform.parent.collider2D.isTrigger = false;
        isInside = true;
    }
}

void OnTriggerExit2D(Collider2D player)
{
    //When we go outside the top trigger we set the platform collider to false
    if (player.isTrigger == false)
    {
        transform.parent.collider2D.isTrigger = true;
    }
    isInside = false;
}
```

Com es pot observar, guardem una variable isInside que marca en quin moment el personatge està col·lisionant amb el trigger superior. Si l'usuari prem la tecla S o fletxa inferior, mentre col·lisiona amb el trigger superior, cau. Això es comprova al Update.

```
void Update()
{
    //Handle going down of the platform pressing S or down
    if(Input.GetKey (KeyCode.S) || Input.GetKey (KeyCode.DownArrow) && isInside == true)
    {
        transform.parent.collider2D.isTrigger = true;
    }
}
```

7.5.10 Controlador de paranys

Els paranys són un dels elements més importants del joc, ja que són les que donen dificultat als nivells. Els paranys van ser creats de manera que es poguessin clonar utilitzant un prefab base. Tenen una animació mitjançant l'editor de Unity en loop, activant i desactivant la trampa, després per fer que funcioni simplement tenen un collider; si el personatge col·lisiona amb la trampa es crida a la funció Die del personatge explicada anteriorment.

```
void OnTriggerEnter2D(Collider2D other)
{
    // We ensure that the collision is with the player collider and no other triggers or objects.
    if (other.tag == "Player"  && other.isTrigger == false)
    {
        characterController.Die();
    }
}
```

7.5.11 Escales

Les escales en un platformer com aquest són vitals, permet moure's verticalment per l'escenari i fa que hi hagi moltes més possibilitats de disseny de nivells i de jugabilitat.

Per realitzar una escala s'ha creat un script que s'afegeix al game object de l'escala, amb un collider. Quan el personatge co·lisiona amb l'escala, una variable isInStair es transforma a true i si marxa de l'abast de l'escala torna a passar a false.

```
void OnTriggerEnter2D(Collider2D other) {
    if (other.tag == "Player"){
        other.GetComponent<CharacterController>().isInStair = true;
        other.GetComponent<CharacterController>().stairPositionX = transform.position.x;
    }
}
void OnTriggerExit2D(Collider2D other) {
    if (other.tag == "Player"){
        other.GetComponent<CharacterController>().isInStair = false;
    }
}
```

Això vol dir que l'usuari està llest per començar a escalar. Un cop prem la lletra W o a la fletxa superior el personatge s'enganxa a l'escala i es canvia la variable 'snapped' a true, en aquest moment ja no es pot moure en l'eix horitzontal, només pot moure's en l'eix vertical. Per sortir de l'estat snapped, l'usuari ha de saltar amb la tecla espai.

```
private void snapToStair(){
    Debug.Log("snap to stair");
    if (snapped == false) {
        move = 0;
        snapped = true;
        transform.position = new Vector2 (stairPositionX, transform.position.y);
        this.gameObject.GetComponent<Rigidbody2D> ().gravityScale = 0;
    }
}

