



Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

**Facultat de Matemàtiques
Universitat de Barcelona**

Joc isomètric RPG amb Unity

Víctor Novo Rodríguez

Director: Oriol Pujol Vila
Realitzat a: Departament de Matemàtica
Aplicada i Anàlisi. UB
Barcelona, 20 de juny de 2014

RESUM

Des de fa molts anys soc una aficionat als videojocs. En un moment donat hem vaig preguntar com es possible crear diferents entorns tan similars al mon reial i que sembli que estàs veien una pel·lícula alhora que ho estàs passant be jugant. Per aquest motiu vaig decidir-me embarcar en aquest projecte.

En aquest projecte he dissenyat i desenvolupat gran part dels models utilitzats en el joc, així com les diferents animacions utilitzades per aquests models mitjançant Blender, un programa de modelatge, i he desenvolupat les diferents accions i interaccions dels personatges utilitzats en el joc mitjançant el motor gràfic Unity3D.

El tipus de joc que vaig escollir finalment es un RPG amb vista isomètrica, basat en un sistema de combats per torns utilitzant regles del popular joc de rol D&D. Aquest joc esta ambientat en un mon fantàstic i medieval on es controlen diferents personatges alhora amb diferents rols on es necessària la col·laboració entre ells per arribar al objectiu final del joc.

En les diferents seccions d'aquest document s'explica el desenvolupament tant a nivell de modelatge com de programació d'aquest projecte.

ABSTRACT

It has been many years since I have been a video game fanatic. I have often wonder how would it be possible to create so many scenarios that are similar to the real world in a way that it would seem that you are watching a film while having fun playing. This was the reason why I decided to take this project.

In this project I have designed and developed big part of the models used in the game, like for instance the different animations which was possible by using Blender, a modelling program. The use of Unity3D has allowed me to develop the different actions and interactions of the characters in the game.

The type of videogame that I finally chose was an isometric view RPG based on a 'turn-based combat' system using the rules of the popular role game D&D. This game is set in a medieval fantasy world in which the different characters with different roles, controlled at the same time, cooperate with each other to accomplish the final objective of the game.

The modelling and programming development is explained in the various sections of this document.

Índex

1.	Introducció i motivació	6
1.1.	Introducció	6
1.2.	Motivació	6
2.	Objectius.....	7
2.1.	Introducció	7
2.2.	Objectius Generals	7
3.	Anàlisi.....	8
3.1.	Escenari.....	8
3.2.	Personatges principals.....	10
3.3.	Enemics.....	11
3.4.	Càmera Isomètrica	12
3.5.	Hud.....	13
3.6.	Menús.....	13
3.7.	Combats	14
3.8.	Argument.....	15
4.	Disseny.....	16
4.1.	L'escenari	16
4.1.1.	El Terreny	16
4.1.2.	L'arquitectura	16
4.1.3.	Il·luminació.....	17
4.1.4.	Sistema de partícules	18
4.2.	Els Personatges.....	18
4.2.1.	Moviment y Controls dels personatges principals	20
4.2.2.	Els enemics	22
4.2.3.	Animacions dels personatges	22
4.3.	Menús.....	23
4.4.	Hud.....	24
4.5.	Habilitats y Encanteris	26
4.6.	Combats	29
4.7.	La Càmera.....	31
4.8.	Fi del joc i morts.....	31
5.	Implementació	33
5.1.	Escenari.....	33
5.1.1.	Terreny	33
5.1.2.	L'arquitectura	35

5.2.	Creació dels personatges	47
5.2.1.	L'esquelet(Armature)	48
5.2.2.	Creació d'una animació	53
5.2.3.	Importació dels models creats a Blender	56
5.3.	Utilització dels objectes a unity	59
5.3.1.	Components utilitzats.....	60
5.4.	Funcionament dels GameObjects al joc.....	65
5.4.1.	Escenari Principal	65
5.4.2.	Inici del joc.....	67
5.4.3.	Moviment y Controls dels personatges principals	67
5.4.4.	Els enemics	73
5.4.5.	Encantaments y Habilitats	75
5.4.6.	El Hud.....	78
5.4.7.	Combats	81
5.4.8.	La càmera.....	84
5.5.	Menús.....	85
5.5.1.	Menú Principal	85
5.5.2.	Menú de pausa	87
5.6.	Fi del joc y morts.....	88
6.	Proves i resultats	89
6.1.	L'escenari	89
6.2.	Personatges	92
6.2.1.	Personatges principals.....	92
6.2.1.1.	Els personatges al joc.....	92
6.2.1.2.	Accions dels personatges al joc.....	93
6.2.1.3.	Interacció dels personatges amb l'escenari	97
6.2.2.	Els enemics	99
6.2.2.1.	Accions des enemics al joc.....	99
6.2.3.	El personatge Neutral	101
6.3.	El joc.....	102
6.3.1.	Inici del joc.....	102
6.3.2.	Èxit al joc	102
6.3.3.	Joc Fallit	103
7.	Conclusions i treball futur.....	104
8.	Referències Bibliogràfiques	104
9.	Manual de usuari	105
10.	Glossari	109

1. Introducció i motivació

1.1. Introducció

Fa ja dècades que existeixen els videojocs. Aquests jocs van néixer amb la intenció d'implementar programes de caràcter lúdic, on l'usuari pot controlar un o varis personatges per aconseguir un determinat objectiu mitjançant unes regles, recreant situacions i entorns virtuals. Els primers intents no van tardar en aparèixer i han anat sorgint fins arribar als nostres dies.

Els primers videojocs van aparèixer a la dècada dels 60 i aquest món no ha parat de créixer amb l'únic límit que ha imposat la creativitat dels desenvolupadors i l'evolució de la tecnologia. En els últims vint anys han evolucionat molt, gràcies a l'aparició de hardware potent capaç de processar grans volums de dades a alta velocitat. Aquestes evolucions han permès millorar els videojocs en aspectes tals com els gràfics, la jugabilitat o l'intel·ligència artificial entre d'altres.

