Universitat de Lleida

TREBALL FINAL DE MÀSTER

ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Adrià Bonet Vidal

Titulació: Màster en Enginyeria Informàtica

Títol de Treball Final de Màster: Desenvolupament d'un videojoc en tres dimensions amb Unity

Director/a: Francesc Sebé Feixas

Presentació

Mes: Octubre

Any: 2020

Resumen

En aquest projecte s'ha desenvolupat un videojoc en tres dimensions utilitzant el software de Unity. S'ha volgut que sigui original i que qui hi jugui pugui passar una bona estona.

El joc tracta d'aconseguir pujar una torre, on cada pis és una pantalla a superar. A cada pantalla el jugador lluitarà contra enemics en un sistema de batalla per torns i anirà millorant per poder avançar en la torre. A mesura que el jugador va pujant pisos la dificultat augmentarà. La idea és que no hi hagi un últim pis, de manera que es pugui anar afegint contingut nou periòdicament.

En aquest document s'expliquen totes les metodologies utilitzades, les decisions preses i el procediment realitzat, com s'han aplicat els coneixements adquirits durant el Màster, juntament amb els problemes que ha calgut resoldre i les conclusions.

Agraïments

Per començar, m'agradaria agrair a tots els que m'han ajudat a realitzar el projecte, ja sigui resolent dubtes, estant allí donant suport, o be ajudant-me directament.

Moltes gracies al meu tutor Francesc Sebé Feixas per agafar el meu projecte i ajudar-me a tirar endavant.

També, als meus companys Marc Ribalta, Adrián Aguilar, Sergio Salcedo i Ricard Cervera per proporcionar idees i donar suport. A Pedro Calero per ajudar-me en dubtes de Unity i C#.

Finalment, agraïments també a tota la familia per tot el suport que s'ha tingut.

Índex

Índex de taules

Índex de figures

1 Introducció

1.1 Context del projecte

Per acabar el Màster en Enginyeria Informàtica és necessari realitzar un projecte final per demostrar la consolidació dels coneixements adquirits durant els estudis. Dins l'empresa on treballava no havia cap projecte suficientment gran per poder tractar-ho com a treball de final de Màster. Finalment, es va decidir fer un joc en 3 dimensions amb Unity, ja que és una temàtica entretinguda i agradable de fer i és una manera de poder aplicar alguns dels coneixements adquirits, concretament la part d'intel·ligència artificial.

1.2 Objectius del projecte

Els objectius generals pel projecte han estat crear un joc en 3 dimensions que sigui entretingut i que reflectís el coneixement adquirit en el Màster.

Un cop acabat tot el desenvolupament també es tenia com a objectiu intentar fer la portabilitat del projecte a una consola de Nintendo (la idea inicial era a una Nintendo3DS o Nintendo2DS). Per desgracia aquesta portabilitat no s'ha pogut realitzar, ja que era necessari obtenir un software específic de Nintendo que té un cost periòdic.

1.3 Estructura del document

En aquest document s'explica el procediment seguit per fer el joc. Es comença per les decisions de disseny i la tria d'eines, per què s'ha decidit fer un joc d'aquest tipus, com s'han organitzat les tasques, etc, i s'explicarà també el funcionament del joc i els aspectes més rellevants.

2 Detalls previs al desenvolupament

2.1 Eines

La primera decisió realitzada ha estat pensar quina eina principal s'utilitzarà per desenvolupar el joc. Com que ja es tenia experiència prèvia respecte el desenvolupament de jocs, s'ha decidit utilitzar Unity ja que proporciona una interfície molt amigable i una facilitat molt alta en gairebé tots els aspectes. A més, permet importar components externs que apliquen noves funcionalitats i també permet descarregar recursos pel joc (models, etc). A l'hora de fer els scripts accepta diferents llenguatges de programació, entre ells C# que és molt sencill i serà el que s'utilitzarà principalment (els altres dos són JavaScript i Boo).

Figura 1: Logotip de Unity

Pel al modelatge i disseny, s'ha utilitzat Blender i Photoshop. De Photoshop es té una experiència previa molt alta i serà molt útil per dissenys de la interfície d'usuari. Blender s'ha escollit ja que és un software de modelatge 3D molt popular, i a falta de coneixement d'altres s'ha considerat la millor opció.

Per programar s'ha utilitzat Visual Studio, ja que és un dels IDE per defecte de Unity i és molt còmode d'utilitzar. L'altra opció era MonoDevelop, però s'ha considerat que Visual Studio és força millor.

2.1.1 Control de versions

Respecte el tema del control de versions del projecte, s'ha intentat utilitzar diversos d'ells.

La primera opció considerada ha estat Github, però com que el projecte tracta d'un joc en 3 dimensions el seu tamany creixia molt ràpidament i el límit gratuït de Github no ha estat capaç de guardar-ho tot.

La segona eina ha estat Unity Collab. Anteriorment s'ha utilitzat Unity Collab en un projecte de dos persones, però va acabar malament ja que es perdien moltes de les referències dels objectes del joc. De totes maneres s'ha intentat utilitzar-ho de nou ja que en aquest cas només treballarà una persona i també ha tingut moltes noves actualitzacions. Tot i donar-li una segona oportunitat, ha acabat apareixent el mateix problema d'espai que amb Github.

Finalment s'ha utilitzat Subversion (svn) com a control de versions, ja que el tamany del repositori és una cosa configurable.

Figura 2: Logotip de Apache svn

2.2 Pressupost

Per tal de començar el desenvolupament del joc s'ha fet un càlcul de tots el pressuposts implicats i del personal. Obviament en el cas d'aquest treball el pressupost no ha estat precisament alt, però igualment es mostra una taula amb els pressuposts del projecte en un àmbit real.

