

Paintgeon

Projecte Síntesi



8 de mayo de 2017

Miquel montaner i antoni martin

Institut Rafael Campalans

2n DAM

ÍNDEX

[Introducció i objectius. 3](#_Toc483875736)

[Descripció 3](#_Toc483875737)

[Finalitats del projecte 4](#_Toc483875738)

[Utilitats 4](#_Toc483875739)

[Tria de llenguatges de programació 4](#_Toc483875740)

[Anàlisi, disseny i construcció del projecte 5](#_Toc483875741)

[Organització 5](#_Toc483875742)

[Carpetes 6](#_Toc483875743)

[Explicació de cada carpeta 6](#_Toc483875744)

[Fases 7](#_Toc483875745)

[Temporització 7](#_Toc483875746)

[Diagrama de Grantt 8](#_Toc483875747)

[Anàlisi D.A.F.O 9](#_Toc483875748)

[Pressupost del projecte 9](#_Toc483875749)

[Anàlisi del hardware necessari 9](#_Toc483875750)

[Anàlisi del software necessari 10](#_Toc483875751)

[Requeriments 10](#_Toc483875752)

[Àmbit i camp del projecte 10](#_Toc483875753)

[Requeriments funcionals 10](#_Toc483875754)

[Requeriments no funcionals 10](#_Toc483875755)

[Rendiment 11](#_Toc483875756)

[Anàlisi 11](#_Toc483875757)

[Diagrames de Casos d’Ús 11](#_Toc483875758)

[Fitxes de Casos d’Ús 12](#_Toc483875759)

[Diagrames d’activitat 13](#_Toc483875760)

[Disseny 14](#_Toc483875761)

[Diagrama de classes 14](#_Toc483875762)

[Disseny modular del projecte 15](#_Toc483875763)

[Estructura de les dades 15](#_Toc483875764)

[Model entitat relació de la BD 15](#_Toc483875765)

[Model relacional 16](#_Toc483875766)

[Acces a la base de dades 16](#_Toc483875767)

[Disseny de les interfícies 16](#_Toc483875768)

[Descripció de les interfícies 16](#_Toc483875769)

[Aspectes de seguretat 18](#_Toc483875770)

[Accés a les dades 18](#_Toc483875771)

[Explicació procés 19](#_Toc483875772)

[Animacions 19](#_Toc483875773)

[Personatges 19](#_Toc483875774)

[Objectes 21](#_Toc483875775)

[Efectes 21](#_Toc483875776)

[Programació 22](#_Toc483875777)

[Personatges 22](#_Toc483875778)

[Jugador 23](#_Toc483875779)

[Enemics 23](#_Toc483875780)

[Profunditat 23](#_Toc483875781)

[Opcions 24](#_Toc483875782)

[Bade de Dades 25](#_Toc483875783)

[Guardar partida 27](#_Toc483875784)

[Musica 28](#_Toc483875785)

[Fons 29](#_Toc483875786)

[Sons 29](#_Toc483875787)

[Conclusions 29](#_Toc483875788)

[Bibliografia 30](#_Toc483875789)

[Comentari personal 30](#_Toc483875790)

# Introducció i objectius.

## Descripció

El nostre projecte consisteix en crear un videojoc a partir del programari Unity, amb ajuda d'altres, com pot ser Photoshop, amb ell hem creat la part artística del videojoc, des d'interfícies, logotip, personatges, enemics, mapejat, etc.

El videojoc consistirà en completar nivells i aconseguir monedes per expandir el poble central. Cada vegada que es completa un nivell es tornarà al poble, on el jugador podrà preparar-se pel següent nivell. A cada nivell hi haurà una sèrie d'enemics i uns quants puzles que haurà de superar. Al final del nivell hi haurà un enemic final que s'ha d'eliminar per superar el nivell.

Amb les monedes aconseguides a cada nivell, podràs ampliar el poble amb més opcions.

La música l'hem extret d'internet. Quan ens referim a música també englobem els sons i petits efectes.

Guardem les nostres partides en un fitxer binari, en una ruta que canvia depenent del dispositiu/plataforma que s'utilitza a l'hora del joc (mòbil, PC, Consola, etc.). Permetem guardar fins a 3 partides.

Hem creat una base de dades per a un bestiari de monstres (Llibre amb informació referent als enemics que apareixen al llarg del videojoc) que incorpora el nostre joc. El bestiari agafa tots els enemics guardats a la base de dades, i carrega els enemics que s’han eliminat en el joc. Com guardem les dades en un fitxer a part, hem hagut de combinar la comanda de la base de dades amb les dades del fitxer. Hem afegit un buscador per filtrar els enemics. També hem afegit informació de cada nivell i quins enemics apareixen.

## Finalitats del projecte

El nostre objectiu des del principi era crear un videojoc, i que tot fos creat per nosaltres.

Tot comença en un poble, en el qual hi ha una sèrie d'opcions assignades a NPC's, com podria ser configuració, guardar, començar a jugar, entre d'altres, com si d'un menú d'inici és tractes. Segons avances en el joc superant cada un dels seus nivells i reunint uns col·leccionables que trobaràs en els mapes, es desbloquejaran noves opcions, com per exemple; habilitats pel personatge principal.

Les habilitats mencionades anteriorment funcionen amb una barra d'energia que va baixant segons la vas utilitzant.

El joc acaba quan has desbloquejat tot el poble per complet i t'has passat tots els nivells.

Cada un dels nivells és rejugable.

La història transcorre en una llibreta d'un nen, i l'art del videojoc simula els seus dibuixos a la llibreta. Per això hem dibuixat cada una de les coses accessibles en el joc, no hi ha res extret d'internet, tot és producció nostra a través del Photoshop i el Paint Tool Sai.

## Utilitats

Tractant-se d'un videojoc, la seva finalitat principal és la de divertir a l'usuari final.

També cal esmentar que a part de diversió, s'haurà d'utilitzar la lògica en alguns casos.

L'usuari podrà mostrar les seves habilitats a l'hora de jugar, ja que a part de puzles (lògica), i podrà trobar monstres finals, el qual hi haurà de lluitar.

## Tria de llenguatges de programació

El llenguatge de programació seleccionat per programar cada script és el C# amb Visual Studio, ja que de les opcions que don a escollir el Unity , és el que més hem vist i utilitzat al llarg d’aquest dos anys.

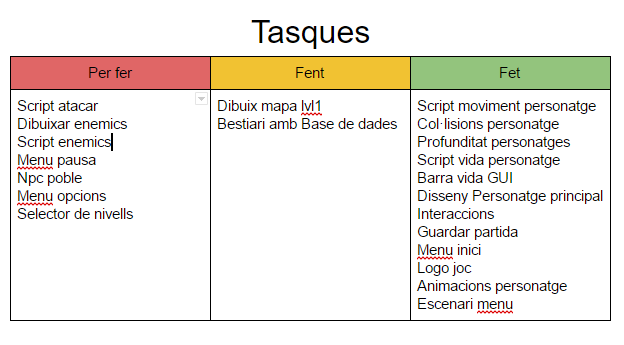
Per a la creació de la base de dades hem seleccionat SQLite, ja que és l’únic que hem vist compatible i de fàcil accés en el programari Unity.