private void leaveStair(){
    Debug.Log("leave stair");
    snapped = false;
    this.gameObject.GetComponent<Rigidbody2D>().gravityScale = 1;
}
```

7.5.12 Utils

Aquest script simplement és un script d'ajuda per als altres scripts, guarda funcions comunes que s'utilitzen en molts llocs del codi, així s'evita repetir codi, per a aquest prototip només conté una funció comuna, la de realitzar un fade out d'un gràfic de color.

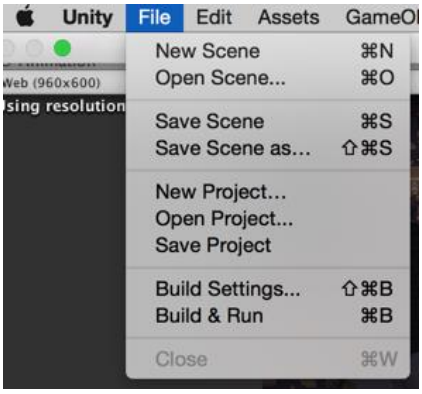
```
public static LTDescr changeColor (Color startColor, Color newColor, GameObject objectToChange, float time = 2f) {
    //objectToChange.GetComponent<Image>().color = Color.Lerp(new Color (0f,0f,0f,0f), new Color (0f,0f,0f,1f), 5f);
    //LeanTween.value (objectToChange, startColor, newColor, 5f);
    return LeanTween.value (objectToChange, (Color updatedColor) => {
        objectToChange.GetComponent<Image>().color = updatedColor;
    }, startColor, newColor, time);
}
```

Com es pot observar és una funció estàtica, el que vol dir que es pot utilitzar des de qualsevol part del codi. Realitza un tween d'un color a un altre durant un temps que es pot passar per paràmetre. Utilitza un Plugin de generació de tweens.

7.6 Build

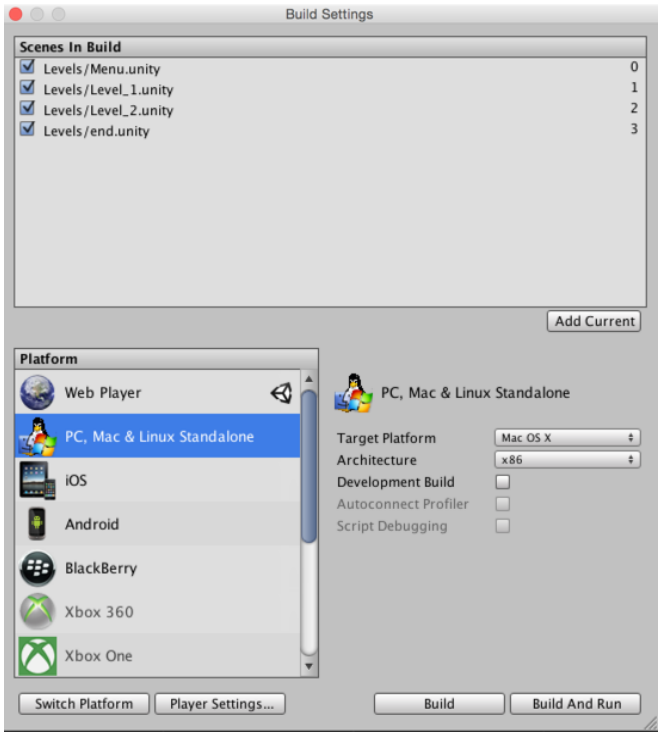
El procés de compilació del projecte es fa directament des de l'editor. En aquest cas per al build del joc es van realitzar builds per a navegador per a realitzar proves ràpides, i es va realitzar un build en forma d'executable per a l'entregable final.

Per a realitzar un build primer de tot cliquem a File -> Build Settings...



Menú File

Seguidament s'ha de preparar la configuració de l'exportable. Primer de tot es seleccionen les escenes que es volen exportar i el seu ordre. Un cop fet això, es selecciona la plataforma a la que volem exportar, en aquest cas serà PC, max & linux Standalone. Seleccionem el target al que va destinat i la seva arquitectura (32 o 64 bits). Finalment quan cliquem a build Unity ens preguntarà on volem guardar l'arxiu i ja s'haurà completat el build.



7.7 Plugins

Els plugins son empaquetats de codi que es poden implementar en un projecte per a realitzar funcionalitats específiques. Un dels avantatges d'utilitzar plugins és que el seu funcionament ha estat provat per molta gent i s'assegura una certa qualitat. Per aconseguir plugins de manera gratuïta i fiable s'ha utilitzat l'asset store de Unity, on es poden descarregar tant plugins com assets o projectes sencers.

S'han utilitzat dos plugins per al projecte:

- Lean Tween
- Texture Packer

7.7.1 Lean Tween

Lean Tween és un plugin de Unity que aporta tota una API per generar Tweens de manera fàcil i ràpida. Es poden fer tweens de gairebé totes les propietats d'un objecte de Unity.

Els tweens es poden realitzar amb uns "Ease" que fará que l'animació sigui més suau, hi han easings de tot tipus

Els tweens també poden contenir una funció que realitza a l'acabar. Aquest tipus de funció s'anomena callback. Els callback permeten realitzar funcionalitats a l'acabar una animació, al repetir l'animació, o inclús al començar-la.

7.7.2 Texture packer

Unity es caracteritza per ser molt bé en un entorn 3D, però encara està començant a obrir-se al mon 2D. Això fa que les seves eines de tractament d'imatges 2D encara siguin bastant rudimentàries.

Per realitzar animacions 2D es necessiten spritesheets o imatges amb tota la sèrie de frames d'una animació i per definir com son aquests frames, quina altura i amplada tenen, quin ordre, etc. Es necessiten una sèrie d'eines que Unity no acabaven de convèncer per a aquest projecte. Per tant es va decidir utilitzar una eina de creació de spritesheets molt habitual en el món HTML5 i web. Per poder utilitzar els spritesheets correctament amb Unity es va utilitzar el plugin oficial d'aquesta eina que han realitzar per a Unity.

8 PROVES D'USUARI

8.1 Introducció

Per millorar el comportament del joc i comprovar que la usabilitat es correcte es va realitzar una prova a un grup d'usuaris tenint el compte el target objectiu (jugadors amb experiència). Si es vol les dades extenses de les proves, es poden trobar a l'apartat d'annexos 'Proves d'usuaris'.