A nivell de personal:

Tipus de treball	**Cost per hora (Euros)**
Animador 3D	12.5€
Programador	14€
Encarregat audio	12€
Tester	12.5€
Director	17.5€

Taula 1: Costs de personal

L'animador 3D s'ocupa de fer els models del mapa, personatges, enemics, etc. Bàsicament s'encarrega dels dissenys.
El programador s'ocupa de fer el desenvolupament general de l'aplicació a nivell de codi.
El director, que s'ocupa de les decisions de disseny i de dirigir una mica a la resta de treballadors.
L'encarregat d'audio ha d'afegir els efectes de so.
I finalment un tester per anar provant els canvis en busca d'errors i bugs.

Els costs per hora s'han basat en la mitjana de sous a Espanya.

A nivell de llicències:

Software	**Cost (Euros)**
Unity	399€/any
Blender	0€
Photoshop	24€/mes
Wwise	0€

Taula 2: Costs de llicències

La llicència de Unity és a nivell d'empresa i permet compartir-la a entre

els treballadors dins d'una empresa

Blender és un software de codi lliure per realitzar el modelatge 3D.
Photoshop, per fer dissenys.
Wwise permet treballar els sons. Aquest programa no s'ha utilitzat en el desenvolupament del joc, però s'havia definit inicialment i en cas de que fos un projecte real si que s'hauria utilitzat.

Finalment també s'hauria de tenir en compte el cost del hardware, ja sigui ordinadors i complements, i el cost dels extres (llum, etc), però aquest preu seria molt variable.

Tipus cost	**Cost (Euros)**
Llum	200€/mes
Ordinadors	1000€/u
Internet i telèfon	50€/mes
Impostos	200€/mes
Extres	1000€

Taula 3: Costs variables

Respecte al cost de la llum i els impostos s'ha considerat un preu dins de la mitja.
Per veure quin preu tindria l'Internet i el telèfon s'han buscat ofertes populars per les empreses.
Pels ordinadors s'ha pensat que si es compressin de més barats, la seva durabilitat seria menor, però també que si es compressin més cars seria innecessari per la quantitat de tasques que els treballadors tindrien (no és necessari un ordinador tan potent). A causa d'això, s'escollirien ordinadors de gamma mitjana.
Finalment, pels costs extres s'entenen els costs de reparacions i urgències que puguin aparèixer durant el desenvolupament.

3 Desenvolupament del treball

En aquest apartat s'explica tot el procediment que s'ha realitzat en quant el desenvolupament del joc. S'inclouen totes les decisions, com s'han estructurat les tasques, etc.

3.1 Gestió de les tasques

Per gestionar les tasques a realitzar s'ha utilitzat l'eina Kanban.

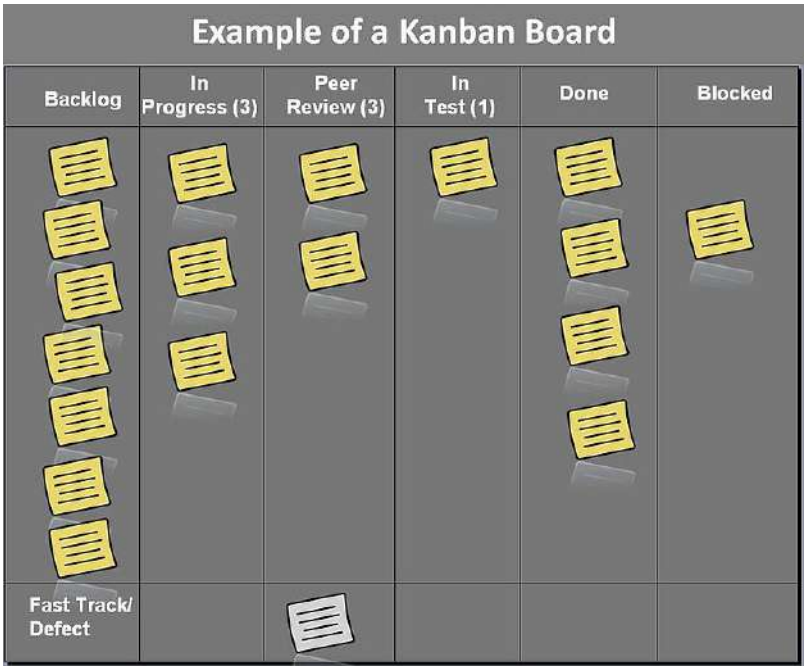

Figura 3: L'eina Kanban

Al estar treballant individualment Scrum no és una manera adequada de gestionar la feina (dividir en sprints), llavors s'ha decidit fer-ho així ja que és una manera de gestionar el treball de manera fàcil i còmoda. Té control de totes les tasques ja fetes i de les que queden per fer i provar.

3.2 Definició inicial del projecte i objectius del joc

Inicialment, el joc s'havia plantejat com un joc del tipus Hack and Slash i d'acció. L'objectiu era superar nivells de plataformes amb diversos enemics i l'objectiu principal era arribar a un punt determinat, derrotar tots els enemics, o sobreviure onades d'enemics.

Durant el desenvolupament d'aquesta versió del projecte van apareixer diverses dificultats, ja sigui amb els components de Unity com a l'hora de la implementació dels coneixements del Màster en codi (la IA, bàsicament). A més, tampoc era un projecte que fos massa original i no era una cosa disfrutable a l'hora de desenvolupar-ho

Degut a aquests problemes es va pensar que fer un replantejament del joc podria ser una bona idea i es va decidir tornar a refer-ho amb una nova idea. Amb aquesta nova idea replantejada, els punts anteriors es faciliten considerablement i a més el joc és considerablement més original i molt més agradable de fer per al desenvolupador.