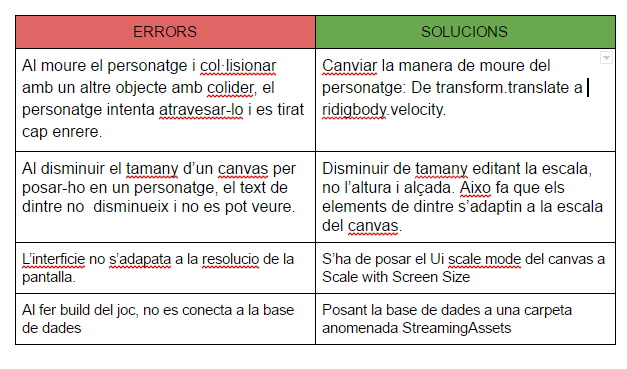
# Anàlisi, disseny i construcció del projecte

## Organització

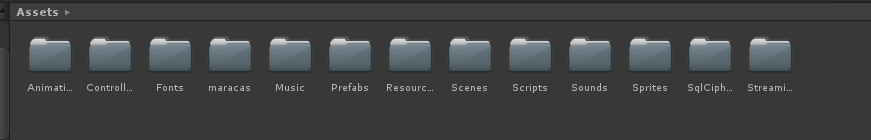
Un s'ha centrat més en la programació base del videojoc i l'altre en els gràfics i la programació de les animacions.

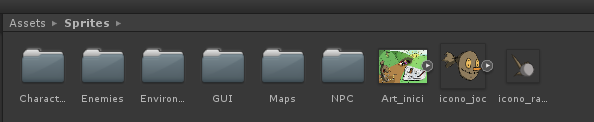
Per a organitzar-nos hem creat un document compartit a Google Drive amb una taula de tasques.

Cada setmana emplenem la primera columna amb la feina que hem de fer durant la setmana. Quan triem que fer, ho posem a la segona columna per marcar que s'està fent. En finalitzar la tasca, la movem a la tercera columna.

Hem creat una taula on posem tots els errors que anem obtinguent en el procés. Si es troba la solució de l'error, s'apunta a la columna de la dreta.

## Carpetes

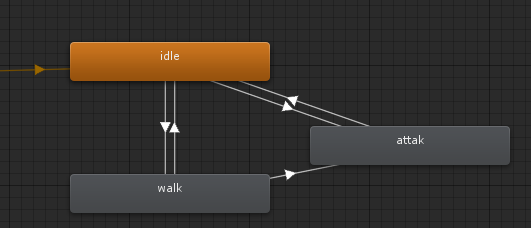
Per ordenar els scripts i les imatges del videojoc, ho hem separat amb carpetes. Cada part important com la música o les animacions tenen la seva pròpia carpeta.

Si el contingut de la carpeta és massa gran, ho separem per carpetes també.

### Explicació de cada carpeta

Animations: Contè totes les animacions del joc. Des de el moviment del personatge fins a l'animació d'un text.

Controllers: Els controllers són els que s'encarreguen d'organitzar i decidir quan s'activarà cada animació. Cada objecte amb animació necessita un controller.



Fonts: Conté les fonts dels textos de la interfície del joc.

Music: Tota la música de fons del joc està en aquesta carpeta.

Prefabs: Els prefabs són objectes que ja han sigut preparats per funcionar. Per exemple, els enemics estan guardats en prefabs, així quan necessitem col·locar un enemic, ja no hem de programar les animacions ni assignar els scripts.

Resources: Quan volem carregar alguna imatge o prefab al videojoc via codi, necessitem tenir-ho a la carpeta Resources per poder accedir. En el nostre cas, ho utilitzem per poder carregar les imatges de cada enemic de la base de dades.

Scenes: Cada escena és un nivell del videojoc. Separar cada nivell en escenes, ajuda a no sobrecarregar una mateixa escena amb molta informació.

Scripts: Es guarden tots els scripts del videojoc.

Sounds: Conté totes els petits sorolls del joc. Per exemple, obrir una porta o rebre mal.

Sprites: Els sprites són totes les imatges del videojoc.

SQLCipher: SqlCipher és un pluguin que hem descarregat per poder accedir a una base de dades xifrada des de Unity.

StreamingAssets: Per poder mantenir la base de dades al crear l'executable, necessitem posar la base de dades dintre d'aquesta carpeta.

## Fases

### Temporització

El temps és un factor que hem de tenir present en tot moment, ja que per realitzar tot el que volem fer necessitem aprofitar-lo al màxim possible.

Part d'aquest el gastarem en saber utilitzar el programari principal (Unity i Photoshop), la primera setmana practicarem la creació dels sprites (imatges per animar un personatge) i com animar-los.

Unes de les coses que hi haurem de dedicar molt temps, és aplicar l'efecte de profunditat en el joc. Quan un jugador està davant d'un objecte, aquest s'ha de sobreposar a l'objecte per tal de donar l'efecte 3D que desitgem.

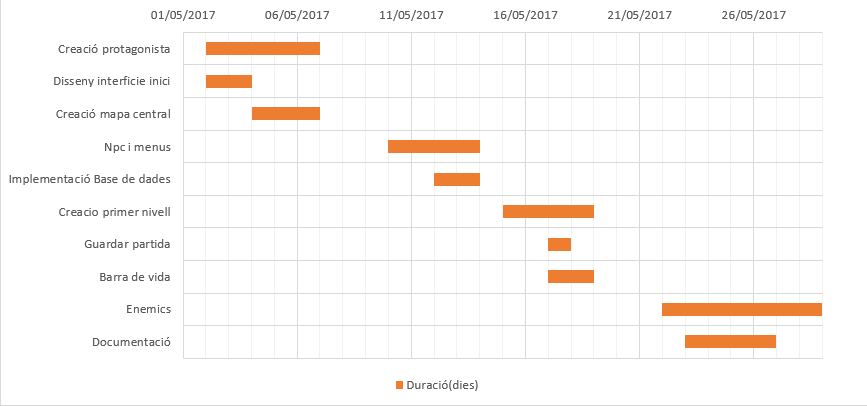
Seguidament en la creació d'interfícies i mapejat, d'aquests dos creiem que deixarem moltes hores en crear els mapes, ja que hem de dissenyar i programar diferents mapes, com poden ser; mapa central, nivells amb una temàtica que els diferencia entre si (neu, lava, bosc, etc.), i un mapa que servirà de guia per a la primera presa de contacte amb el joc.

El mapa central servirà com a menú del joc, contindrà moltes opcions, des d'un NPC que et mostrarà les opcions, fins a un altre que et mostrarà un bestiari, el qual serà una base de dades amb SQLite. Això implica molta programació, el més segur és que treballem fins al final del projecte.

Com hem mencionat anteriorment, crearem una base de dades per a un bestiari, creiem que gastarà molt de temps, ja que hem de mirar com adjuntar Unity amb SQLite, aquesta part també inclou part de disseny, ja que tractant-se d'un bestiari dibuixarem un llibre, així augmentant el temps que dedicarem en ell.

Un altre de les parts en la que gastarem molt de temps serà en la creació dels enemics, ja que cadascun d'ells necessita ser dibuixat per parts, animat i per últim la programació. Per aprofitar el temps, utilitzarem herència per estalviar codi entre els enemics.