La prova consistia a cronometrar la partida, apuntar el número de morts 'soft'*, el número de morts 'hard'* i en quines escenes es realitzaven. També es comptava el número de relíquies agafades i si l'usuari les buscava. Finalment un cop acabat el joc, es preguntava al usuari quin era l'objectiu i quines eren les mecàniques, si contestava correctament s'apuntava que entenia aquests dos conceptes.

* Les mort soft o hard es diferencien de quan el jugador mor perdent una vida, o perd totes les vides i ha de tornar a començar, respectivament.

8.2 Resultats

Els resultats numèrics extrets va ajudar a arribar a unes conclusions que d'altra manera hauria sigut impossible.

Els resultats van ser molt bons, la mitja de duració de la demo va ser de 8 minuts i 30 segons, que es molt acceptable, els usuaris, tret d'una excepció que es comentarà més endavant, van ser capaços de superar la demo en un temps bastant ràpid, no es van encallar durant hores.

Un altre factor important es que quan preguntàvem sobre quin era l'objectiu del joc, el 90% van contestar correctament, i l'altre 10% no ho sabia perquè havia passat la introducció.

Un 70% dels jugadors van aprendre com funcionaven totes les mecàniques ràpidament, l'altre 30% fallava sobretot amb la mecànica de baixar plataformes i al intentar sortir de l'escala. Aquest resultat va ser molt útil per que es va veure una falta de tutorial en aquestes dues mecàniques i es va corregir.

El 40% dels usuaris buscaven les relíquies amagades per el mapa, al no haver un boss final a la demo, i al no estar implementat el registre de relíquies trobades, realment els usuaris no veien la necessitat d'agafar relíquies, simplement agafaven la més fàcil i prou. Possiblement es podria implementar més recompenses per agafar relíquies, com power ups, o un sistema de logros, així s'evitaria el desinterès per aquesta funcionalitat.

La mitjana de relíquies aconseguides es d'1.7, això no vol dir que no les agafessin totes abans, sinó que com morien, ja no tornaven a agafar-les.

Amb el recompte de morts van sortir números molt interessants. Tot i que el número mig de morts totals no era gaire alt per a la dificultat que es buscava en aquest videojoc, 12.6, la quantitat de morts estaven acumulades a una sola escena, i aquesta era la segona. Això suposa un problema, ja que no pot ser que la corba de dificultat incrementi tant a la segona escena, i després les altres escenes es tornin fàcils. Per tant com a solució no seria rebaixar la dificultat general del joc, sinó canviar d'ordre les sales, per tant de que la corba fos més progressiva.

Es va realitzar una prova a una noia que no entrava al target d'usuaris, i es va veure com va ser incapaç de passar de la segona escena. Al no ser el target objectiu podria semblar irrelevant, baixar la dificultat del joc no es la solució per que llavors tota la filosofia de joc com a repte per a usuaris experimentats es perdria, però si que seria bona idea implementar una selecció de dificultat on l'usuari pogués triar un joc més fàcil, amb més vides, més lent, menys trampes, etc.

9 Usabilitat

A nivell d'usabilitat es van estudiar dos parts fundamentals. La primera part va ser el menú i la interfície d'usuari, i la segona part van ser els controls.

9.1 Menú / UI

El menú consisteix en quatre botons, dos d'ells desactivats ja que encara no es poden utilitzar. Els altre són el botó de començar partida i el de crèdits.

La UI va començar amb botons separats, petits, situats just al centre de la pantalla. Per millorar la seva visibilitat es van agrupar i es van situar davant d'un fons especial per al grup de botons, millorant també el focus d'atenció de l'usuari cap a aquesta zona de la pantalla.

9.2 Controls

Els controls es la part més important del projecte, es volia un control molt fluït del personatge, i si els controls fallen, es perdria tot el feeling del joc. Per aconseguir això es va realitzar no només un control, sinó que l'usuari pot jugar amb dos controls, amb les fletxes o amb les lletres 'wasd', d'aquesta manera sigui quin sigui el tipus d'usuari tots juguen de la manera més còmode.