A partir d'aquí es tindrà en compte la nova versió del joc, ja que és la que ha donat lloc al producte final.

3.3 Conceptes de Unity

Per tal de seguir explicant els següents apartats serà necessari refrescar la memòria sobre alguns elements de Unity, ja que s'aniran anomenan durant el transcurs del document. Aquests elements són els GameObjects, els tipus de GameObject més utilitzats i alguns dels components d'aquests objectes.

3.3.1 Objectes i tipus

Un **GameObject** es podria definir com qualsevol element que apareix en una escena de Unity, ja sigui un personatge, un botó, una barra de vida, un text, etc. Un GameObject per sí mateix no és res especial, però gràcies als components que li pots associar es poden aconseguir comportaments específics per cada objecte.

Un altre element molt important de Unity són els prefabs. Un **Prefab** es pot definir com un GameObject de l'escena emmagatzemat com a fitxer, mantenint la configuració, tots els components i els valors originals del GameObject al qual està associat. Gràcies a això es poden fer instàncies d'un

objecte fàcilment ja sigui des de codi o manualment a la escena. Un prefab pot incloure diversos GameObjects simultàniament, per exemple, si es fa el prefab del canvas de l'escena també es guardaran els panells, els botons, els menús, etc.

3.3.2 Components principals de Unity més utilitzats

Inicialment un GameObject sempre té el component **Transform**. El component Transform indica la posició, rotació i escala d'un GameObject, en els 3 eixos de coordenades x, y i z. La posició ve definida com un Vector3 (de 3 dimensions). En canvi la rotació ve definida com un Quaternion. En cas que l'objecte només tingui aquest component el tractarem com a objecte buit, ja que en una escena no es veurà a temps real (serà invisible), ja que no té forma ni cap model a representar.

Alguns dels objectes que s'utilitzen a les escenes són objectes buits amb scripts associats, de forma que no apareixen directament a temps real però segueixen estant allí com a controladors. Això és molt útil quan es vol aconseguir un comportament genèric que no pertany a cap objecte de l'escena específicament, com per exemple el sistema de torns de la batalla (el controlador decideix de qui és el torn).

Un dels components més importants utilitzats és el component **RigidBody** o els **Colliders** de qualsevol tipus (quadrat, circular, adaptable, etc), que permeten que un objecte tingui un cos físic (no es pugui travessar, etc) i col·lisioni amb altres. La majoria d'objectes de les escenes tenen aquest component associat.

Un altre component molt important i que ha facilitat molt la feina és el component **Character Controller**. Aquest component és relativament nou a Unity3D i, a part de que també proporciona allò que dóna el component RigidBody, permet un control a nivell de codi dels moviments dels objectes molt més simple i també et permet configurar altres paràmetres, com per exemple en cas de que hi hagi una pujada si el personatge es capaç de pujarla o és massa alta per ell (tot es configurar modificant els paràmetres del component). A nivell de codi, utilitzant aquest component es pot aconseguir moure un personatge molt més fàcilment que amb altres mètodes.

3.4 El joc actual

En aquest subapartat s'explica el joc: com funciona, les mecàniques i com es pot guanyar, tots els elements que hi apareixen, la IA aplicada, com es guarden les dades i algunes de les idees per a futurs desenvolupaments.

3.4.1 Descripció i objectius

La idea bàsica del joc es basa en una torre amb infinits pisos, on el jugador té com a objectiu pujar el més amunt possible, derrotant als enemics que es troba. Actualment el joc no té cap història, però a partir d'aquesta idea es podria pensar en alguna cosa.

El joc comença al primer pis. Un cop el jugador completa cada pis pot anar al següent. El fet de que s'hagi decidit que hi hagi infinits pisos és una manera de donar al joc una manera d'anar expandint-se en el temps, de manera que es pugui anar afegint nou contingut de manera il·limitada.

Per tal de completar un pis, és necessari aconseguir una clau (que es pot aconseguir en el mateix pis). També hi han diversos enemics als quals pots combatre o evitar.

Figura 4: Visualització d'una escena del joc

3.4.2 Objectes

En una escena del joc hi ha sempre els mateixos objectes bàsics:

- **El personatge principal.** És el personatge que el jugador controla. En els escenaris el jugador pot moure el personatge lliurement i investigar. Pot obrir cofres, entrar als portals i combatre els enemics.

Figura 5: Personatge principal

- **Els enemics.** Inicialment es troben dins el mapa realitzant un patró predeterminat, ja sigui moures aleatòriament o a punts definits. Un cop el jugador entri al seu camp de visió el perseguiran fins que l'atrapin o el perdin de vista. En cas que atrapin el jugador, o bé el jugador sigui el que s'ha apropat suficientment, començaria el combat.

Figura 6: Exemple enemic

- **La UI.** La interficie d'usuari mostra la vida, l'energia, la barra amb la experiència que té l'usuari, el botó del menú i la icona que mostra si tens la clau o no.
- **Cofre i clau.** El cofre apareixerà sempre per l'escenari. En obrir-lo s'obté la clau que permet superar el nivell.

Figura 7: Cofre a l'escenari

Un cop el jugador s'apropi el suficient s'obrira i donarà la clau. La icona de la part superior dreta de la UI et mostrarà si es té o no la clau.

Figura 8: Clau a la interficie d'usuari

- **El portal**. El portal és la sortida de cada pis. Inicialment el portal

estarà tancat i per tal d'entrar-hi és necessari trobar la clau que es troba al pis actual.