Paral·lelament a la evolució dels videojocs han sorgit diferents plataformes on les persones podem gaudir de la experiència de joc, des de fa ja bastants anys a les videoconsoles, però des de fa relativament poc fins i tots als *smartphones* amb resultats sorprenents.

Gràcies a la imaginació dels creadors dels videojocs per tal d'explicar una història i a que avui en dia s'ha arribat a tal punt de realisme que sembla que mentre estàs jugant estiguis mirant una pel·lícula. Tot un art.

1.2. Motivació

Des de ben petit he sigut un aficionat als videojocs. En algun moment de la meua vida hem vaig preguntar com es poden crear videojocs tant realistes com la saga *Final Fantasy*, amb una bona història com la saga *Metal Gear* o amb una bona jugabilitat com la saga *Legend of Zelda*, entre d'altres.

També he sigut un aficionat als jocs de Rol, on amb imaginació i seguint unes regles especificades per aquest joc pots passar una bona estona jugant. Per aquesta raó em vaig decidir a crear un joc RPG. En aquest tipus de jocs controles un grup de personatges que tenen diferents habilitats ben diferenciades, on la cooperació entre aquests personatges es essencial per assolir els diferents objectius del joc.

En aquest projecte m'he centrat en el desenvolupament i disseny dels diferents objectes del joc i en la programació per interactuar amb l'entorn, la intel·ligència dels enemics i els efectes visuals.

2. Objectius

2.1. Introducció

En aquest projecte he realitzat una iniciació al món de la creació de videojocs desenvolupant la demo d'un joc RPG amb vista isomètrica, on un equip de tres personatges, explora un escenari on interactuen amb alguns objectes d'aquest entorn, i on l'objectiu final es trobar una espasa que apareixerà màgicament després de derrotar els diferents enemics del mapa.

Per a poder assolir l'objectiu de la creació d'aquesta demostració he utilitzat el motor gràfic Unity3D amb llenguatge de programació C#. Per tal de modelar i animar els diferents objectes que componen el joc he utilitzat Blender. Unity és un motor gràfic molt potent, i relativament jove, cal dir, que he utilitzat la versió lliure per a finalitats didàctiques, que disposa de menys eines que la versió professional.

2.2. Objectius Generals

El joc disposarà de tres personatges principals, cadascun d'ells amb un rol diferent, controlats per ratolí, tant el moviment com les diferents habilitats de les que disposen. Utilitzarem una càmera isomètrica que enfocarà a cada un dels personatges segons el personatge que estigui seleccionat. Disposarem també d'un conjunt d'enemics que utilitzarà una intel·ligència artificial. El jugador podrà explorar un escenari mitjançant els personatges seleccionats i lluitar contra els enemics, durant l'exploració i els diferents combats que sorgeixin podrem detenir el temps del joc per pensar les diferents estratègies per tal de poder guanyar les nostres batalles. Aquest escenari és una cripta. El jugador també disposarà d'un menú principal i un menú de pausa.

Detalladament els objectius finals d'aquest projecte són els següents:

- Mitjançant Blender es realitzaran les següents tasques:
 - Modelar l'arquitectura del escenari on es realitzaran els diferents events del joc.
 - Creació dels esquelets dels personatges jugables i dels enemics.
 - Creació de les animacions dels personatges jugables i dels enemics.
- Mitjançant Unity i el seu editor de modelatge es realitzaran les següents tasques:
 - Creació del *terrain* on es mouran els diferents personatges del joc.

- Creació del menú principal i els *scripts* necessaris per donar-li un comportament adequat.
- Programació del *script* de la càmera per donar-li el comportament desitjat.
- Programació dels diferents *scripts* per moure i realitzar les diferents accions i comportaments dels personatges principals en funció de les entrades per teclat o per ratolí.
- Creació i programació del *script* que s'encarrega de la HUD del joc.
- Programació del *script* de la IA dels enemics.
- Programació del comportament i gestió de les lluites entre enemics i personatges principals
- Programació dels *scripts* que realitzen el comportament de diferents elements del escenari.

Cal comentar que els personatges principals, són models ja creats, descarregats de la xarxa, així com alguns elements del escenari com les estàtues. Els enemics són models descarregats del *asset Store* de Unity i de la web.