A més a més el salt es pot realitzar clicant a la lletra W o la fletxa superior, o clicant la barra d'espai. D'aquesta manera s'aconsegueix que qualsevol tipus d'usuari, el que esta acostumat al platformer comú on es salta amb la tecla superior o un usuari que no està tant acostumat puguin jugar sense cap problema.

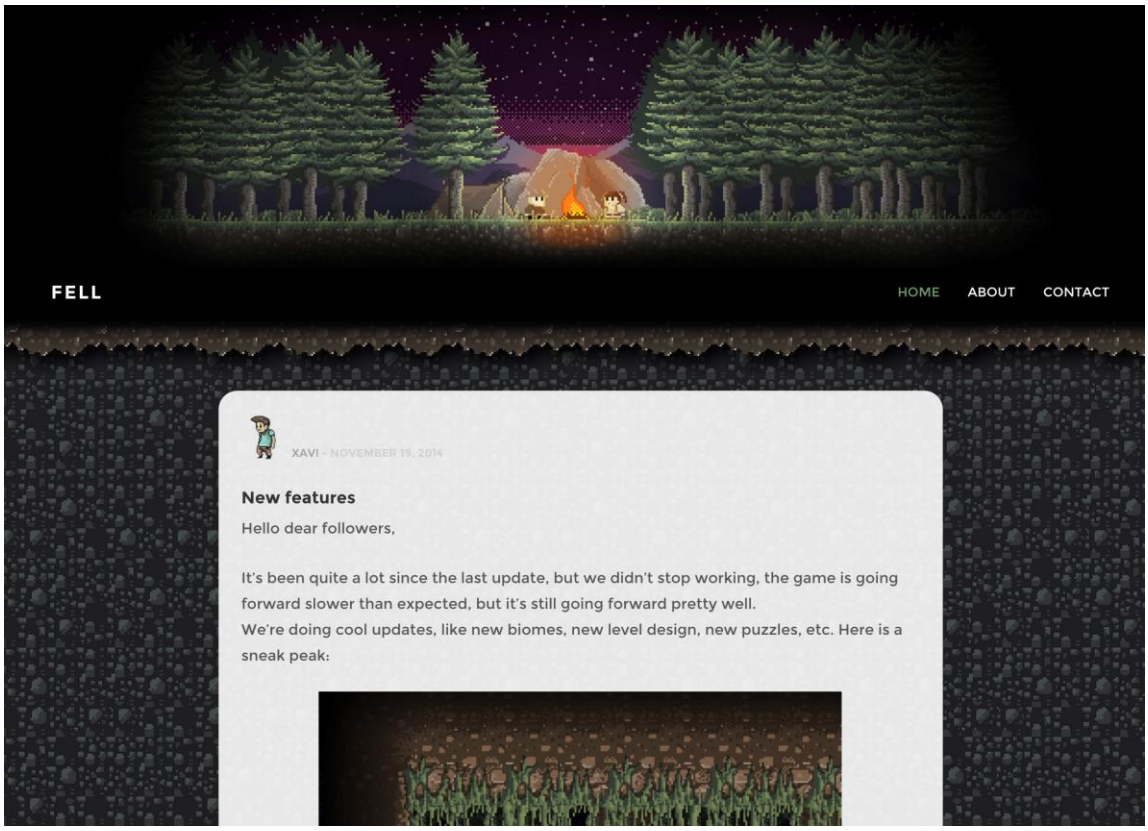
Per que l'usuari es senti còmode i aprengui a utilitzar els controls, el primer nivell serveix de tutorial, i amb uns pop-ups, s'ensenya al jugador com realitzar les accions.

10 MARKETING

10.1 Pàgina web

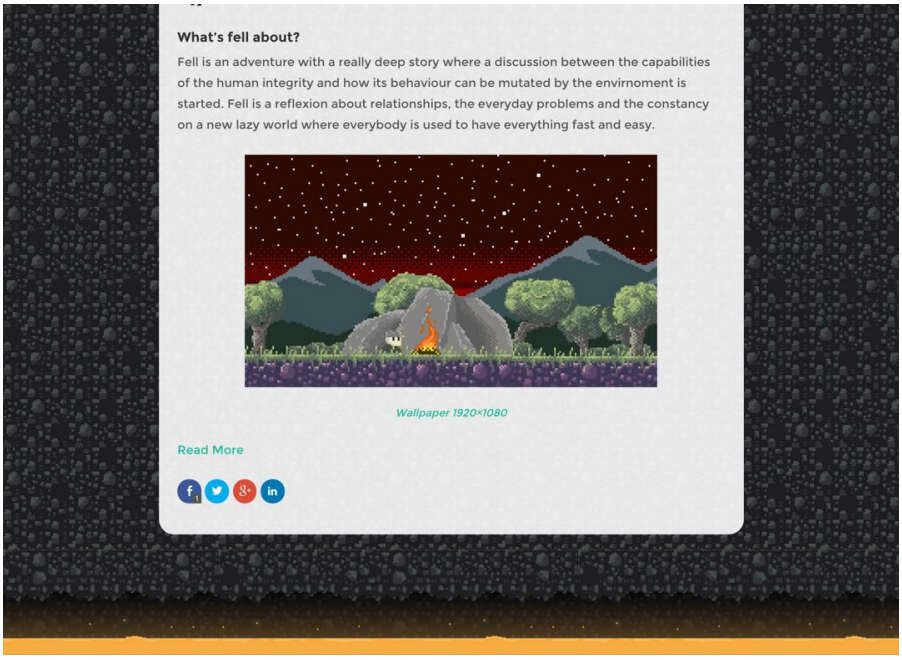
Per a aquest projecte es va crear una pàgina web en format blog, que contindria un roadmap del que s'havia de fer i una actualització quan s'afegissin noves features. Així s'aconseguiria una mica de visibilitat i de "hype" per a quan es tragués el joc al carrer.

Per a realitzar la pàgina web es va utilitzar un wordpress. El disseny de la pàgina es va fer customitzat especialment per al joc. Tota la pàgina sembla un escenari del joc i això crea una sensació d'immersió molt gran i millora l'experiència d'usuari.



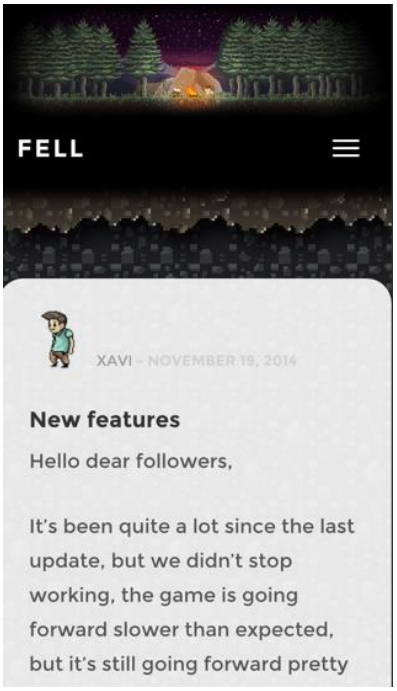
Header

La web conté un scroll infinit; quan s'arriba a l'últim post del blog apareix el fons de la cova, amb magma, per donar un toc de sorpresa i encara més immersió.



Footer

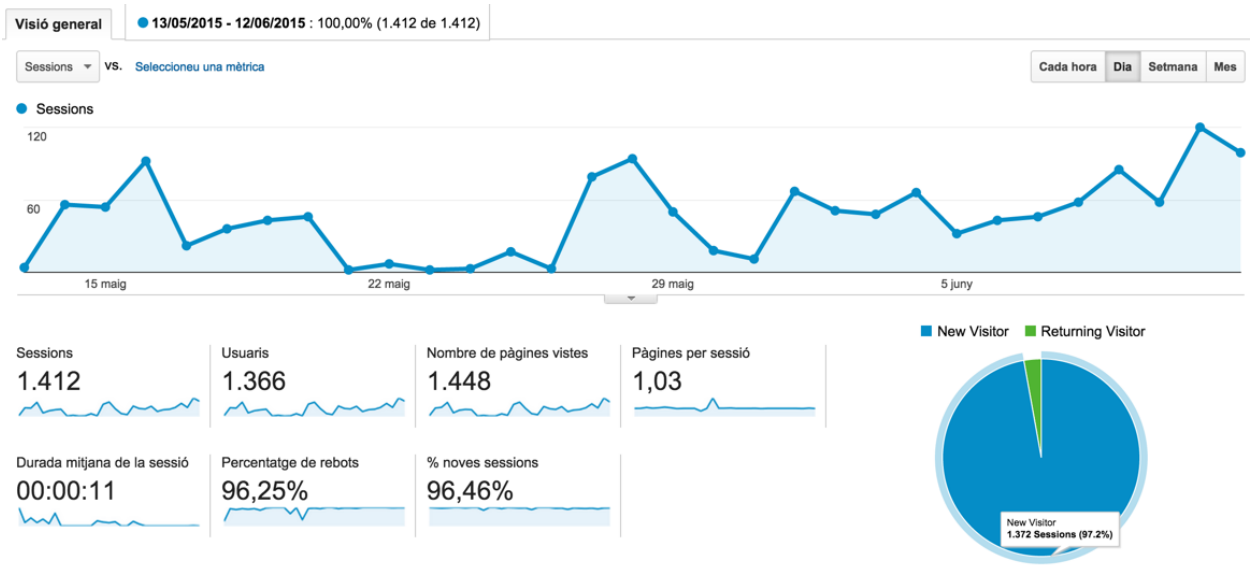
La web es completament responsive ja que la majoria del tràfic, al vindre de twitter, entrava des del mòbil i era necessari fer que l'experiència d'usuari fos òptima.



El menú es col·lapsa en un botó que fa aparèixer el menú i totes les imatges s'escalen per adaptar-se a la resolució necessària.

10.2 Analytics

Per saber quin impacte tenia la pàgina web i actuar en conseqüència es va aplicar google analytics per tenir una visió de les visites, visió a temps real, nacionalitat i idioma dels usuaris, etc.



Dades analytics

Com es pot observar a la gràfica hi ha una entrada constant d'usuaris, on el 97% son nous. Aquestes visites s'han aconseguit abans de fer cap campanya de marketing, ja que el joc no està acabat i està en fase de prototipat. Només s'ha linkejat la web al twitter de l'artista i del desenvolupador. Es pot observar la potència que pot tenir una web com a element distribuïdor.

Per aplicar analytics a la web simplement s'ha d'afegir un script al document html amb un codi que proporciona Google.

<script> "Codi de google" </script>

11 PRESSUPOST

A continuació s'explicarà com s'ha realitzat el pressupost d'aquest projecte, igualment, a l'apartat d'annexos hi ha la taula amb tots els costos de cada concepte.

Per preparar el pressupost es va tenir en compte la realització d'un prototip presentable per a una demo, que mostrés de manera ràpida les mecàniques del joc, dos nivells diferents i un mínim de guió (intro).

Per calcular el pressupost del projecte es va comptar amb dues persones, un dissenyador i un programador, a 25 euros bruts l'hora, a més a més es van afegir costos d'instal·lacions i equipament. S'ha dividit el pressupost del projecte en cinc parts, preproducció, producció – tècnica, producció – artística, postproducció i instal·lacions.

A la part de preproducció s'ha afegit tot l'estudi de mercat i tecnologies, game design i level design. A la part de producció tant tècnica com artística s'ha afegit tot el desenvolupament del projecte. A la postproducció s'ha tingut en compte la creació i manteniment del blog. Per últim també s'ha comptat amb el cost de llum, instal·lacions, etc.

Els costos totals de cada apartat (en brut) són:

- Preproducció: 1450€
- Producció tècnica: 4275€
- Producció artística: 3000€
- PostProducció: 600€
- Instal·lacions / manteniment: 4710€

Per tant, el cost total és de 14035€.

12 CONCLUSIONS

12.1 General