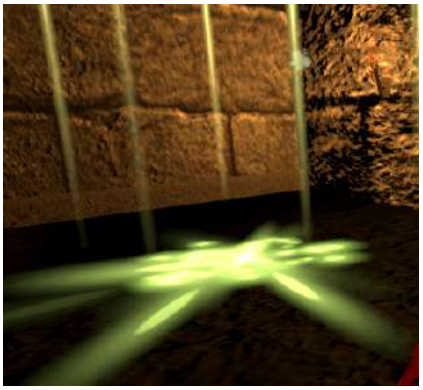

Figura 9: Portal tancat

Figura 10: Portal obert

- **El controlador.** El controlador de cada escena és un GameObject buit i té responsabilitats diferents. L'única cosa en comú que tenen és que carreguen les dades guardades i les assignen als GameObject corresponents (volum, atributs del jugador, etc).
  - El primer controlador és el WorldController, que s'ocupa de controlar tot el que passa en una pantalla (quan no s'està en batalla). Aquest controlador carrega les dades del personatge necessaries en aquest moment i de controlar els enemics que apareixen. S'ocupa també de controlar el portal (si s'obre o es tanca), de l'obtenció de la clau i de carregar la batalla.
  - El segon controlador és el BattleController i és el més complex de tots. Igual que l'anterior, carrega les dades guardades necessàries i les assigna. Aquest controlador crea un Enum que s'encarrega de controlar l'estat de la batalla (si s'està sel·leccionant l'objectiu, si toca atacar al jugador, si és el torn dels enemics, si s'han acabat els torns, etc). També s'ocupa de fer apareixer els enemics corresponents i d'amagar o mostrar les cartes.
  - Finalment l'últim controlador és el del menú principal, i bàsicament s'ocupa de mostrar els menús corresponents i de carregar els nivells que toquen.
- **Altres elements:** En els escenaris també es trobaran altres tipus

d'objectes, ja siguin parets, terra, sostre, elements que proporcionen il·luminació, decoració. Cap d'ells no tenen interacció possible.

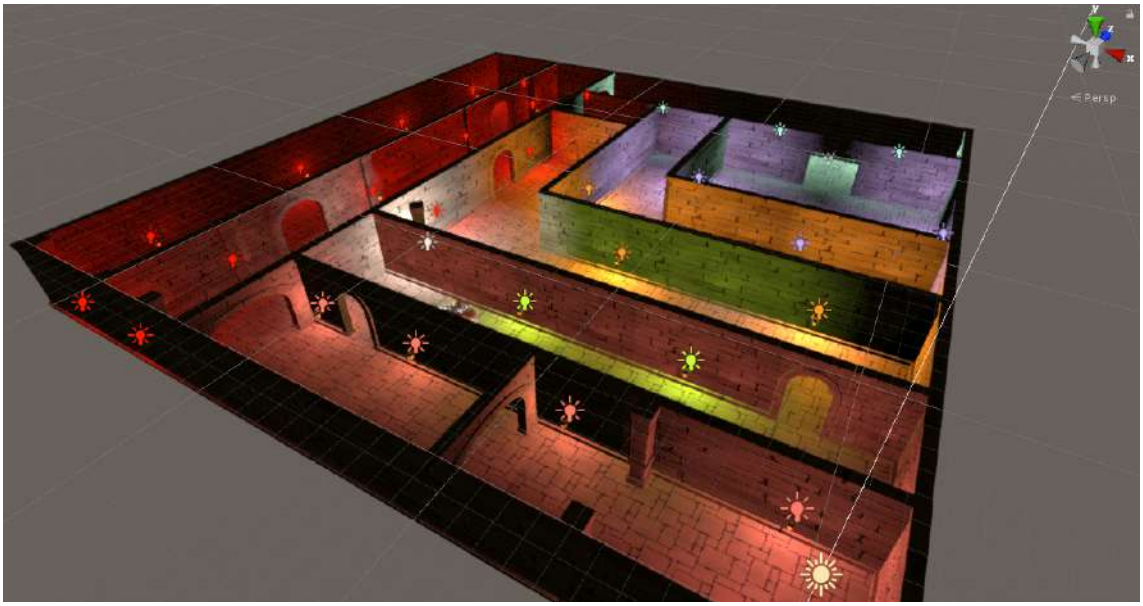

Figura 11: Exemple d'escenari

3.4.3 Sistema de combat

El combat es basa en un sistema de torns. El jugador és el primer en atacar, després la resta d'enemics. La quantitat d'enemics és variable. Com a mínim sempre n'apareixerà un (concretament el que hagis interactuat a l'escenari per combatre) i com a màxim quatre. Tant el jugador com l'enemic tenen una quantitat específica del que podríem anomenar "energia", "stamina"o "màgia", que és el que permet realitzar accions durant la batalla.
En cas que un dels combatents es quedi sense aquesta energia o no en tingui prou per realitzar una acció, haurà de passar el seu torn sense fer res. D'aquesta manera s'ha definit un sistema de batalla que requereixi pensar en quina estratègia seguir. Quan s'acaba el torn, s'hagi atacat o no, es recuperarà 10 punts d'energia. El passar el torn no recupera una quantitat extra d'energia, ja que es considera que si has passat el torn significa que t'has

quedat sense energia aplicant llavors una penalització.

Cada combatent, ja sigui jugador principal o enemic, tindrà tres atacs diferents amb un efecte i cost d'energia diferent en cada un d'ells (evidentment, com més poderós, més energia gastarà). El cost de les habilitats és el mateix per tots els participants de la batalla. Les accions dels enemics vindran definides per una IA que se'ls hi ha programat.