3. Anàlisi

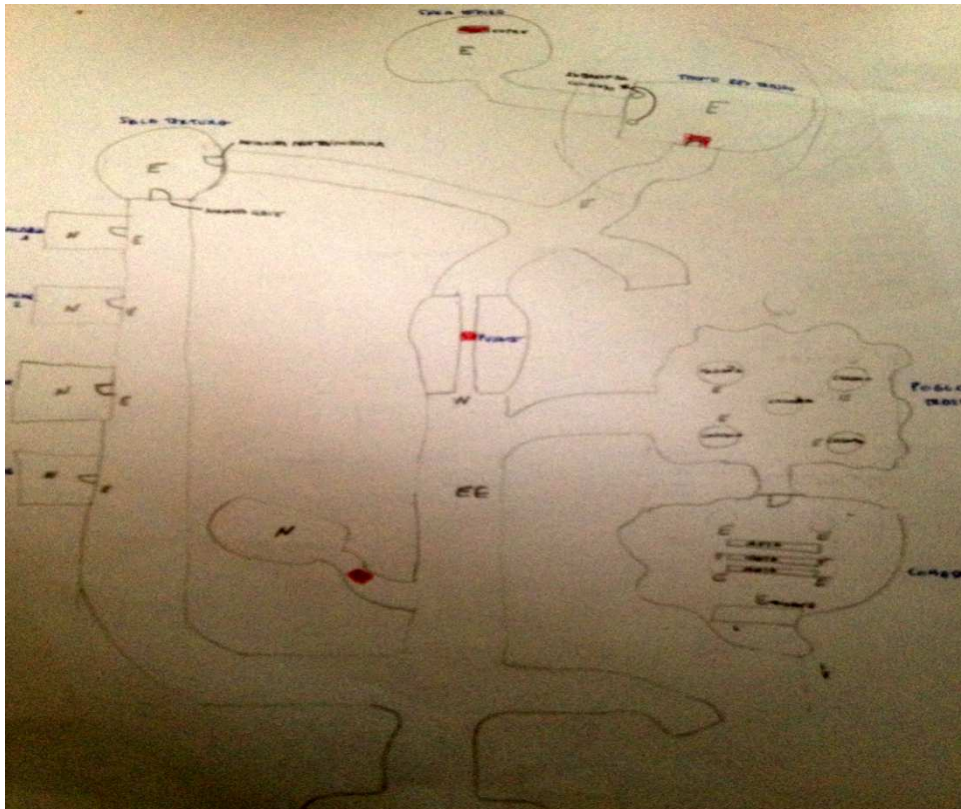
3.1. Escenari

L'escenari on es desenvolupa el joc ha estat creat majoritàriament amb Blender pel que fa a la arquitectura de la cripta, excepte el terreny que s'ha creat amb Unity gracies a l'eina *terrain* que disposa *l'engine* per a crear terrenys.

En un principi es va pensar en una ampla cova com a escenari del joc. Els personatges haurien d'explorar-lo fins arribar a la zona on estaria situat el *boss* final. Per arribar-hi es podria optar per rutes alternatives, aquestes rutes consistiria en petits *puzzles* com agafar una determinada clau situada a un lloc del mapa per obrir una determinada porta, o be interactuar amb determinats personatges del mapa per que et donessin informació sobre com arribar-hi al teu destí, etc.

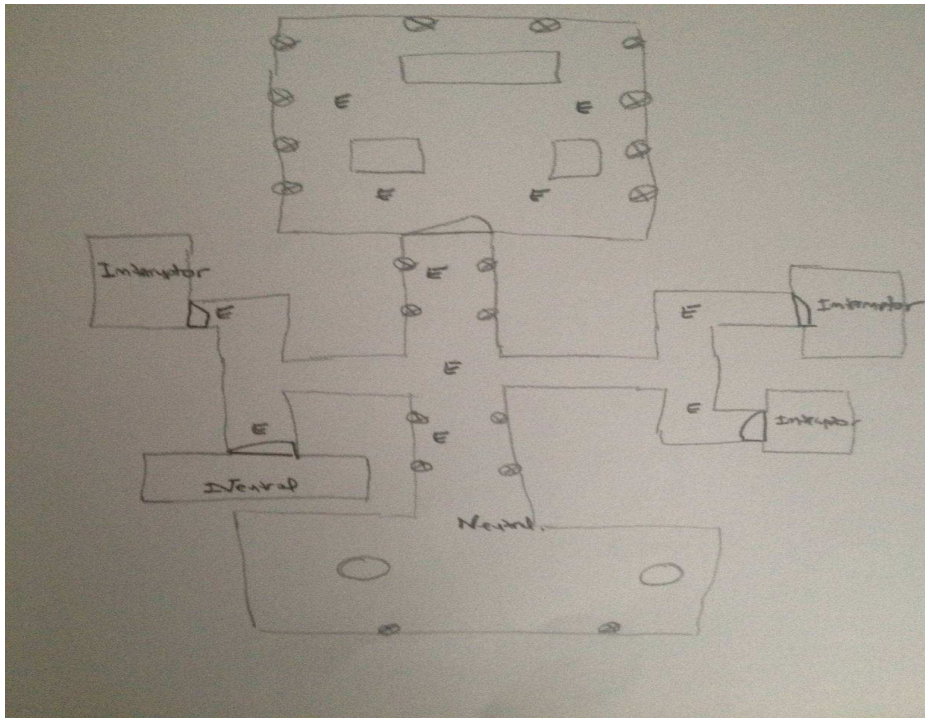
En la imatge 1 es pot veure l'escenari imaginat en un principi. En la part sud de la imatge es situaria l'entrada on s'iniciaria el joc, en la zona esquerra estarien situades les masmorres i una sala on hauria una porta que sols s'obris des d'aquesta sala després de derrotar a un enemic, en la zona central existiria un pont on un personatge t'impedís passar si no li donessis un objecte especial ubicat a la zona de la dreta, zona on hi hauria el poblat dels enemics, amunt estaria el cap dels

enemics. les zones vermelles serien trampes que un dels personatges hauria de desactivar per que no baixes vida al equip.



Imatge 1: Escenari inicial imaginat

Aquesta primera idea va ser substituïda pel mapa d'una cripta, on els personatges jugables estarien ubicats inicialment a la part sud del mapa i haurien d'obrir la porta situada a la part de dalt del mapa per poder aconseguir la recompensa després d'haver eliminat a tots els enemics. Aquesta porta s'obriria activant els diferents interruptors situats en les altres zones del mapa. Un parell de personatges neutrals donarien certa informació a l'usuari sobre la ubicació dels interruptors. A la imatge 2 es pot veure el segon escenari imaginat.



Imatge 2: Escenari de la cripta imaginat

3.2. Personatges principals

Ja que l'objectiu es fer un joc RPG del tipus *Baldur's Gate*, on es necessari la col·laboració de diferents personatges amb diferents habilitats, aquests personatges disposen de diferents característiques com son la força, destresa, intel·ligència, saviesa i carisma i habilitats com veure, buscar amb diferents puntuacions segons el personatge i amb diferents habilitats de combat segons el seu rol.

Amb aquest objectiu m'he decantat per un personatge que tingui atacs de curt abast com a característica predominant la força, aquest personatge es un guerrer; un mag capaç de fer encantaments tant defensius com d'atac. Aquests encanteris defensius pujaran les característiques del grup i els d'atac baixaran la vida als enemics. Aquest personatge també podrà atacar a curta distancia però no serà tan efectiu com el guerrer i per finalitzar un personatge de llarga distancia , un arquer, on la destresa serà el seu punt fort, així con la característica de veure, aquest personatge en un principi seria capaç de detectar trampes del escenari, però aquesta funcionalitat no ha sigut implementada. A la imatge 3 es pot veure un equip de personatges.