Realitzar aquest projecte m'ha ajudat a entendre, de manera global, què significa realitzar un videojoc de grans dimensions, m'ha ajudat a veure les coses bones i les dolentes. Per una banda veure com avança el progrés del videojoc i el producte que acabes tenint és molt satisfactori, però eliminar objectius i idees per a que el projecte acabi sent realista i possible és complicat. Per a aquest projecte s'han intentat complir tots els objectius, tants els principals com els secundaris o personals, i amb certa mesura s'han complert, però també es cert que no s'ha pogut realitzar un prototip tant detallat com es volia, Al començament s'apuntava tant alt que es volien realitzar tres nivells, un motor de generació de nivells aleatoris, diferents enemics, puzles. Era un error pensar que una persona sola es capaç d'aconseguir tots aquests objectius en un curt període de temps, a més una persona amb un nivell de programació gens avançat.

Tot i no assolir tots els objectius del prototip, el prototip final ha sigut molt satisfactori, la web ha tingut moltes visites i la sensació general dels usuaris al jugar és molt bona. És cert que s'ha centrat molt al desenvolupament de detalls que fan la experiència del usuari molt més agradable.

12.2 Tecnologies

Durant el desenvolupament del joc el mercat a variat molt i les tecnologies també. Unity ha sigut una bona eina per a realitzar el joc, però el coneixement d'HTML5 i la seva progressió amb els nous frameworks que han aparegut l'ha fet per a mi la tecnologia més òptima per a la realització de jocs petits i mitjans.

12.3 Mercat

El mercat el darrer any ha canviat moltíssim, l'aparició de jocs, o directament es poden anomenar "minijocs", ha fet que es pugin llençar al mercat jocs en setmanes, que poden arribar a generar una quantitat de diners igual o millor a la d'un joc molt més gran.

La grandària dels jocs no vol dir que afecti a la qualitat d'aquests. Hi han minijocs molt treballats, amb una gràfica que arriba a l'art i mecàniques tant fluides que fan el joc molt adictiu.

Aquest nou estil de joc són completament gratuïts, normalment d'estil freemium on es poden comprar millores, però sobretot d'on treuen benefici és dels anuncis, ja que apareixen de tant en tant quan s'acaba la partida.

Si ara comencés el TFG segurament realitzaria un joc d'aquest estil, ja que es demana molt no només per a game developers independents sinó que hi han estudis que es dediquen completament a aquest mercat i contracten gent amb experiència al sector.

13 BIBLIOGRAFIA

Unity vs HTML5

<http://www.develop-online.net/news/develop-live-unity-vs-html5/0198301>

(consultat 5 febrer de 2015)

Pros and cons of HTML5 gaming

< http://www.rivellomultimediaconsulting.com/pros-and-cons-of-html5-for-gaming/ >
(consultat 5 febrer de 2015)

Mobile game development html5 vs Unity

< http://intersog.com/blog/mobile-game-development-unity-vs-html5/ >

(consultat 10 febrer de 2015)

General tips to work with Unity

<http://www.gamasutra.com/blogs/JohnWarner/20130910/194559/The_top_5_things_Ive_learned_as_a_Unity_developer.php >

(consultat 10 febrer de 2015)

Mobile sales numbers

< https://thinkgaming.com/app-sales-data/ >

(consultat 12 febrer de 2015)

Learning to code in Unity

< http://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/coding-for-the-absolute-beginner >

(consultat 5 marc de 2015)

Unity 4.3 - 2D Game Development Walkthrough

< https://www.youtube.com/watch?v=4qE8cuHl93c >

(consultat 10 març de 2015)

Live Training 16 Dec 2013 - 2D Character Controllers

< https://www.youtube.com/watch?v=Xnyb2f6Qqzg >

(consultat 14 març de 2015)

Unity 2D getting Started

<http://www.raywenderlich.com/61532/unity-2d-tutorial-getting-started>

(consultat 14 març de 2015)

Change animation speed in mecanim

<http://forum.unity3d.com/threads/mecanim-change-animation-speed-of-specific-animation-or-layers.160395/>

(consultat 20 març de 2015)

Roguelike 2D game development tutorial

< http://unity3d.com/learn/tutorials/projects/2d-roguelike >

(consultat 1 maig de 2015)

2D catch game tutorial

< http://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/2d-catch-game-pt1 >

(consultat 10 maig de 2015)

ANEXOS

Anex de codi

Camera controller