El jugador pot triar quina acció realitzar mitjançant un sistema de cartes. En començar el jugador tindrà 10 cartes, 5 d'elles representen l'atac bàsic (costarà 20 d'energia), 3 de les cartes representen l'atac especial (costarà 30 d'energia) i les últimes 2 cartes representen l'atac definitiu (costarà 50 d'energia).

Figura 12: Carta de atac bàsic

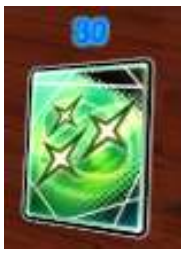

Figura 13: Carta de atac especial

Figura 14: Carta de atac definitiu

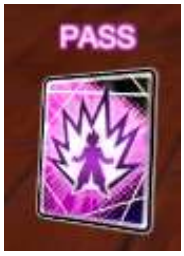

Figura 15: Carta de passar torn

En iniciar el combat, les 10 cartes es barrejaran entre elles i el jugador rebrà a l'atzar 4 de les 10 a la seva mà, de manera que només té 4 accions a triar alhora.

Figura 16: Cartes a la mà

Un cop s'escull una carta, el personatge atacarà l'objectiu seleccionat i es canviarà la carta per una altra de les cartes restants. En cas que només quedin 4 cartes en total i s'en utilitzi una, només en quedarien 3 disponibles, i així fins que no quedi cap. Un cop s'han utilitzat les 10 cartes es tornaran a reiniciar.

El fet de que s'hagi plantejat el sistema d'aquesta manera, buscant que el jugador pensi en una estrategia abans de donar-li a l'atac més fort que tingui, ve de que a més de que tens un limit alhora de fer accions també saps quines són les cartes que t'han d'aparèixer, i això en combats llargs implica decidir si és bona idea gastar energia en un atac fort o gastar-n'hi poca, és a dir, no és un joc d'atzar completament.

El derrotar als enemics donarà experiència de combat al jugador. El personatge principal començarà al nivell 1 i podrà anar pujant de nivell indefinidament derrotant enemics (és un motiu per repetir pisos ja acabats). El pujar de nivell no donarà més atac al usuari, però sí que li augmentarà la vida màxima permanentment, el curarà completament i també se li regenerarà l'energia en cas d'estar en combat.

Figura 17: Imatge d'un combat

3.4.4 La Intel·ligència artificial dels enemics

Per tal que els enemics no ataquin de manera aleatòria s'ha desenvolupat una inteligència artificial per a que siguin capaços de decidir quin atac és el millor en aquell moment. L'enemic no comparteix el mateix mètode per combatre que el jugador. La diferència és que aquests no utilitzen el mètode de les cartes. Els enemics segueixen tenint la mateixa diversitat d'atacs (els 3 atacs diferents) i el sistema d'energia, així que no poden utilitzar sempre l'atac més fort sense conseqüencies. D'aquí ve que establir una IA sigui una cosa molt útil en un tipus d'enemic així.

L'algorisme que s'ha seleccionat per fer la IA és l'Expectimax. L'algoritme Expectimax és una variació del algoritme Minimax. La única diferència que hi ha és que l'expectimax hi aplica una mica l'atzar. Per tal de poder entendre l'Expectimax, es definirà què és l'algoritme Minimax i després la diferència entre ells.

El Minimax utilitza backtracking per aconseguir trobar quin és el moviment òptim per un dels agents del joc suposant que l'altre agent juga òptimament també.

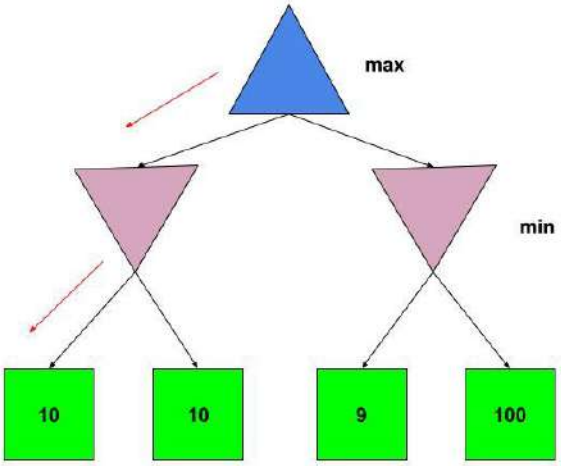

Figura 18: Arbre de l'Algorisme Minimax

En la imatge anterior podem veure un exemple que mostra l'arbre de les accions dels agents. Cada agent té un proposit diferent. Un intenta maximitzar i l'altre minimitzar. Com es pot veure a les fulles, cada una té un valor diferent. L'objectiu del que intenta maximitzar és trobar el camí cap al node amb més puntuació, mentre que l'objectiu del que vol minimitzar és portar a l'altre agent cap al node amb menys puntuació. En el joc l'enemic buscarà el camí que el porti a la seva victòria. En el cas de l'Expectimax, l'objectiu segueix sent el mateix, pero el valor de cada node és una possibilitat d'anar per aquell camí. Com més valor més possibilitat (l'expectimax ja de per sí no és un algoritme òptim)

En el Minimax hi ha variacions de manera que també es puguin tenir en compte les accions del aliats (en el cas del joc, diversos enemics). Per exemple, seria un arbre amb dos o més agents maximitzadors, però s'ha decidit

fer que l'enemic no tingui en compte als seus aliats i que només consideri les seves accions i les del jugador.

Tal i com s'ha comentat, el Minimax és un algorisme que considera que el jugador prendrà la decisió òptima, però com que el jugador és un ésser humà i no una màquina ja podem veure que no serà òptim al 100%. També, l'únic coneixement que l'enemic pot saber de l'usuari és la vida i l'energia que té. En cap moment sabrà quines cartes té actualment. Llavors considerarà que és capaç de fer els tres tipus d'atac en qualsevol moment, tingui o no les cartes corresponents.