Imatge 3: Equip de personatges de diferents classes. Imatge de Never Winter Nights 2

Els personatges principals podran caminar, atacar als enemics, utilitzar màgies i habilitats pròpies i activar interruptors.

Podrem utilitzar tots els personatges del equip o una part d'ells per a realitzar diferents accions, també es podran utilitzar de forma individual segons el personatge que es seleccioni.

3.3. Enemies

Els enemics a l'igual que els personatges principals també tindran unes característiques i habilitats, ja que aquestes característiques són necessàries pel correcte funcionament de les lluites. Disposarem de dos tipus d'enemics, de curta i llarga distància. Aquests enemics seran els *goblins* que estaran dispersats per l'escenari. Les funcionalitats de rebre una recompensa en forma punts d'experiència, o d'obtenir inventari del enemic una cop morts a sigut eliminada. Així doncs els personatges principals no augmentaran les seves característiques o habilitats això implica que no augmentaran de nivell. Com els personatges principals podran caminar, seguiran unes rutes predefinides, i podran atacar.



Imatge 4: Imatge de Goblins extreta del joc Never Winter Nights 2

3.4. Càmera Isomètrica

La càmera utilitzada serà una càmera isomètrica. A la gran majoria d'aquests tipus de joc s'utilitzen aquests tipus de càmeres. Aquesta càmera esta orientada a una posició i alçada fixa on es pot veure una determinada part del escenari segons el personatge escollit. A la Imatge 5 corresponent al videojoc Diablo 2 es pot veure com es vol visualitzar l'escenari.



Imatge 5: Visualització d'una càmera Isomètrica del joc Diablo 2

3.5. Hud

El Hud [*head-up display*] és el mètode per el qual l'usuari pot visualitzar la informació del joc. En el nostre cas, l'usuari podrà veure la vida dels personatges i dels enemics, els encantaments o habilitats que resten als personatges i la informació de les lluites. Podrà seleccionar a partir dels *portraits* els diferents personatges (aquesta selecció també es podrà fer polsant amb el ratolí sobre el model 3d del personatge), seleccionar els diferents encanteris o habilitats especials dels personatges, descansar i fer selecció múltiple. A la imatge 6 podem observar el Hud que es desitja.



Imatge 6: Hud utilitzat en el joc Baldur's Gate

3.6. Menús

Disposem de dos tipus de menús, l'inicial i el menú de pausa. El menú inicial consta de les opcions inici de joc, amb aquesta opció començarem la partida; opcions, on podrem seleccionar la dificultat del joc, hi haurà dues dificultats, fàcil i normal. Segons quina dificultat escollim augmentarà el nombre d'enemics en la partida. I la opció de sortir, on sortirem de l'aplicació.

En quant el menú de pausa, quan l'activem, el joc s'aturarà i podrem escollir l'opció de continuar joc, sortir de joc o sortir de la partida. Aquest menú s'activarà polsant la tecla "P".

3.7. Combats

Per tal de fer els combats, m'he basat en unes regles d'un joc de rol conegut anomenat *Dungeons & Dragons*. Aquests combats són per torns i es divideixen per etapes, que són tirada d'iniciativa, tirades d'atac i tirades de dany. Aquestes dues últimes etapes es repetiran fins que a algun dels personatges no li quedin PG(punts de cop^{*}), es a dir mori.

A la tirada d'iniciativa els personatges faran una tirada i tindrà el primer torn aquell que hagi fet la tirada més alta, es a dir serà el primer en atacar. Aquesta tirada d'iniciativa es farà un sol cop per combat.

La tirada d'atac consisteix en mirar si l'atac impacta contra al teu adversari, per fer això el personatge que tingui el torn, llançarà els daus, un dau de 20 cares, i se li sumarà a aquest resultat l'atac més el modificador de força. El personatge que rep l'atac farà una tirada de CA(classe d'armadura^{*}), aquesta tirada es el resultat de la suma de la CA base de tots els personatges que es 10, la CA de l'armadura i el modificador de destresa. Així doncs si l'atac es més gran que la CA, el personatge haurà tingut èxit en el seu atac.

El torn finalitzarà amb una tirada de dany, aquesta tirada restarà tants PG del oponent com el resultat de la tirada del personatge que te el torn. Aquesta tirada consisteix en fer una tirada segons els teu atac base i sumar-li el modificador de força del personatge. Una tirada d'atac base es una tirada on s'utilitza un dau de tantes cares com atac base.

En aquest projecte s'han fet unes modificacions en quant a la part dels combats, els torns s'han gestionat en funció de la destresa dels personatges, això vol dir que un personatge podrà tenir més torns d'atac que un altre personatge en el combat, com si de una cua de prioritats es tractes, on la prioritat be donada per la destresa. A la imatge 7 podem veure el codi d'una cua de prioritats implementat en Java.

* Consultar al glossari de vocabulari al final del document

```

public class ColaPrioridad<T> implements colaPrioridadInterface.ColaPrioridad {
    class Celda<T> {
        T elemento;
        int prioridad;
        Celda sig;
    }
    private Celda cola;
    public ColaPrioridad() {
        cola = new Celda();
        cola.sig = null;
    }
    public boolean vacia() {
        return (cola.sig==null);
    }
    public <T> primero() throws ColaVacíaException {
        if (vacía()) throw new ColaVacíaException();
        return cola.sig.elemento;
    }
    public int primero_prioridad() throws ColaVacíaException {
        if (vacía()) throw new ColaVacíaException();
        return cola.sig.prioridad;
    }
    public void inserta(T elemento, int prioridad) {
        Celda<T> p,q;
        boolean encontrado = false;
        p = cola;
        while((p.sig!=null)&&(!encontrado)) {
            if (p.sig.prioridad<prioridad)
                encontrado = true;
            else p = p.sig;
        }
        q = p.sig;
        p.sig = new Celda();
        p = p.sig;
        p.elemento = elemento;
        p.prioridad = prioridad;
        p.sig = q;
    }
    public void suprime() throws ColaVacíaException {
        if (vacía()) throw new ColaVacíaException();
        cola = cola.sig;
    }
} // fin clase ColaPrioridad

```

Imatge 7: Classe Cua de prioritats

També s'ha tingut en compte que pot hi haver-hi un combat múltiple, es a dir es pot donar el cas que si el personatge esta lluitant contra un enemic un altre personatge es pot afegir a la lluita, de la mateixa forma els enemics també es poden afegir als combats. Cada cop que això succeeixi es farà una nova tirada de iniciativa i es reiniciaran els torns dels combats.