```
using UnityEngine;
using System.Collections;

public class CameraController : MonoBehaviour
{
   private float limitX;
   private float limitY;
   public float offsetY = 1.25f;
   private bool canMoveX, canMoveY, block;
   public Transform character;

   void Awake()
   {
      offsetY = 1.25f;
      canMoveY = true;
      canMoveX = true;
      block = false;
   }

   void Update ()
   {
      if (!block) {
         if(canMoveX == true && canMoveY == true)transform.position = new Vector3
(character.position.x, character.position.y + offsetY, transform.position.z);
         if(canMoveX == true && canMoveY == false)transform.position = new Vector3
(character.position.x, transform.position.y, transform.position.z);
         if(canMoveX == false && canMoveY == true)transform.position = new Vector3
(transform.position.x, character.position.y + offsetY, transform.position.z);
      }
   }

   void OnTriggerStay2D (Collider2D other)
   {
      if (other.tag == "CameraLeft" && transform.position.x > other.transform.position.x)
      {
         canMoveX = false;
         if (character.position.x >= transform.position.x){
            canMoveX = true;
         }
      }
      if (other.tag == "CameraRight" && transform.position.x < other.transform.position.x)
      {
         canMoveX = false;
         if (character.position.x <= transform.position.x){
```

```
                canMoveX = true;
            }
        }
        if (other.tag == "CameraTop")
        {
            canMoveY = false;
            if (character.position.y <= transform.position.y - offsetY){
                canMoveY = true;
            }
        }
        if (other.tag == "CameraBottom")
        {
            canMoveY = false;
            if (character.position.y >= transform.position.y + offsetY){
                canMoveY = true;
            }
        }
    }

    public void setBlock(bool status) {
        block = status;
    }
}
```

Change scene controller

```
using UnityEngine;
using System.Collections;

public class ChangeSceneController : MonoBehaviour {

   public Transform character;
   public Transform camera;
   public Transform target;
   public Transform targetCamera;
   public float nextOffsetY;
   public float nextScale = 2.05f;

   void OnTriggerEnter2D (Collider2D other) {
      if(other.tag == "Player") Change ();
   }

   private void Change() {
      camera.GetComponent<CameraController> ().setBlock (true);
      character.position = new Vector3 (target.position.x, character.position.y,
character.position.z);
      camera.position = new Vector3 (targetCamera.position.x, camera.position.y,
camera.position.z);
      camera.GetComponent<CameraController> ().setBlock (false);
      camera.GetComponent<CameraController> ().offsetY = nextOffsetY;
      camera.GetComponent<Camera> ().orthographicSize = nextScale;
   }

}
```

Character controller

```
using UnityEngine;
using System.Collections;

public class ChangeSceneController : MonoBehaviour {

   public Transform character;
   public Transform camera;
   public Transform target;
   public Transform targetCamera;
   public float nextOffsetY;
   public float nextScale = 2.05f;

   void OnTriggerEnter2D (Collider2D other) {
      if(other.tag == "Player") Change ();
   }

   private void Change() {
      camera.GetComponent<CameraController> ().setBlock (true);
      character.position = new Vector3 (target.position.x, character.position.y,
character.position.z);
      camera.position = new Vector3 (targetCamera.position.x, camera.position.y,
camera.position.z);
      camera.GetComponent<CameraController> ().setBlock (false);
      camera.GetComponent<CameraController> ().offsetY = nextOffsetY;
      camera.GetComponent<Camera> ().orthographicSize = nextScale;
   }

}
```

Demo end

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;


public class GameController : MonoBehaviour {
   public GameObject blackBg;
   public Transform character;
   public Transform endPoint;
   private bool fadeIn = false;
   private Color alpha0 = new Color (0f,0f,0f,0f);
   private Color alpha100 = new Color (0f,0f,0f,1f);
   private float dist;
   // Use this for initialization
   void Start () {
      Utils.changeColor (alpha100, alpha0, blackBg, 3f);
   }

   // Update is called once per frame
   void Update () {
      dist = character.position.x - endPoint.position.x;
      if(dist >= -1) {
         Debug.Log (1+dist);
         blackBg.GetComponent<Image>().color = Color.Lerp(alpha0, alpha100, 1+dist);
      }
   }


   void changeLevel() {
      Application.LoadLevel (2);
   }


   public void nextLevel() {
      //Utils.changeColor (alpha0, alpha100, blackBg).setOnComplete(changeLevel);
      changeLevel ();


   }
```

Menu controller

```
using UnityEngine;
using System.Collections;

public class LevelController : MonoBehaviour {

   public GameController gameController;

   void OnTriggerEnter2D(Collider2D other)
   {
      if (other.tag == "Player")
      {
         gameController.nextLevel();
      }
   }
}
```

menu controller

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class MenuController : MonoBehaviour {
   private Color white0 = new Color (255f,255f,255f,0f);
   private Color white100 = new Color (255f,255f,255f,1f);
   private Color black0 = new Color (0f,0f,0f,0f);
   private Color black100 = new Color (0f,0f,0f,1f);
   public GameObject blackBg;
   public Button newGameBtn;
   public Button continueBtn;
   public Button creditsBtn;
   public Button soundBtn;

   public Text text1;
   public Text text2;

   public void startGame () {
      hideCredits ();
      blackBg.GetComponent<Image>().enabled = true;
      Utils.changeColor (black0, black100, blackBg, 3f).setOnComplete(() => {
         Application.LoadLevel (1);
      });
   }

   public void showCredits () {
      LeanTween.value (text1.gameObject, (Color updatedColor) => {
         text1.GetComponent<Text>().color = updatedColor;
```

```
        }, white0, white100, 0.7f).setOnComplete (() => {
            LeanTween.value (text2.gameObject, (Color updatedColor) => {
                text2.GetComponent<Text>().color = updatedColor;
            }, white0, white100, 0.7f);
        });
        LeanTween.moveLocalY (text1.gameObject, -210f,
1f).setEase(LeanTweenType.easeInOutCirc).setOnComplete(() => {
            LeanTween.moveLocalY (text2.gameObject, -210f,
1f).setEase(LeanTweenType.easeInOutCirc);
        });
    }

    public void hideCredits () {
        LeanTween.value (text1.gameObject, (Color updatedColor) => {
            text1.GetComponent<Text>().color = updatedColor;
        }, white100, white0, 0.7f).setOnComplete (() => {
            LeanTween.value (text2.gameObject, (Color updatedColor) => {
                text2.GetComponent<Text>().color = updatedColor;
            }, white100, white0, 0.7f);
        });
        LeanTween.moveLocalY (text1.gameObject, -210f,
1f).setEase(LeanTweenType.easeInOutCirc).setOnComplete(() => {
            LeanTween.moveLocalY (text2.gameObject, -210f,
1f).setEase(LeanTweenType.easeInOutCirc);
        });
    }
}
```

Move sky