### Diagrama de Grantt



## Anàlisi D.A.F.O

|  |  |  |
| --- | --- | --- |
|  | Fortalezas | Debilidades |
| Análisis interno | -Disseny artístic innovador.  -Joc amb sentit de l'humor.  -Interfície simplificada que no mareja al jugador amb massa informació. | -La lluita amb els enemics és molt simple.  -Poc contingut.  -Alguns petits problemes que poden afectar la jugabilitat. |
|  | Oportunidades | Amenazas |
| Análisis externo | -Poca competència amb jocs del mateix gènere.  -Avui en dia és relativament fàcil aconseguir ajuda pel desenvolupament amb plataformes com Kickstarter. | -Molta competència amb jocs de diferent gènere, però molt semblant.  -Altres competidors tenen més temps per treballar en el videojoc. |

## Pressupost del projecte

### Anàlisi del hardware necessari

Només és necessari un ordinador que pugui utilitzar el software necessari per programar el joc i eines per editar imatges i àudio. Hem decidit utilitzar els nostres ordinadors portàtils que hem utilitzat durant tot el curs, ja que superen els requisits mínims del software necessari.

### Anàlisi del software necessari

Necessari tenir Unity instal·lat per poder programar el videojoc. El mateix Unity ja t'instal·la l'IDE necessari com VisualStudio o MonoDevelop. Nosaltres hem decidit utilitzar VisualStudio pel coneixement que ja tenim del programa.

Per dibuixar i editar imatges, programes com Paint Tool Sai o Photoshop. Els dos programes incorporen les eines que hem utilitzat. Cadascú ha triat el programa que sap utilitzar.

Per editar la música i l'àudio utilitzem l'Audacity. Poques vegades necessitarem editar la música i l'àudio, així que Audacity és una bona opció per la seva simplicitat.

Per crear i editar la base de dades hem utilitzat DB Browser for SQLite.

Per facilitat el treball conjunt, hem utilitzat dropbox per compartir el projecte. Així cada vegada que havíem canviat alguna cosa, no ha fet falta passar el projecte a l'altre.

Llicencies:

Adobe Photoshop 2017 – 22,99/mes

## Requeriments

### Àmbit i camp del projecte

-El producte està inclòs en l'àmbit de l'entreteniment, mes específicament en el camp dels videojocs.

### Requeriments funcionals

-El jugador s'ha de moure, atacar o interaccionar en prémer les tecles corresponents.

-Si la vida arriba a 0, el jugador ha d'aparèixer a l'última zona de descans.

-Si parla amb un NPC del poble central, s'ha de mostrar un menú corresponent a l'NPC.

-Quan s'elimina un enemic, pot sortir un objecte per agafar.

-Si s'agafa una moneda, s'ha d'augmentar el comptador de monedes.

-Si s'agafa una curació, hauria d'augmentar la vida sense passar del màxim.

-Si prem la tecla escape en un nivell, s'ha d'obrir el menú de pausa.

-Si prem l'opció sortir, s'ha de tancar el joc.

### Requeriments no funcionals

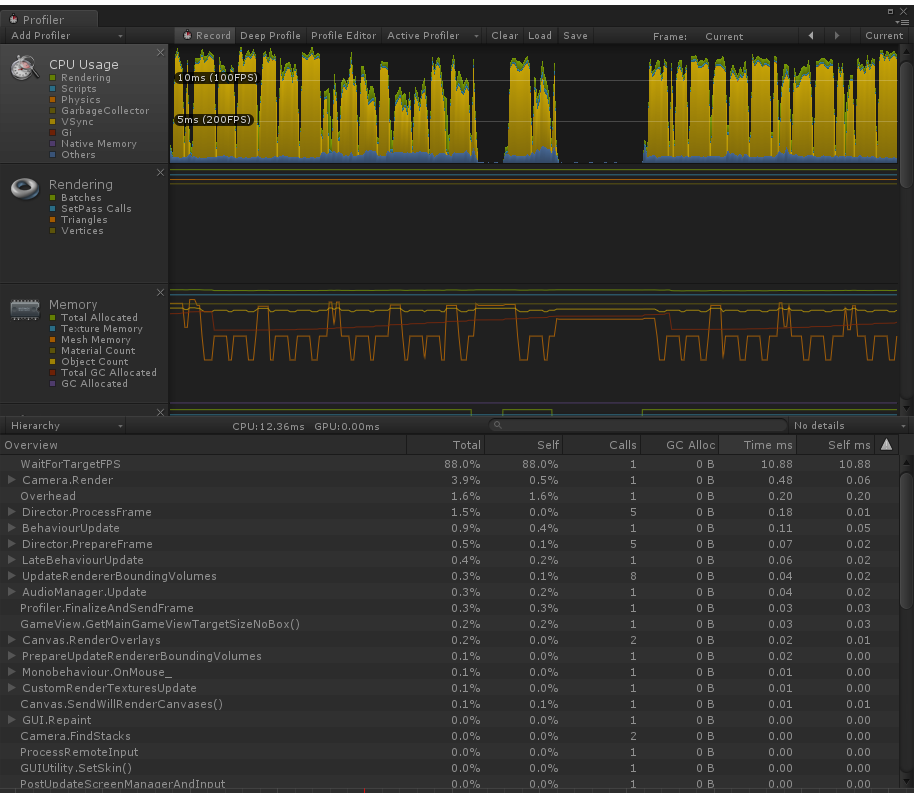
Espai necessari: 500mb

Un ordinador.

Windows.

### Rendiment

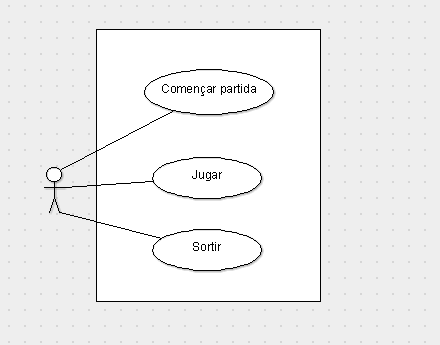
Unity ofereix una eina anomenada ‘Profiler’. Aquesta eina s’encarrega de mostrar-te informació del rendiment del joc. A la part d’abaix surten tots els processos que el joc executa i quant gasta cadascu. Aquesta eina es útil per detectar errors de rendiment i millorar-lo.