Cada node de l'arbre és un estat. Cada estat ve determinat per la següent classe:

```
public class State
{
    //atributs enemic
    public int health;
    public int mana;

    //atributs player
    public int mainHealth;
    public int mainMana;

    //agent - enemic o jugador
    public int agent
}
```

A partir d'un estat, podem obtenir els següents estats a partir d'una acció. Per exemple, si l'enemic fa un atac especial en un estat determinat podem definir el següent estat calculant la vida i energia dels agents en aquest nou estat (mètode GetNextState). També a partir d'un estat podem calcular la puntuació d'aquest node amb la EvaluationFunction.

Per calcular el valor d'un estat hem d'associar un pes a cada atribut, ja sigui vida o energia, de manera que a l'hora de calcular-ho el tenir més vida i energia i que el contrincant tingui menys vida i energia doni més valor al node.

La utilitat d'aquesta EvaluationFunction és que es pot configurar la importància dels atributs de manera que es poden aconseguir diferents comportaments en els enemics (es poden fer més passius o agressius depenent només dels coeficients que s'estableixin).

Exemple de funció d'evaluació (els valors poden canviar en la versió final):

```
public int EvaluationFunction()
{
    int val = 0;
    float weightMana = 0.7f;
    float weightHp = 0.3f;
    val += (int) (health * weightHp);
    val -= (int) (mainHealth * weightHp);
    val += (int)(mana * weightMana);
    val -= (int)(mainMana * weightMana);
    return val;
}
```

La profunditat de l'arbre a la que arriba l'algorisme ve definida per la variable depht. Depenent d'aquesta variable, es mirarà una quantitat determinada de possibles moviments i el seu valor. Actualment la profunditat està definida a 2 o 3, ja que augmentar aquest valor implica augmentar exponencialment el cost de processament.

Per saber quan s'ha de deixar de buscar es té la funció TerminalTest, que a partir d'un estat comprova si en poden sortir més accions. Bàsicament comprova si ja s'ha arribat a la profunditat màxima o s'ha mort l'objectiu o el mateix agent.

```
bool TerminalTest(State state, int depth)
{
    return depth == 0 || state.mainHealth == 0 || state.health == 0;
}
```

3.4.5 Menú principal

Figura 19: Menú principal

Al menú principal del joc hi haurà bàsicament tres funcionalitats.
La primera és la de començar un joc nou. En cas que es cliqués en aquest botó es borrarien les dades guardades i es començaria de nou al primer pis.
La segona funcionalitat seria continuar el joc. Et deixarà escollir a quin pis vols anar i, s'hagi passat el pis o no, un cop s'entra des del menú principal es tractarà com si no s'hagués passat. És a dir, es començarà el pis amb tota la vida i energia, i els enemics reapareixeran (també s'haura d'aconseguir la clau un altre cop).

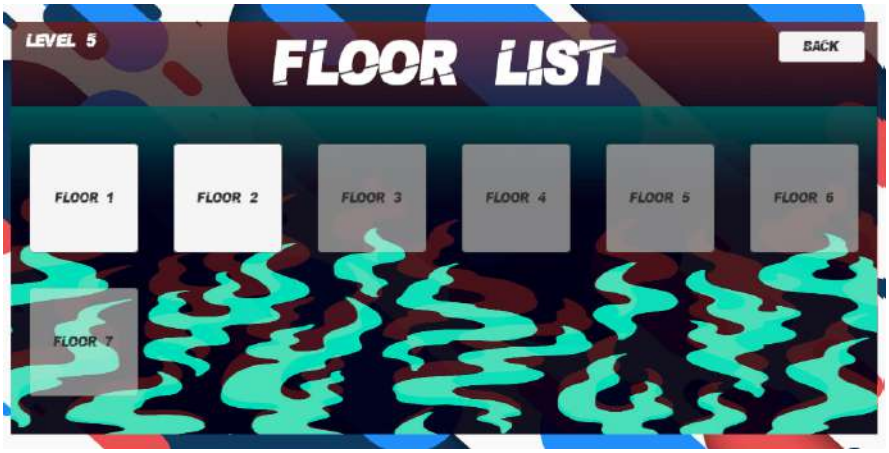

Figura 20: Menú seleccionar pis

L'última funcionalitat seria el menú d'opcions, des del qual es podrà establir el valor del volum de la música del menú, escenaris i combat.

Figura 21: Menú opcions

3.4.6 Persistencia de dades

Tenim dos tipus d'emmagatzematge de dades:

- El primer tipus és l'emmagatzematge de les dades mentre el joc està obert. Es a dir, des de que s'obre fins que s'apaga el joc. Aquest tipus d'emmagatzematge es fa des d'una classe estàtica anomenada ApplicationStaticData.

```
public static class ApplicationStaticData
{
    public static int mobToBattle;
    public static List<string> mobsToDelete = new
        List<string>();
    public static string mobToBattleName;

    public static Vector3 lastCharPosition;
    public static bool hasKey;
    public static int actualEsceneId;

    public static bool clearedStage;
}
```

  En guardar les dades en aquesta classe les podem consultar des de qualsevol escena sense perdre-les (en cas que es tanqui el joc es borrarien). Aquest és el metode fet per poder guardar les dades entre escenes.