3.8. Argument

El joc comença a la part sud del escenari, on hi seran els tres personatges, hauran de travessar el passadís per arribar a la sala on són els enemics, res més començar a caminar es trobaran amb un enemic que no els atacarà, avisarà als demés enemics per a que tanquin la porta per la qual es pot accedir a la sala final.

Per poder obrir la porta els personatges hauran d'accionar uns interruptors que son a uns panells que hi ha a les parets del passadís, D'això s'esventarà l'usuari ja que a l'entrar en contacte amb la

porta apareixerà un missatge informatiu si els personatges superen una tirada de habilitat que té a veure amb l'habilitat buscar.

Acte seguit la porta s'obrirà, al entrar a la sala els personatges hauran de derrotar als enemics per que apareix-hi l'espasa màgica. Si tots els membres del equip son eliminats apareixerà una pantalla de joc finalitzat i es tindrà que començar de nou des de el inici del escenari. Els enemics reapareixeran de nou.

4. Disseny

4.1. L'escenari

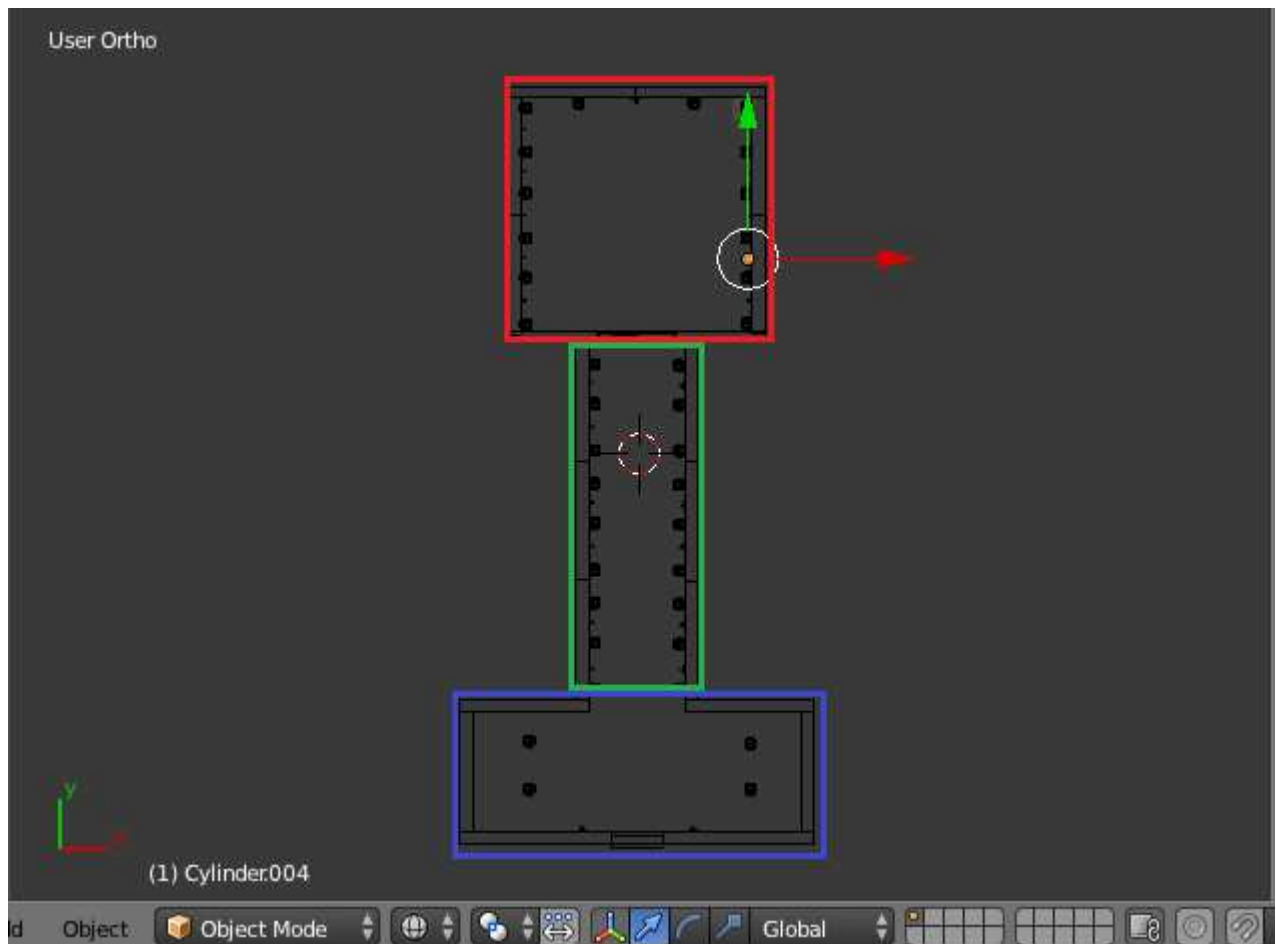
L'escenari s'ha creat utilitzant tant Blender com Unity, la creació del terreny ha sigut feta amb Unity i l'arquitectura de la cripta amb Blender que disposa de múltiples opcions per tal de modelar objectes.