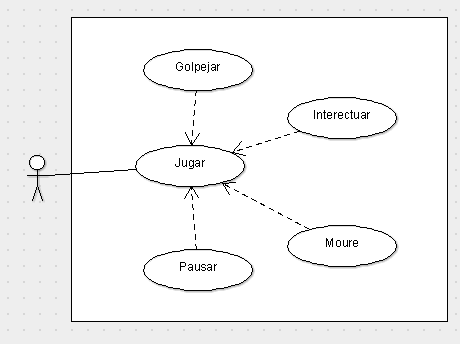
## Anàlisi

### Diagrames de Casos d’Ús

Cas d’ús principal



Cas d’ús de refinament de Jugar



### Fitxes de Casos d’Ús

|  |  |  |
| --- | --- | --- |
| CAS D’ÚS | Nom del cas d’ús | |
| Versió/Data | 1,0 | 20/05/2017 |
| Autors | Miquel Montaner i Antoni Martin. | |
| Descripció | Un usuari vol jugar al joc. | |
| Autors | Jugador. | |
| Precondició | Ha de obrir el joc i triar una partida. | |
| Postcondició | El jugador ha de sortir del joc. | |
| Flux principal | 1. Comprovar quina partida tria el jugador. 2. Comprovar si la partida ja estava creada o és una nova partida. 3. Si interactua amb un NPC, mostra un menú. 4. Comprova quin nivell vol jugar 5. Si la vida del jugador arriba a 0, teletransporta al jugador fins a l'últim respawn. 6. Si el jugador prem la tecla escape, pausa el joc. 7. Si prem sortir, s'acaba el joc. | |
| Flux alternatiu | 1. Si la partida ja està creada, col·loca al jugador en el poble central. 2. Si la partida és nova, carrega el tutorial. | |

### Diagrames d’activitat

Diagrama dels menús

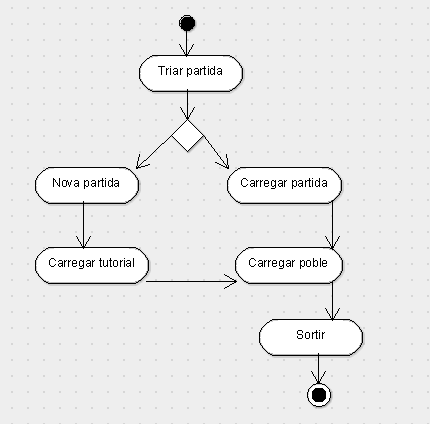
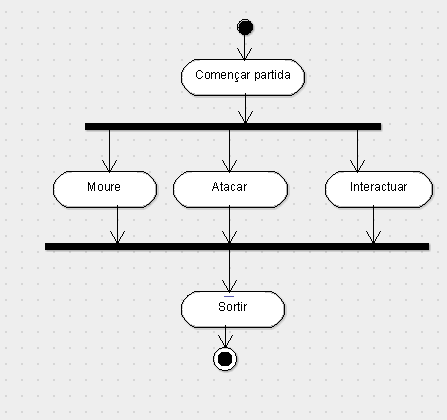
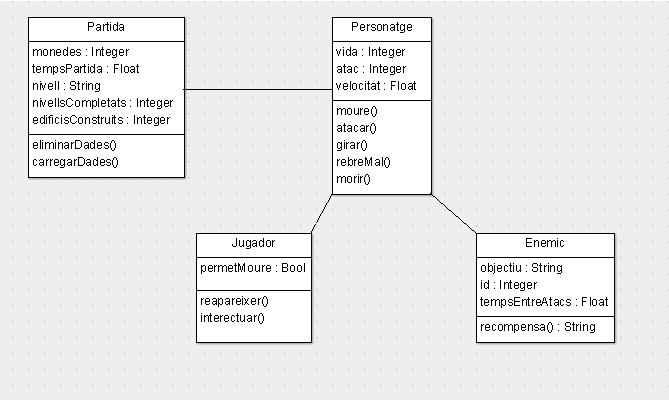


Diagrama del joc

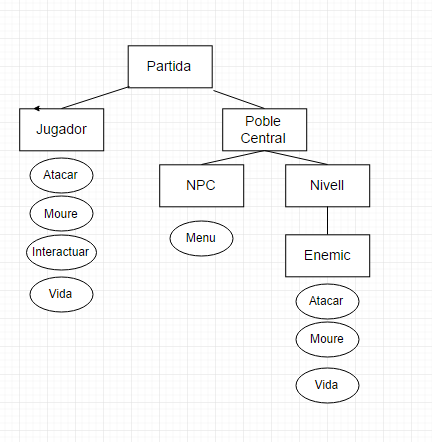


## Disseny

### Diagrama de classes

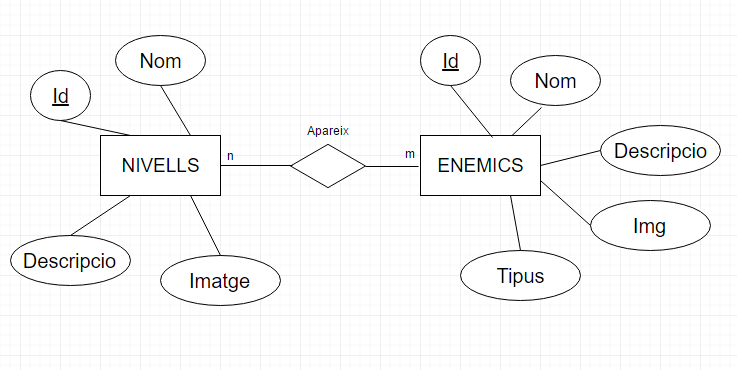


### Disseny modular del projecte



## Estructura de les dades

### Model entitat relació de la BD



### Model relacional

Enemics(Id,Nom,Descripcio, Img, Tipus)

Nivells(Id,Nom,Descripcio, Img)

Nivells\_Enemics(IdEnemics, IdNivells)

### Acces a la base de dades

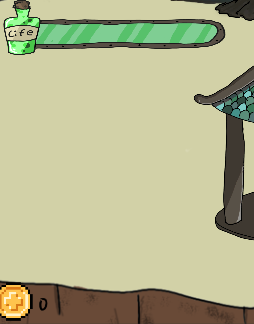
Per accedir a les dades desde Unity, hem creat la base de dades amb Bd Browser for SQLite. Per que Unity implementi la base de dades quan es crea l’executable, cal posar la base de dades a una carpeta anomenada StreamingAssets.

## Disseny de les interfícies

### Descripció de les interfícies

Les Interfícies de l'usuari utilitzen un dibuix simple i poques dades per no sobrecarregar la pantalla amb molta informació. Totes les interfícies s'han posat perquè s'adaptin a la resolució de la pantalla.

Durant el joc l'únic que sortirà és la barra de vida i les monedes obtingudes.

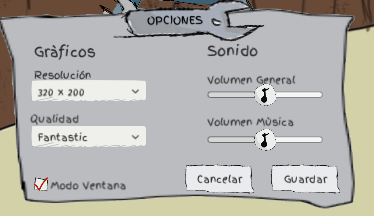


Sobre cada enemic apareix una mini barra que representa la seva vida.

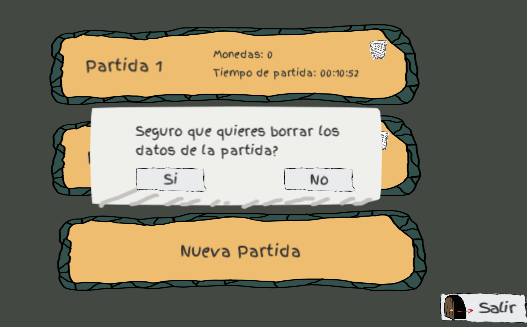


Quan pots interaccionar amb algun objecte, surt un diàleg sobre el cap del personatge principal mostrant l'acció que s'executarà.

Dintre del poble hi ha variïs NPC's que obren menús amb informació o opcions a editar.

Hem separat totes les opcions en blocs per saber que s'està editant en cada moment amb facilitat. Hem volgut simplificar-ho tot el màxim possible.