  - **mobToBattle**: Serveix per saber quin tipus d'enemic és amb qui has entrat en col·lisió, de manera que es pugui assegurar que al entrar al combat aparegui el que toca. El que es guarda es l'ID que té l'enemic en particular, per identificar quin ha d'instanciar primer. Un cop s'acaba el combat es reinicia aquest camp.
  - **mobsToDelete**: Es una llista amb els noms dels enemics que s'han de borrar. Per exemple, si es derrota a l'enemic d'un escenari (encara que n'apareguin 4 al combat), un cop es guanyi aquest no ha de tornar a aparèixer a l'escenari. Aquest camp es reiniciar en acabar la pantalla.
  - **mobToBattleName**: Igual que el primer camp mobToBattle, aquest guarda el nom de l'enemic amb qui s'ha xocat, de manera que en cas que es guanyi el combat s'afegirà a la llista anterior i es reiniciarà aquest camp.

- **lastCharPosition**: Serveix per guardar la posició del personatge principal abans d'entrar en batalla. S'utilitza per a que quan s'acabi el combat el personatge aparegui on toca.
- **hasKey**: Indica si s'ha obtingut la clau.
- **actualEsceneId**: Serveix per saber l'escena que ha de carregar el controlador un cop acabes el combat.
- **clearedStage**: Serveix per a que un cop s'ha acabat una pantalla, durant la càrrega del menú principal, et redirigeixi al menú dels pisos depenent de si s'acaba de finalitzar la pantalla o no.

- El segon sistema de persistència de dades és el que es guarda inclús quan es tanca el joc. Per fer-ho es guarda un fitxer binari que representa una clase serializable. D'aquesta manera el jugador en cas que vulgués modificar algun paràmetre del joc (bàsicament fer trampa) tindria difícil trobar el fitxer i editar-lo.

  Les classes que emmagatzemen les dades a guardar són les següents:

```
[System.Serializable]
public class SaveData
{
    public int level;
    public float exp;
    public float expneeded;

    public int maxHealth;
    public int maxMana;
    public int health;
    public int mana;

    public int damage_basic;
    public int damage_skill1;
    public int damage_skill2;

    public int floor;

    public SaveData(GameObject player)
    {
        Character avelyn = player.GetComponent<Character>();
        level = avelyn.level;
        exp = avelyn.exp;
        expneeded = avelyn.expneeded;
```

```
        maxHealth = avelyn.maxHealth;
        maxMana = avelyn.maxMana;
        health = avelyn.health;
        mana = avelyn.mana;
        damage_basic = avelyn.damage_basic;
        damage_skill1 = avelyn.damage_skill1;
        damage_skill2 = avelyn.damage_skill2;
        floor = avelyn.floor;
    }
}
```

```
[System.Serializable]
public class OptionsData
{
    public float volume;

    public OptionsData(float db)
    {
        volume = db;
    }
}
```

A la primera classe les dades que es guarden són bàsicament els atributs del personatge principal i l'últim pis completat. A la segona classe es guarda el volum de les opcions. S'ha decidit que cada classe es guardi en fitxers diferents ja que al ser dos conceptes diferents és millor tenir-los separats.

3.4.7 Problemes durant el desenvolupament

Alguns dels problemes que han aparegut durant el transcurs del desenvolupament són els següents:

- El primer problema i més important possiblement hagi estat el problema amb les col·lisions a la primera versió del joc. En ser un joc en tres dimensions, a part de que detectar les col·lisions té un cost més elevat també augmenta la dificultat a l'hora de tractar-les. Per exemple, en utilitzar models 3D externs pels personatges, la dificultat per afegir un collider a un element del model (espasa, etc) i que no influeixi amb la resta era força complicat.
- El segon problema no afecta cap funcionalitat però sí a un dels punts inicials definits del joc. Aquest problema és la incapacitat de poder

traslladar el joc a una consola de Nintendo, ja que per poder-ho fer és necessari descarregar una nova versió de Unity de Nintendo (no és el mateix Unity). Aquesta versió a part de ser de pagament mensual, utilitza tecnologies diferents (llenguatge de programació, etc).

3.4.8 Idees per una futura continuació en el projecte

La base del joc està complerta, però tal i com està feta l'estructura del joc deixa la possibilitat d'afegir noves funcionalitats en un futur. A continuació s'explicaran algunes d'aquestes possibles funcionalitats que es podrien afegir de manera que es doni més vida al joc.

- La cosa a afegir més imporant i més lògica és posar nous pisos. Això implica anar afegint nous tipus d'enemics i d'escenaris amb més dificultat. Afegir pisos cada X temps dóna vida al joc de manera que no esdevingui avorrit i el jugador no es quedi sense motivació per continuar jugant. Aquestes millores serien les més dificultoses i més costoses.

- En pujar de nivell ara mateix no es guanya cap atribut més que la vida màxima. Això vol dir que a mesura que es vagi pujant la dificultat els combats es tornaran, o bé molt més llargs o massa difícils, ja que els enemics augmentaran en vida i atributs. L'opció que s'havia pensat era afegir un sistema d'equipament. Ara mateix els cofres d'un escenari et donen únicament la clau. El que es podria fer és que hi hagi una probabilitat de que et donin una arma o armadura equipable que pugui augmentar els teus propis atributs d'atac i defensa. Com més amunt de la torre, millors equipaments et podries trobar, i, en cas que sortís un equipament repetit, millorar el que ja tens actualment o fer que es pugui vendre (això implicaria un desenvolupament d'una altra funcionalitat relacionada amb diners del joc). A nivell de desenvolupament i costs això implicaria afegir un nou menú amb els equips, ja sigui als escenaris o al menú principal, on es pugués equipar l'equip, i també afegir la probabilitat de que els cofres donin un equipament, pero sempre assegurant que un d'ells doni la clau per passar al següent pis. També s'hauria de fer el disseny de cada equip.