4.1.1. El Terreny

Per crear el terreny del joc s'ha utilitzat l'editor de Unity, en concret l'eina *terrain* de que disposa el engine, aquesta eina disposa de opcions tals com la opció de pintat del terreny on es mouran els personatges, com l'aplicació de textures i opcions per ajustar la mida del terreny.

4.1.2. L'arquitectura

Per fer la l'arquitectura de la cripta he utilitzat Blender. L'arquitectura s'ha dissenyat utilitzant polígons basics tals com cubs i cilindres i utilitzant modificadors dels que disposa Blender. L'escenari ha estat creat per etapes, les parts que han sigut desenvolupades independentment i mes tard fusionades son el salo principal, el passadís i la entrada. A imatge 8 es pot veure la distribució de la cripta, els rectangles vermell, verd i blau son el salo principal, el passadís i la entrada respectivament.



Imatge 8: Distribució de la cripta

En quant els elements que formaran part del escenari en la part del salo s'han utilitzat estàtues, una tomba, una catifa i entorxes. Al passadís existiran entorxes i dos panells que serviran per obrir la porta que permet l'accés al salo. I a la entrada principal dues estàtues i torxes. Las torxes són importants perquè es el objecte que mes ajuda a donar a l'escenari la ambientació pròpia d'una cripta.

4.1.3. Il·luminació

La il·luminació que es vol aplicar a l'escena es la d'un lloc tancat on no existeixi gaire llum ambiental, així doncs es vol que existeixi un foc de llum d'aquest tipus amb una intensitat gairebé nul·la, i que existeixin focs de llums puntuals, per tal de simular las llums de entorxes . Per tal de aplicar els efectes de llum s'ha utilitzat Unity, en concret gracies al editor de Unity, s'han creat els focus de llum, tant direccionals com puntuals. També s'ha utilitzat *scripting* per canviar la intensitat de la llum i el color d'aquesta quan es consideri oportú.

4.1.4. Sistema de partícules

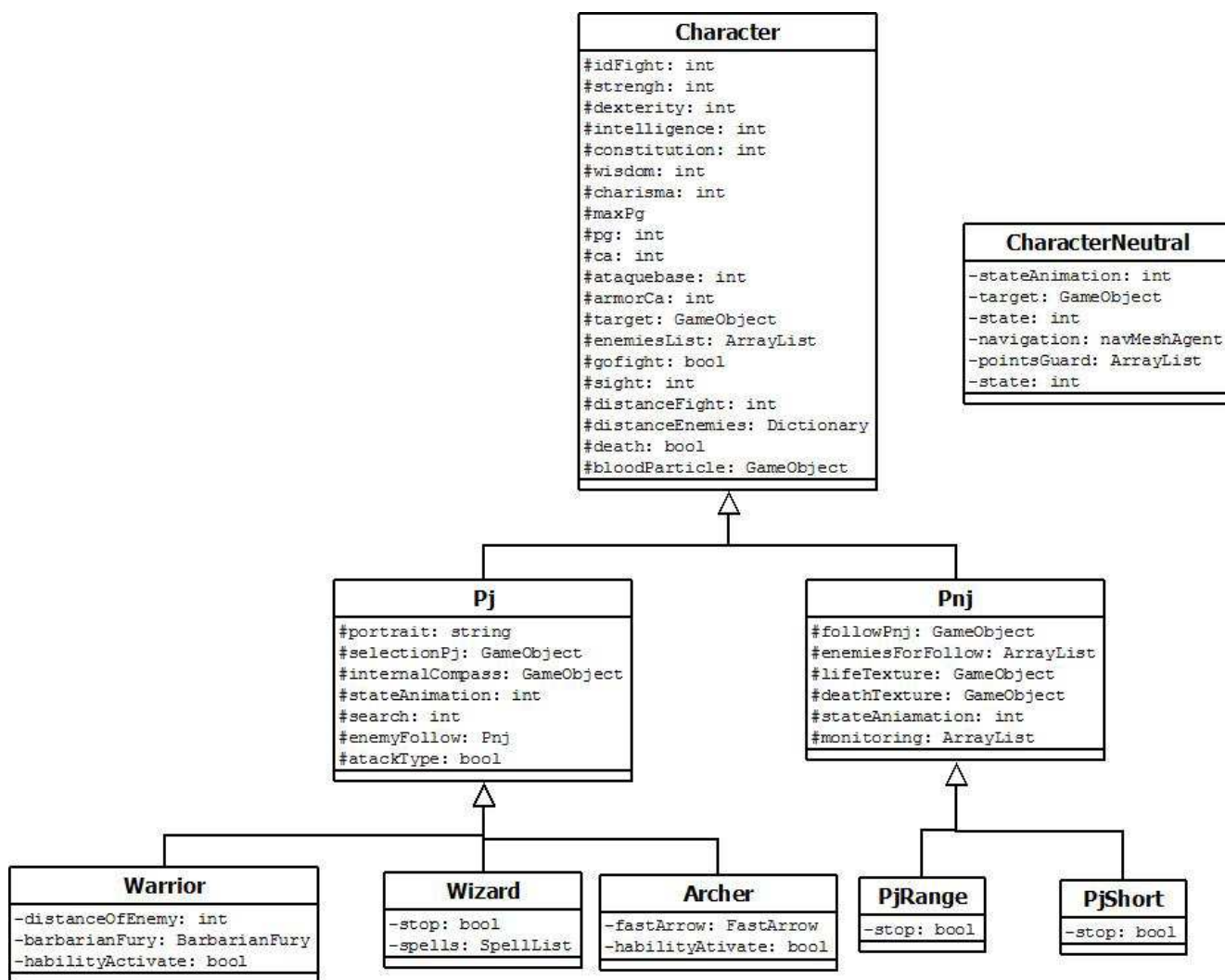
Un sistema de partícules es un sistema on imatges 2D son renderitzades a imatges 3D. A Unity existeixen el sistema de partícules Legacy i el sistema de partícules Shuriken, aquest últim des de la versió 4 de Unity. Els sistema de partícules que he utilitzat ha sigut el Shuriken ja que és menys complicat d'utilitzar que l'antic sistema Legacy. A l'escenari s'utilitzarà per tal de crear el foc i el fum de les torxes.