Quan l’usuari vol executar alguna acció que pot ser algú irreparable, apareix un missatge per assegurar-se que l’usuari realment vol fer aquella acció.



# Aspectes de seguretat

## Accés a les dades

Les dades del videojoc, les guardem en un fitxer binari. Quan guarda partida, totes les dades es guarden en el fitxer de la partida triada. Per defecte, els fitxers es creen a la mateixa carpeta del videojoc. Per assegurar-nos que qualsevol persona trobi les dades fàcilment i les esborri o editi, hem canviat la ruta on guarda els fitxers.

Es guarden en una ruta diferent segons els dispositiu on es juga. En el cas de PC, es guarda a: C:\Users\Toni\AppData\LocalLow\'CompanyName'\'GameName'\

La base de dades l'hem creat amb SQLite i accedim amb Unity per mostrar les dades en el videojoc. Com hem après a l'assignatura de M06, hem creat un fitxer a part per accedir a la base de dades, així el codi del joc no ha d'utilitzar comandes per accedir la base de dades. Per cada taula, hem creat un arxiu amb els atributs de la taula per passar les dades entre scripts de manera fàcil.

El problema és que no podem canviar la carpeta on està col·locada i és molt fàcil accedir-hi i canviar les dades.

Per això, hem volgut xifrar la base de dades mitjançant DB Browser for SQLite. Unity no permet connectar-se a una base de dades xifrada, així que hem buscat solucions creades pels usuaris a GitHub.

Entre molta cerca i anar provant, vam trobar un pluguin anomenat SqlCipher4Unity3D. És molt fàcil aplicar-ho al projecte, però al principi és molt difícil d'entendre. Hem hagut de canviar tot el fitxer per accedir a la base de dades i intentar entendre com funcionaven les comandes.

SqlCipher4Unity3D s'aplica de manera fàcil i és més o menys fàcil d'entendre, però al ser un pluguin creat per un usuari, si sorgeix algun problema, no hi ha molta gent que sàpiga quin és l'error.

Volíem afegir "Try & catch" cada vegada que accedia a cada dada i escriure l'excepció en un fitxer log a la carpeta del videojoc. Però no ha sigut necessari, ja que Unity ja controla les excepcions i crea el seu propi fitxer log amb tota la informació necessària.

# Explicació procés

## Animacions

### Personatges

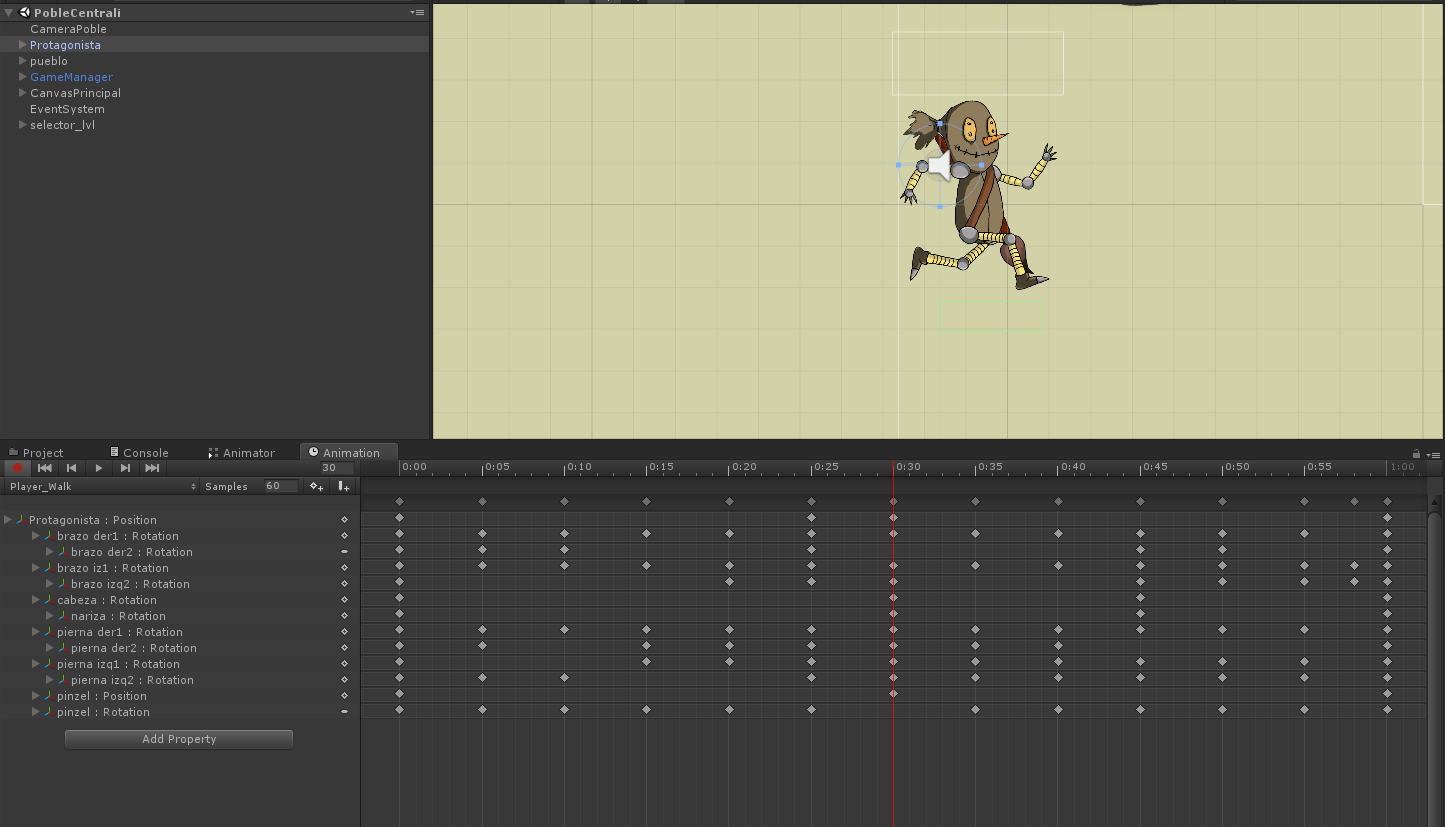
Hi ha diverses maneres d'animar els personatges. Nosaltres hem decidit animar-ho separant els personatges per parts.

Primer s'han de dibuixar totes les parts dels personatges separades perquè Unity pugui agafar cada part individualment



Una vegada tenim totes les parts, les col·loquem per crear el personatge principal.