```
using UnityEngine;
using System.Collections;

public class MoveSky : MonoBehaviour {

  // Use this for initialization
  void Start () {
    LeanTween.rotateZ (this.gameObject, 180f, 80f).setLoopClamp();
  }

  // Update is called once per frame
  void Update () {

  }
}
```

One way platform

```
using UnityEngine;
using System.Collections;

public class OneWayPlatform : MonoBehaviour
{
  private bool isInside;          //Boolean to know if it's inside the trigger or not, just in case

  // Use this for initialization
  void Start ()
  {
    isInside = false;
    transform.parent.GetComponent<Collider2D>().isTrigger = true; // Set the collider of the
platform to false
  }

  void Update()
  {
    //Handle going down of the platform pressing S or down
    if(Input.GetKey (KeyCode.S) || Input.GetKey (KeyCode.DownArrow) && isInside == true)
    {
      transform.parent.GetComponent<Collider2D>().isTrigger = true;
    }
  }

  // When the player enter on the top trigger it actives the platform collider
  void OnTriggerEnter2D(Collider2D player)
  {
    //If the player enter the top trigger and is not falling, we enable the collider of the platform
    if (player.isTrigger == false)
```

```
        {
            transform.parent.GetComponent<Collider2D>().isTrigger = false;
            isInside = true;
        }
    }

    void OnTriggerExit2D(Collider2D player)
    {
        //When we go outside the top trigger we set the platform collider to false
        if (player.isTrigger == false)
        {
            transform.parent.GetComponent<Collider2D>().isTrigger = true;
        }
        isInside = false;
    }

}
```

Stair controller

```
using UnityEngine;
using System.Collections;

public class StairController : MonoBehaviour {

   public GameObject character;

   void OnTriggerEnter2D(Collider2D other) {
      if (other.tag == "Player"){
         other.GetComponent<CharacterController>().isInStair = true;
         other.GetComponent<CharacterController>().stairPositionX = transform.position.x;
      }
   }
   void OnTriggerExit2D(Collider2D other) {
      if (other.tag == "Player"){
         other.GetComponent<CharacterController>().isInStair = false;
      }
   }
}
```

Switch controller

```
using UnityEngine;
using System.Collections;

public class SwitchController : MonoBehaviour
{

   private bool nearChest = false;
   private bool isOpen = false;
   private bool isMoving = false;
   private float travelTime;
   public GameObject character;
   public GameObject platform;
   public Transform platformTargetUp;
   public Transform platformTargetDown;
   private bool goingUp = true;
   private Animator animSwitch;
   private float distanceToTravel;
   public Animator animPlatform1;
   public Animator animPlatform2;
   //Information about open chest
   public GameObject infoAction;                  //Link to the information panel GameObject
   private SpriteRenderer infoActionRenderer;     //SpriteRenderer of the panel
   private Color infoActionColor;                 //


   void Awake ()
   {
      travelTime = 2f;
      animSwitch = GetComponent<Animator>();
      infoActionRenderer = infoAction.GetComponent<SpriteRenderer>();
      SetAlpha (0f);
   }

   void Update ()
   {
      if (nearChest && Input.GetKeyDown (KeyCode.E))
      {
         SwitchPlatform();
      }
   }

   void OnTriggerEnter2D(Collider2D other)
   {
      if (other.tag == "NearTrigger")
      {
         nearChest = true;
         SetAlpha (1f);
      }
```

```
    }

    void OnTriggerExit2D(Collider2D other)
    {
        if (other.tag == "NearTrigger")
        {
            nearChest = false;
            SetAlpha (0f);
        }
    }

    public void SwitchPlatform () {
        if(!isMoving) {
            animSwitch.SetTrigger ("goActive");
            animPlatform1.SetBool("Active", true);
            animPlatform2.SetBool("Active", true);
            isMoving = true;
            MovePlatform();
        } else {
            animSwitch.SetTrigger ("goOff");
            animPlatform1.SetBool("Active", false);
            animPlatform2.SetBool("Active", false);
            isMoving = false;
            LeanTween.cancel (platform);
        }
    }
    public void SwitchDirection () {
        goingUp = !goingUp;
        MovePlatform ();
    }

    public void MovePlatform() {
        distanceToTravel = Vector3.Distance(platform.transform.position,
platformTargetUp.position);
        if(goingUp) {
            distanceToTravel = Vector3.Distance(platform.transform.position,
platformTargetUp.position);
            LeanTween.move (platform, platformTargetUp.position,
travelTime*distanceToTravel).setEase( LeanTweenType.linear ).setOnComplete(
SwitchDirection );
        } else {
            distanceToTravel = Vector3.Distance(platform.transform.position,
platformTargetDown.position);
            LeanTween.move (platform, platformTargetDown.position,
travelTime*distanceToTravel).setEase( LeanTweenType.linear ).setOnComplete(
SwitchDirection );
        }
    }

    void SetAlpha (float value)
    {
```

```
        infoActionColor = infoActionRenderer.color;
        infoActionColor.a = value;
        infoActionRenderer.color =  infoActionColor;
    }
}
```

Trap controller

```
using UnityEngine;
using System.Collections;

public class TrapController : MonoBehaviour
{

   public CharacterController characterController; //We call the script characterController

   // Use this for initialization
   void Start () {
   }

   // Update is called once per frame
   void Update () {

   }


   void OnTriggerEnter2D(Collider2D other)
   {
      // We ensure that the collision is with the player collider and no other triggers or objects.
      if (other.tag == "Player"  && other.isTrigger == false)
      {
         characterController.Die();
      }
   }
}
```

Utils