4.2. Els Personatges

En quan als personatges, s'han de distingir dues fases, la creació dels personatges com a models dels joc i la interacció amb l'entorn al joc. La part de creació dels models, així com la creació del seu esquelet i les animacions es realitzarà fent servir Blender. Per a la interacció de l'entorn, en concret, al que es refereix al moviment d'aquests i les accions que duran a terme en el joc han sigut programades mitjançant *scripting*.

A següents apartats s'explicarà la creació dels esquelets per als diferents models i com s'han creat les animacions.

En quan a la part lògica del joc s'ha gestionat les diferents accions dels personatges, tals com el moviment, les lluites, el llançaments d'encanteris, utilització de habilitats, etc mitjançant el paradigma de programació orientat a objectes. En el següent diagrama de classes que corresponen a la imatge 9 es pot observar la representació que he seguit envers als personatges.



Imatge 9: Diagrama de classes simplificat dels Personatges utilitzats en el joc

A la classe Character es troben les propietats comunes a tots els personatges del joc, com atributs existeixen la representació del *GameObject* que s'utilitza en el joc per a poder accedir als diferents components del que formen part, així com els atributs que identifiquen les característiques dels personatges com són la força, destresa, intel·ligència...; les propietats necessàries per la realització dels combats com la vida (PG), l'atac base, la defensa (CA), la llista d'enemics, la distància entre els enemics... o el sistema de partícules de sang comú a tots els personatges controlables o enemics.

Tots els Caràcters tindran accions comunes entre ells com són les tirades de daus necessàries per la lluita com tirades d'atac, de defensa...

A la classe Pj que hereta de Character on es troben els atributs propis dels personatges controlables com són els *portraits*, els estats de les animacions, habilitats que solament tenen els personatges que controlem, brúixoles internes de cada personatge que s'utilitzaran per calcular el gir que faran en el moviment.

A las classes que hereten de Pj, es adir, ja, els diferents personatges que utilitzarem es trobem els atributs característiques d'aquestes classes com ara son las habilitats pròpies de cada personatge es a dir les habilitats úniques de cada classe de personatge i el constructor de cada una d'aquestes classes.

A la classe Pnj trobem las llistes dels personatges que seguiran els personatges principals per atacar-los quan siguin a la distancia adequada, les barres de vida propias, els diferents punt de rutes que segueixen entre d'altres, es a dir els atributs propis dels Pnj.

A las classes PnjRange i PnjShort com a las de Wizard, Warrior i Archer es troben els atributs propis de cada classe d'enemic així com el seu constructor.

La classe CharacterNeutral representa un personatge presencial en el lloc, només seguirà una ruta definida per uns punts i quan trobi a algun dels personatges controlables pel jugador, anirà fins a un punt del escenari i desapareixerà, abans si mes no, quan hagi localitzat els personatges principals sortirà un missatge pel *textArea* alertant als seus companys.

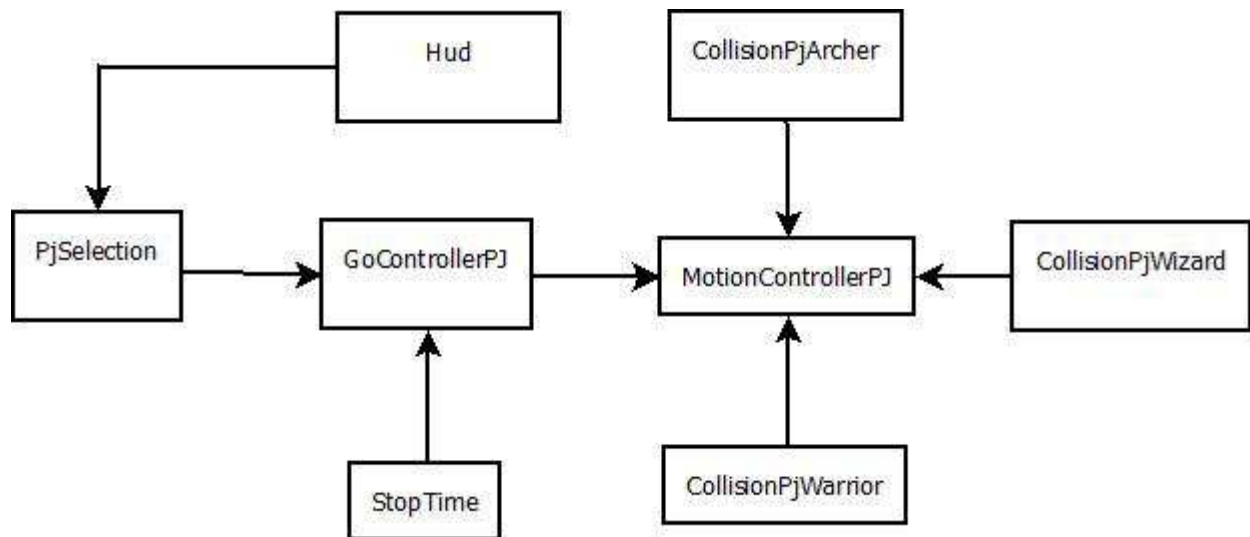
Per crear las instancies dels objectes que pertanyen a les classes definides anteriorment s'ha utilitzat el script SearchCharacter. En aquest script s'ha decidit carregar de forma dinàmica els *GameObjects* que seran atributs de la classe Character pertanyents als enemics. S'ha decidit carregar-los de forma dinàmica pel fet que des de el menú de opcions es pot escollir la dificultat del joc, així doncs depenent d'aquesta dificultat es carregaran un número de enemics o un altre.

Per carregar els *GameObjects* de forma dinàmica s'utilitzen un *prefab*. Un *prefab* es un tipus de *GameObject* reutilitzable, es a dir, que el pots inserir a qualsevol escena i al número de cops que es vulgui. Quan s'afegeix un *prefab* a una escena, es crea una instancia del mateix. Totes aquestes instancies del *prefab* son vinculades al original y son copies d'aquest.