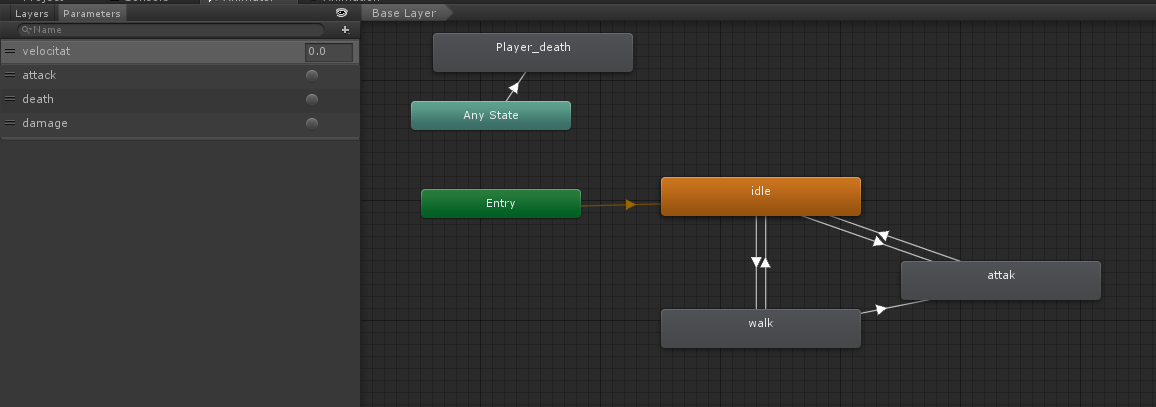
Una vegada preparat el personatge, hem de pensar quines són les animacions que volem pel personatge i crear una animació per cada un. En el nostre cas, hem animat: Quiet, caminar, atacar i morir.

Per animar hem creat un fotograma clau a cada segon i anar moguen les parts perquè quadri tot.

Una vegada hem creat totes les animacions necessàries, les programem.

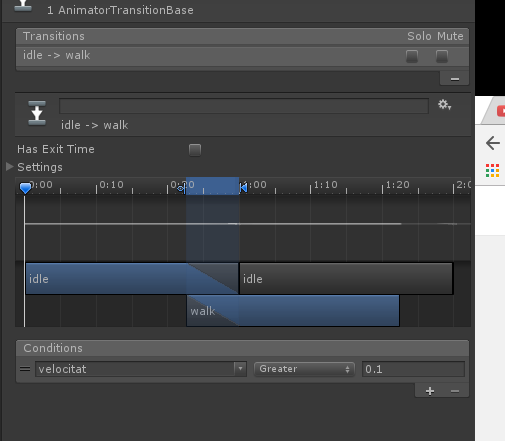
La finestra Animator conté totes les animacions creades dels personatges i permet configurar quan s'executarà cada animació.

Hem creat uns paràmetres que canviarem via codi per canviar d'animació. Per exemple, hem creat un paràmetre float que guardarà la velocitat del personatge. Si la velocitat és superior a 0 s'executarà l'animació de caminar, si és inferior a 0, l'animació de quedar-se quiet.



L'animació de color taronja és la que s'executarà en iniciar. El camp 'Any State' permet activar l'animació assignada des de qualsevol animació. En el nostre cas hem posat que s'executi l'animació de morir.

Per millorar l'efecte de les animacions, hi ha opcions avançades per canviar el temps de la transició entre animacions. Nosaltres hem canviat els temps en algunes animacions perquè l'animació quedes més fluida.



### Objectes

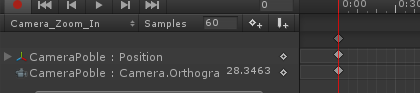
En el cas d'animar objectes, no hem pogut separar-ho per parts, ja que el canvi era mínim i sense moviment. Per poder animar aquests objectes, primer hem dibuixat cada fotograma de l'animació.

Després només s'ha de col·locar cada fotograma en cada segon que volem que s'executi i obrir la finestra Animator per configurar-ho.

### Efectes

En alguns casos, hem volgut animar només el moviment o la mida de certs objectes. Per exemple, un efecte de zoom amb la càmera en entrar a cada nivell.

Per poder animar aquest tipus d'objectes, només s'ha d'afegir quines propietats es canviaran i preparar cada fotograma

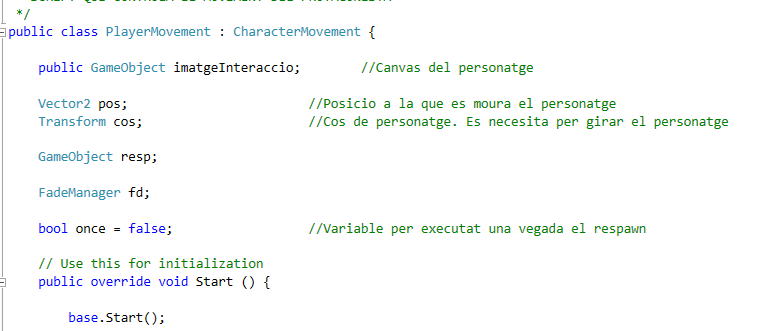


## Programació

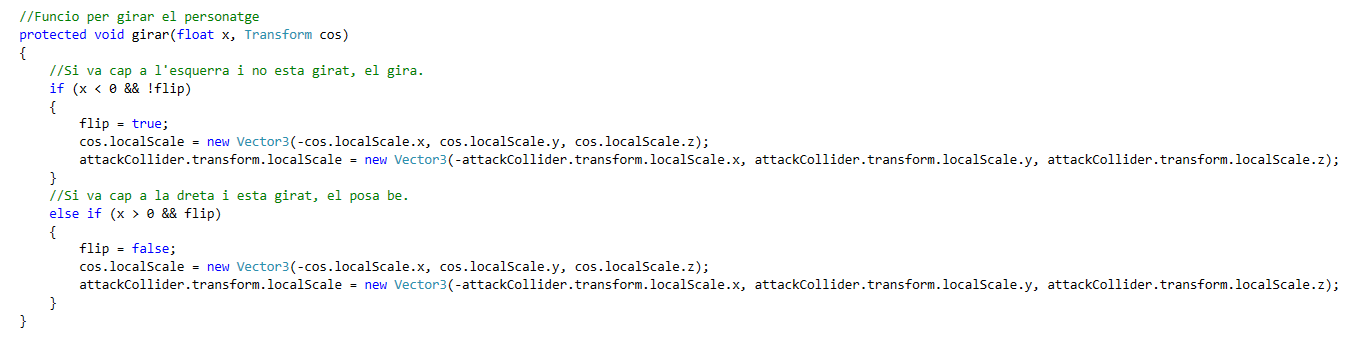
### Personatges

Hi ha variïs mètodes que cada personatge utilitza. Per no repetir el codi d'aquest metodes (girar, atacar,moure, rebre,morir), hem creat una classe base anomenada CharacterMovement i tots els scripts de cada personatge utilitzar herència amb aquest script.

Per alguns mètodes com Start() hem hagut de sobreescriure per poder utilitzar-ho amb els fills.

Base.Start() executa el mètode Start() del pare.

El mateix passa amb els scripts dels enemics. Cada enemic utilitza herència amb un script base anomenat EnemyMovement.

Quan un personatge va cap a una direcció, s'ha de girar la imatge perquè l'animació de caminar es vegi correctament. Per poder fer això, hem creat un mètode on restem la posició actual amb la posició que et mous en el següent frame. Si el resultat és negatiu, vol dir que caminarà cap a l'esquerra i si és positiu, cap a la dreta.

Atacar amb un personatge no és simplement activar l'animació d'atacar. Hem controlat que quan ataqui, el personatge es pari de moure. Si l'animació d'atacar s'està executant, prohibim que el jugador es pugui moure o que torni a atacar fins que l'animació acabi.