- Una altra funcionalitat que es podria incorporar és l'afegir nous personatges jugables. La manera d'obtenir aquests personatges seria comprant-los o a l'atzar. Però amdós casos és necessari un sistema de diners com el que s'ha comentat al punt anterior. Els nous personatges jugables tindrien atributs i accions diferents, per a que el jugador tingui una

sensació de novetat al provar-los. També es podria afegir un sistema per a que en cas que vulguessis i tinguessis més d'un personatge, et puguessin ajudar dins dels combats i que es pugués acabar fent un 4 contra 4.

- Ja que el pujar nivell no augmenta res més que la vida, del personatge es podria afegir alguna cosa extra, com per exemple noves habilitats que substitueixin les accions actuals de les cartes. Per exemple, que en alguns nivells determinats s'aconsegueixin noves habilitats que es puguin canviar per les actuals, amb costs d'energia i efectes diferents o similars. D'aquesta manera es crea un incentiu en pujar de nivell. Aquest canvi implicaria afegir noves animacions per cada habilitat i implementar-les. A nivell de cost aquest seria el canvi menys costós.

Havent definit els canvis, es podrien classificar de diverses maneres. Considerant el cost, de major a menor:

- Noves pantalles,
- Nous personatges,
- Sistema d'equipament,
- Noves habilitats.

Fer nous personatges implica també fer les seves animacions a part de tot el modelatge general (i la quantitat de personatges nous), així que el sistema d'equipament tindrà un cost total menor al d'afegir nous personatges.

Considerant la quantitat de contingut que donarien els canvis, la classificació quedaria així, també de major a menor:

- Noves pantalles,
- Sistema d'equipament,
- Noves habilitats,
- Nous personatges.

Per "quantitat de contingut"en aquesta classificació es considera el temps que el jugador passarà realitzant els punts anteriors. El fer noves pantalles seria el més imporant ja que és la part principal del joc i influeix molt a la resta de punts (tal i com s'ha comentat abans, com més avançat s'està s'aconseguirà millor equip i experiència). A part d'això el sistema d'equipament

seria per a tots els personatges. Això vol dir que es pot compartir l'equip per tots els personatges i no és necessari aconseguir un nou personatge per obtenir nou equip. Les noves habilitats és un incentiu per pujar de nivell i a part també facilita l'avançar en les pantalles. Els nous personatges finalment, encara que sigui novetat per al jugador, començarien des del nivell 1 (cada personatge tindria un nivell diferent).

La classificació dels futurs canvis tant per cost com per capacitat de donar contingut al jugador s'ha fet per poder decidir millor quins són els canvis amb més prioritat. El que ha quedat clar és que l'afegir noves pantalles és el més important. Però per la resta s'hauria de fer un estudi per decidir amb què avançar.

4 Conclusions

El canvi realitzat en el plantejament del joc i començar de zero un altre cop és una cosa que no esperava. El fet que comenci comentant aquest punt és perque trobo interessant que s'haguès començat el desenvolupament del joc amb una idea que possiblement ve amb la influència d'altres jocs jugats i al final s'hagi acabat convertint en una idea més o menys original i que s'ha pogut disfrutar molt a l'hora de fer-la realitat. També penso que en cas d'haver seguit la primera idea segurament la dificultat de seguir hauria estat molt més gran i no s'hagués passat tant bon temps fent el projecte. Tampoc no s'haurien tingut tantes ganes de seguir endavant i el producte final hauria estat pitjor.

Respecte les tecnologies utilitzades, la majoria havien estat ja utilitzades anteriorment i ha estat agradable tornar-les a fer servir un cop més. El control de versions és l'únic problema que ha donat molts mals de cap i ha fet perdre molt de temps, ja que inicialment el projecte es podia guardar correctament, però un cop importats un parell d'assets la quota de disc ja ere massa petita.

Sobre el modelatge, durant el treball de fi de grau també es va fer un joc, pero en 2 dimensions. Al ser de 2 dimensions els models del joc són bàsicament sprites i es poden fer amb facilitat. En el cas d'un joc en 3 dimensions els models s'ha vist que és un món molt més complex i que és necessari tenir algú que entengui del tema per fer els models i animacions. Personalment es va intentar crear un model 3D amb la intenció de no dependre d'assets externs, però a part de que va consumir una quantitat de temps considerable, el resultat no era suficientment bo, cosa que va animar a descarregar assets gratuïts de la store de Unity. També cal dir que la música del joc és música amb copyright. No hi ha cap problema ja que el joc no es publicarà de moment, però en cas que es fes possiblement s'hauria de canviar la música i potser algun dels models.

Els coneixements del Màster crec que s'han aplicat força bé en el joc, i crec que la IA seleccionada és la correcta en un joc d'aquest tipus. És cert que hi ha molts més algorismes, uns que fan que els enemics vagin aprenent en el temps, altres que són similars al que s'ha aplicat, però igualment penso que al ser un combat per torns l'enemic no ha d'acabar sent tant complexe i el fet de donar-li una mica d'aleatorietat és un punt a favor per al jugador.

Finalment, considero que el resultat final del joc és correcte i estic orgullós d'haver desenvolupat un joc més o menys original i haver-li aplicat intel·ligència artificial.

5 Bibliografia

Referències

[1] Unity. https://docs.unity3d.com/Manual/index.html

[2] Visual Studio. https://docs.microsoft.com/es-es/visualstudio/?view=vs-2019

[3] Photoshop. https://www.softwaredoit.es/definicion/definicion-adobe-photoshop.html

[4] Scrum. https://www.scrum.org/resources/what-is-scrum

[5] Kanban. https://www.versionone.com/what-is-kanban/

[6] Blender. https://docs.blender.org/

[7] Tutorials YT de Unity - Brackeys. https://www.youtube.com/user/Brackeys