```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine.UI;

public class Utils {
   public static LTDescr changeColor (Color startColor, Color newColor, GameObject
objectToChange, float time = 2f) {
      //objectToChange.GetComponent<Image>().color = Color.Lerp(new Color (0f,0f,0f,0f), new
Color (0f,0f,0f,1f), 5f);
      //LeanTween.value (objectToChange, startColor, newColor, 5f);
      return LeanTween.value (objectToChange, (Color updatedColor) => {
         objectToChange.GetComponent<Image>().color = updatedColor;
      }, startColor, newColor, time);
   }
}
```

Anex de pressupost

Pressupost el·laboració del prototip d'un videojoc			
Preproducció			
Conceptes	Hores tècnic	Cost / h brut	Total
Estudi de mercat	12	25	300
Estudi de tecnologies	12	25	300
Creació del guió	4	25	100
Creació del Document de disseny - Personatges	8	25	200
Creació del Document de disseny - Nivell 1	4	25	100
Creació del Document de disseny - Nivell 2	4	25	100
Creació del Document de disseny - Nivell 3	4	25	100
Creació del Document de disseny - Jugabilitat	10	25	250
Producció - Tècnica			
Conceptes	Hores tècnic	Cost / h brut	Total
Preparació entorn i creació projecte	2	25	50
Implementació sistema git	4	25	100
Generació entorn de proves	4	25	100
Càrrega d'assets i creació de spritesheets amb texture packer	8	25	200
Creació d'obstacles i col·lisions	10	25	250
Creació del sistema de cofres	2	25	50
Implementació inputs d'usuari (moviment, salt i interacció amb l'entorn)	8	25	200
Animacions de personatge	10	25	250
Creació del primer nivell	12	25	300
Creació plataforma elevadora	4	25	100
Creació plataformes d'un sentit	4	25	100
Creació de trampes	6	25	150
Implementació plugin de tweens al projecte	2	25	50
Creació escales	8	25	200
Creació del segon nivell	20	25	500
Creació de transicions entre nivells	6	25	150

Creació de transicions entre escenes	12	25	300
Personalització moviment de la càmera	4	25	100
Creació sistema de vides	3	25	75
Creació sistema de reliquies	3	25	75
Creació del menú	8	25	200
Creació de la introducció al joc	4	25	100
Creació sistema d'events amb moviment de càmera i efectes	4	25	100
Millorar sistema de moviment, salt i col·lisions	10	25	250
Implementació salt de la introducció	2	25	50
Implementació animació de textos amb efecte de màquina d'escriure	6	25	150
Implementació UI	3	25	75
Exportació	2	25	50
Producció - Artística			
Conceptes	Hores tècnic	Cost / h brut	Total
Estudi de disseny	4	25	100
Concept art	12	25	300
Disseny de personatges	10	25	250
Disseny d'entorn	8	25	200
Disseny del HUD	8	25	200
Animacions de personatges	12	25	300
Disseny d'atrezzo	12	25	300
Animacions d'atrezzo	6	25	150
Disseny del nivell 1	16	25	400
Disseny del nivell 2	24	25	600
Implementació de gràfica de nivell 1	4	25	100
Implementació de gràfica de nivell 2	4	25	100
Postproducció			
Conceptes	Hores tècnic	Cost / h brut	Total
Creació del blog	8	25	200
Disseny del blog	12	25	300
Creació de posts i notícies	4	25	100

Instal·lacions / Manteniment			
Conceptes	Cost	Mesos	Total
Alquiler estudi 40m2 durant 6 mesos (Barcelona)	700	6	4200
Llum	60	6	360
Aigua	25	6	150
			Total cost
			14035

Proves d'usuari

Proves amb usuaris				
Usuari	Juan Carlos	Vicente	Carles Núñez	Albert Bailo
Edat	30	26	24	16
Sexe	Home	Home	Home	Home
Temps de finalització	4m 08s	12m 45s	6m	5m 03s
Entén l'objectiu?	Si	Si	Si	si
Entén totes les mecàniques?	No/baixar de plataformes	Si	No/baixar de plataformes	si
Busca les reliquies?	Si	No	No	si
Nº reliquies conseguides	3	1	1	3
Nº soft deaths escena 1	0	0	0	0
Nº hard deaths escena 1	0	0	0	0
Nº soft deaths escena 2	3	8	8	8
Nº hard deaths escena 2	0	2	2	2
Nº soft deaths escena 3	3	7	3	8
Nº hard deaths escena 3	1	3	1	2
Nº soft deaths escena 5	0	3	0	1
Nº hard deaths escena 5	0	1	0	0
Nº soft deaths TOTAL	6	18	11	17
Nº hard deaths TOTAL	1	6	3	5

Proves amb usuaris				
Usuari	Antonio Gonzalez	Carles Prieto	Lorena Mota	Javier Alcantara
Edat	17	17	23	25
Sexe	Home	Home	Dona	Home
Temps de finalització	8m 27s	8m 27s	--	10m 16s
Entén l'objectiu?	Si	Si	No	Si
Entén totes les mecàniques?	si	si	No/Escales	Si
Busca les reliquies?	No	No	No	Si
Nº reliquies conseguides	0	0	1	3
Nº soft deaths escena 1	2	2	5	1

Nº hard deaths escena 1	0	0	1	0
Nº soft deaths escena 2	13	13	30 +	5
Nº hard deaths escena 2	3	3	10 +	1
Nº soft deaths escena 3	2	2	-	0
Nº hard deaths escena 3	0	0	-	0
Nº soft deaths escena 5	0	0	-	1
Nº hard deaths escena 5	0	0	-	0
Nº soft deaths TOTAL	17	17	30 +	9
Nº hard deaths TOTAL	5	5	10 +	2
Proves amb usuaris			Mitja	
Usuari	Sergi Leon	Albert Oriol		
Edat	24	23		
Sexe	Home	Home		
Temps de finalització	16m 14s	8m 12s	8m 30s	
Entén l'objectiu?	Si	Si	90%	
Entén totes les mecàniques?	Si	Si	70%	
Busca les reliquies?	No	Si	40%	
Nº reliquies conseguides	2	3	1,7	
Nº soft deaths escena 1	0	0	1	
Nº hard deaths escena 1	0	0	0,1	
Nº soft deaths escena 2	9	7	8,2	
Nº hard deaths escena 2	3	2	2	
Nº soft deaths escena 3	1	1	3	
Nº hard deaths escena 3	0	0	0.7	
Nº soft deaths escena 5	1	0	0.6	
Nº hard deaths escena 5	0	0	0,1	
Nº soft deaths TOTAL	11	8	12,6	
Nº hard deaths TOTAL	3	2	3,5	