Per poder fer mal, hem creat un àrea a la posició on fa l'atac. Aquesta àrea s'ha d'activar quan l'animació doni el cop. Per fer això, hem creat un event a l'animació que executa un mètode que activa i desactiva l'àrea.

Tant el jugador com els enemics, han d'executar una animació en rebre mal o morir. Per això hem creat un mètode una vegada el cop t'ha tret la vida, comprova si encara estàs viu i executa una animació o altra.

### Jugador

El jugador ha de reviure cada vegada que mor. Per això hem creat un objecte invisible al principi del nivell i quan el personatge mor, apareix a la posició d'aquest.

Hem creat punts de controls que al passar per davant, mouen l'objecte per reaparèixer a la posició del punt de control.

### Enemics

Tots els enemics simples els hem programat de la mateixa manera a l'hora d'atacat al personatge. Si el jugador està viu i està a cert rang de l'enemic, es mou fins davant del jugador i l'ataca.

Per l'enemic final hem creat un script amb un mètode per cada atac. Després es crea un número aleatori i depenent del número, executa un mètode o altre.

Per cada atac, creem un comptador que va disminuint cada segon. Si al executar una atac el seu comptador no està a 0, no ataca.

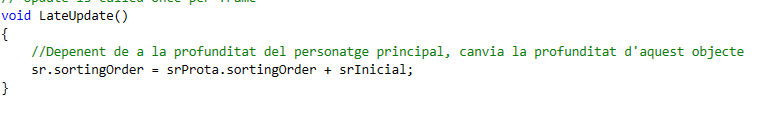
### Profunditat

La vista del videojoc és 2.5D, així que ha de donar un efecte de profunditat perquè es pugui veure bé. Tots els sprites tenen un component que es diu 'Sorting Order'. Aquest component aplica l'orde en què es veuran els personatges, quant mes numero, mes per endavant estarà.

Per poder aplicar aquest efecte agafem la posició en la Y de tots els personatges i objectes del joc. Passem la Y a negatiu i la posem com a Sorting order del objecte. Això vol que quan mes a baix estigui el personatge, mes endavant es mostrarà.



Un dels principals problemes que hem tingut és que en ser un personatge separat per parts, cada part té el seu propi Sorting Order. Si posem a cada part l'script encarregat de la profunditat, a vegades un braç està per sobre o una cama per darrere.

Per evitar això hem hagut de crear un script per les parts. Cada part agafa el sorting order del pare i li suma el sorting order inicial de la part. Això fa que l'efecte s'apliqui bé.



Això soluciona l'error si un objecte separat amb parts i un altre que no es troben. Però si dos personatges separats per parts es troben, les parts es col·loquen unes per sobre d'altre deixar un efecte poc realista.

Aquest error és un dels errors que no hem pogut solucionar.



### Opcions

Per programar el menù d’opcions, hem carregat totes les resolucions i qualitats disponibles en aquell ordinador als ComboBox corresponents. Quan el valor del ComboBox canvia(OnValueChanged), la resolució/qualitat es canvia amb ell.

Per canviar el volum general, Unity permet baixar el volum de tota la musica del joc amb una linea.

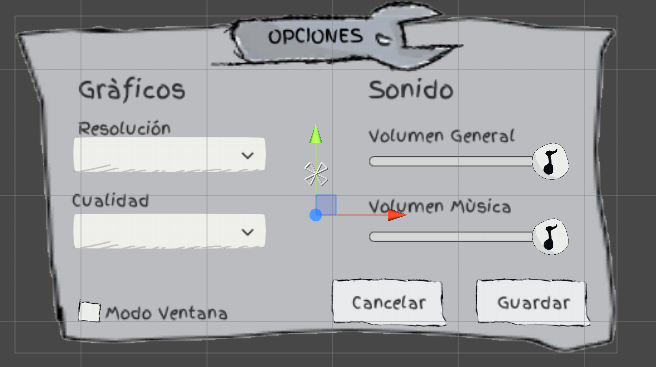


Per baixar nomes el volum de la musica de fons, agafem per referencia l’audio de la musica de la escena, ubicada a la camara.

A l’apretar el botò Cancelar, tots els canvis fets es perden i es renicien els ComboBox i barres.

A l’obrir el menu, es carrega el volum que tens guardat a la partida, que es com ho havies deixat l’ultima vegada. Per defecte esta al maxim.

Per la resolucio i la qualitat nomes carreguem l’actual, ja que no tè sentit guardar-ho si a l’iniciar el joc, et deixen canviar-ho.

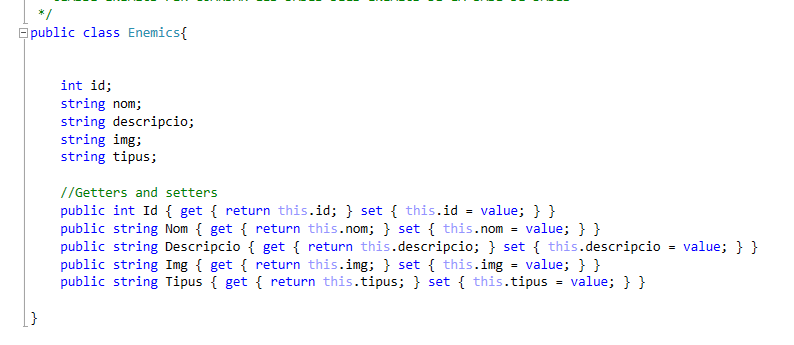


### Bade de Dades

Per accedir a la base de dades hem creat un fitxer amb totes les funcions que accedeixen a la base de dades.



I un fitxer per cada taula de la base de dades amb les variables de les columnes.



Hem creat un boto Prefab que carrega les dades de l’enemic a la pagina del a dreta. Quan s’obre el llibre, executa un select de tots els enemics i els compara amb el fitxer que guarda partida. Si l’enemic ha sigut eliminat, creem el prefab dintre d’un Scroll.



Hem afegit eines de cerca, on utilitzem comandes SQL com ‘LIKE’ i ‘WHERE’ per trobar els enemics filtrats.

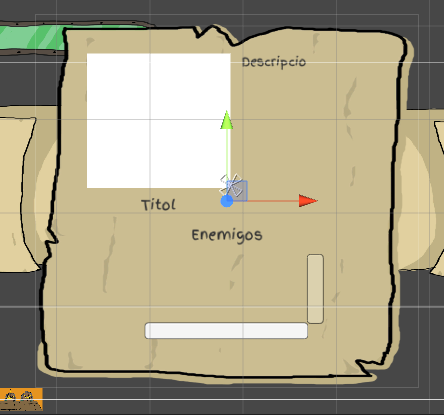


Cada vegada que el ComboBox canvia de posició o l’EditText canvia de text, es tornet a carregar les dades dels enemics amb el filtre posat.

Utilitzem el mateix metode per els mapes.



Quan prems un nivell, s’executa una comanda SQL que recull la informació del nivell i la mostra en una plantilla.

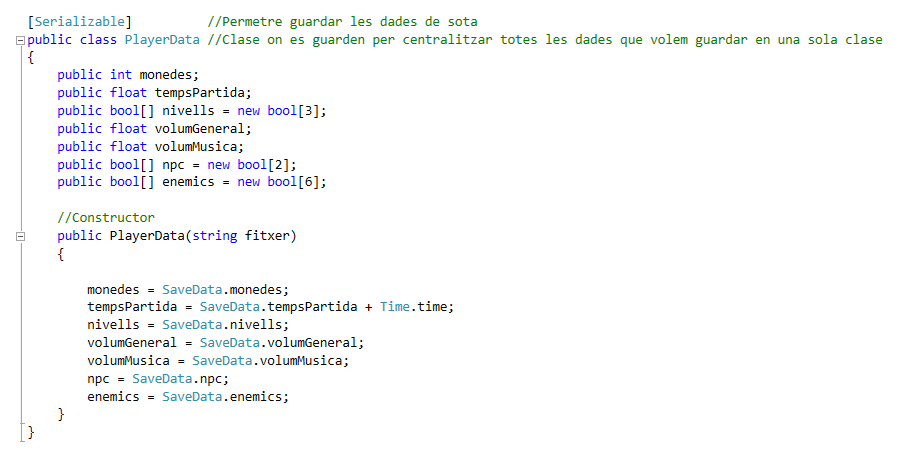


### Guardar partida

Hi han varies maneres recomenades de guardar partida amb Unity. Nosaltres hem escullit utilitzar FileStream per guardar les dades en un fitxer binari.

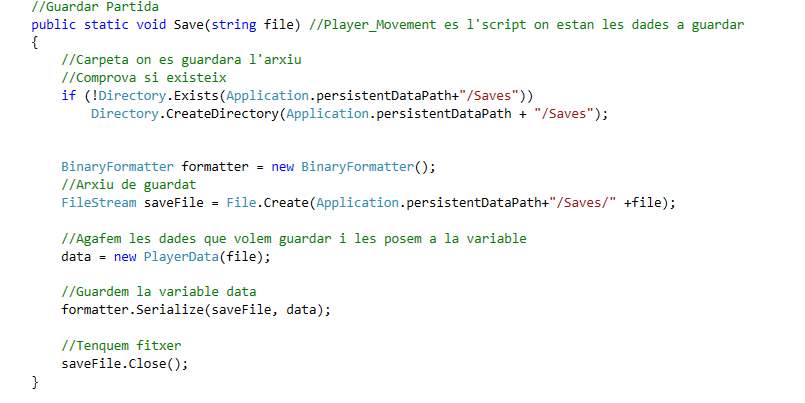
Hem creat un fitxer centrar en guardar i carregar partida, aixi els altres scripts no necesiten tocar res relacionar amb llegir/escriure fitxers.

Per poder guardar totes les dades del joc facilment, hem creat una classe amb totes les variables que volem guardar.



El camp Serializable fa que la classe es pugui guardar en un fitxer.

Despres nomes hem de escriu el fitxer que volem guardar depenent de la partida, i guardar la classe.



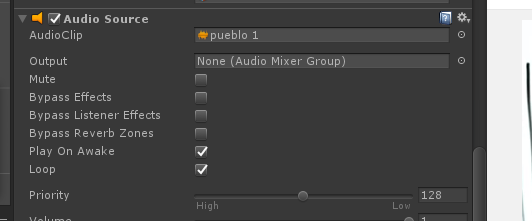
Al carregar les dades, es retorna una classe del mateix tipus que guardem les dades, i desde l’script on hem executat el metode nomes hem de guardar la classe i agafar les dades.

## Musica

### Fons

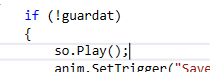
Per posar la musica de fons es molt simple. Nomes cal afegir el component ‘AudioSource’ a on objecte de la escena i arrosegar l’audio. En el nostre cas ho hem posat a la camera.

De les opcions disponibles nomes cal marcar ‘Play On Awake’ per que soni desde el prinipi i ‘Loop’ per que continui la canço sempre.



### Sons

Per posar sons hem de arrosegar un ‘AudioSource’ pero sense marcar cap opció. Despres via codi, programem quan volem que s’executi el so.



# Conclusions

Un cop hem acabat el videojoc, hem tret les nostres pròpies conclusions.

Quan vam decidir de que aniria el nostre projecte ens vam motivar molt, ja que teníem un munt d’idees, i moltes d’aquestes les vam haver de suprimir, ja que cada cop veiem que faltava menys temps per acabar i el rellotge anava en contra nostre, tot i així creiem que el resultat és bo.

Això sí, veure des de zero com les teves idees cobren vida no té preu, ja que a partir d'una petita idea extreta de la nostra imaginació hem pogut crear tot un món del qual n'estem molt orgullosos. Però, no tot és bo, ja que amb un sol mes com ja he explicat, no hem tingut el suficient temps per a introduir totes les grans idees que ens rondaven i que encara ens ronden pel cap. Així que moltes d’aquestes les hem hagut d'adaptar-les d'una manera més 'light', ja que no requeríem ni de temps ni de les habilitats necessàries per realitzar-les. Perquè la seva realització fos possible, gran part del temps l'haguéssim hagut de redirigir-lo en cercar per internet la solució. Així que vam decidir fer una mica de cada, encara que es desviés una mica de la idea principal.

Cal esmentar que una part del temps el vam destinar en crear els nostres propis dissenys. El motiu d’això és el següent; Si ens posem a pensar en un futur com a programadors de videojocs ens podem trobar en el cas de no comptar amb cap company que sigui un dissenyador (Concept Artist), cosa que és fonamental si vols crear el teu propi videojoc des de zero. Ja que no només és programació, hi ha altres factors com el disseny, que hem hagut d’assolir per el nostre propi compte. Per això, crear els nostres propis dissenys ens a servit de cares al futur si mai ens faltes personal artístic.

Quan parlem de disseny també i podem afegir el saber controlar cada animació d’un personatge, objecte, o inclús el mapa. Òbviament hi ha moltes més coses que necessiten d’animació, però aquestes 3 son molt importants, ja que fan que el joc cobri vida.

# Bibliografia

Informació sobre codi i funcionament de Unity.

<https://docs.unity3d.com/ScriptReference/>

Tutorials per saber animar els personatges amb Unity.

<https://www.youtube.com/watch?v=OoQo2MmvwXw>

<https://www.youtube.com/watch?v=JNqpKb3ocNg>

Tutorials per agafar de guia per programar en 2D.

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

Connectar Unity a una base de dades SQLite

<http://answers.unity3d.com/questions/743400/database-sqlite-setup-for-unity.html>

<https://github.com/netpyoung/SqlCipher4Unity3D>

# Comentari personal

El món dels videojocs és impressionant. Si des de fora ja ho veiem atractiu, un cop ets a dintre i et poses a realitzar-lo pel teu propi compte, veus que és tot un món immens de possibilitats, el qual ens agradaria explorar-lo en un futur no molt llunyà.

No sabem si després d'acabar el projecte el seguirem, però del que si estem segurs és que de cares al futur tenim pensat crear el nostre propi videojoc.

Creiem que aquest mes de cerca a facilitat la introducció al Unity, però no per controlar-lo a la perfecció, això requereix més temps, ja que Unity té moltes opcions. Tot i que, podem dir obertament que Unity no és un programari difícil d'utilitzar, al principi ho pot semblar, però sàpiguen el llenguatge bàsic de C# i una mica de cerca per internet pots realitzar treballs com el nostre, o el dels nostres companys.