4.2.1. Moviment y Controls dels personatges principals

Als *scripts* que tenen que veure amb la interacció dels personatges amb l'escena son GoController PJ, MotionControllerPJ, StopTime ubicats com components del *gameobject* Cube;PJSelection que funciona com a component dels *gameobjects* warrior, archer i wizard; i el Hud ubicat al *gameobject* hud. A la imatge 10 es mostra el diagrama de interacció que posseeixen els *scripts* respecte el moviment i els controls.

* Consultar al glossari de vocabulari al final del document



Imatge 10: Diagrama de relació entre els *scripts* que controlen el moviment i control dels personatges

La interacció entre aquests *scripts* es la següent, PjSelection, establirà els diferents personatges seleccionables per teclat si es volen seleccionar tots els personatges, o mitjançant el ratolí polsant sobre els personatges si cap d'ells ha estat escollit com a receptor d'algun encanteri del mag.

Amb el *script* GoControllerPJ el personatge o personatges seleccionats podran ser moguts per les diferents parts del escenari. Aquest *script* també es el encarregat de fer el quadre de selecció.

La selecció de personatges es farà de la següent manera:

- Prenent la tecla "A" es seleccionaran tots els personatges.
- Prenent sobre cada personatge es seleccionen de forma individual.
- Arrossegant a mesura que es polsa el ratolí es podrà escollir als personatges que vulguis utilitzant un quadre de selecció.
- Escollint el portrait de cada personatge es pot escollir al personatge.

Posteriorment el *script* MotionControllerPJ s'encarregarà de dotar de moviment tant als personatges seleccionats com els que ja estaven en moviment.

Per el que fa el Hud, també s'encarregarà de seleccionar els personatges, si es seleccionen els *portraits* de cada personatge, així com seleccionar els diferents encanteris o habilitats del personatge

seleccionat, GoControllerPj també controlarà quin personatge ha estat seleccionat en cas d'haver escollit un encanteri que necessiti d'un personatge receptor per tal d'executar l'encanteri.

El *script* StopTime serà l'encarregat d'aturar el temps, si el temps ha estat parat es podran fer totes les accions descrites de la mateixa forma. El temps es parará utilitzant la tecla espai.

També s'ha tingut en compte que si els personatges principals xoquen contra un *collider* del escenari es parin, això es resolt als *scripts* CollisionPjWarrior, CollisionPjWizard i CollisionPjArcher ubicats com a components dels diferents *gameobjects* dels personatges principals.

4.2.2. Els enemics

El *script* que controla als enemics es el pnjController, que actua de component del *gameobject* Cube. En aquest *script* es controla el moviment dels enemics. Es a dir las rutes predefinies que segueixen abans de trobar al personatge principal; l'assignació dels seus enemics,es a dir els personatges principals que seguiran per després lluitar contra ells i com trobar als personatges principals. Cal destacar que depenent de si un enemic pertany a la classe PnjRange o be PnjShort actuaran d'una manera o d'un altre.

Amb el *script* FollowPnj ubicat al *prefab* SeguimientoPnj fem que aquest *prefab* segueixi als enemics, per fer això necessitem el atribut de tipus *gameobject* anomenat SeguimientoPnj. Aquest atribut forma part de la classe Pnj. Per canviar la imatge del punter quan seleccionen un enemic utilitzem el *script* SchangeCursor, quest *script* també es un component del *prefab** SeguimientoPnj.

Per tal de que Les barres de vida pròpies creades per a cada personatge segueixin als enemics s'ha creat el *script* DeathFollowPnj i LifeFollowPnj. A mes per a recalcular la barra de vida restant dels enemics també s'utilitza el *script* LifeFollowPnj.

4.2.3. Animacions dels personatges

Totes les animacions han sigut fetes en Blender. Els personatges principals i enemics disposaran de les següents accions.

Las accions en comú entre personatges principals i enemics son:

- Aturada: Acció que s'executa quan el personatge esta parat.
- Caminar: Acció que s'executa quan el personatge es mou.
- Atac: Acció que s'executa quan el personatge esta en un combat i es el seu torn per atacar.

Las accions que solament te el personatge mag són:

- Fer encantament Armor :Acció que fa el mag quan llança aquest encantament.
- Fer encantament Bullet: Acció que fa el mag quan llança aquest encantament.

Las accions que solament tenen els personatges guerrer i arquer respectivament son:

- Fer habilitat Barbarian Fury: Acció que fa el mag quan activa aquest encantament.
- Fer habilitat Fast Arrow: Acció que fa el mag quan activa aquest encantament.

Les habilitats que tenen en comú els enemics son:

- Córrer: Acció que fa un enemic quan veu a un dels personatges i el persegueix fins a una certa distancia.

A mes els enemics deperent si son de curt o llarg abast tindran dos animacions d'atacs diferents. El personatge neutral tindrà les accions de córrer, caminar i aturada

En quant l'execució de les animacions al joc es controlen mitjançant el *script* Animations. En aquest *script* deperent dels atributs animationState de la classe Pj i Pnj que es van modificant si un personatge ataca, camina, fa una encanteri, va cap a un personatge principal corrent o segueix una ruta caminant si es un personatge enemic, es reproduex l'animació corresponent i es modifica la velocitat de la animació. Aquests *script* es troba al *gameobject* Cube com a component.

4.3. Menús

Existeixen 2 tipus de menús, el menú principal que s'ubicarà a la escena inicial, es a dir abans de jugar i el menú de joc, menú que apareixerà polsant la tecla "P" a la escena del joc.

El menú de joc, s'ha fet amb Unity, utilitzant el editor . Aquest menú consta de tres opcions, nou joc, opcions i sortir. Si es polsa la opció de nou joc, s'executarà el joc, on es carregarà l'escena de joc. En cas de polsar la opció de opcions, es carregarà una nova escena amb el menú de opcions. Aquestes opcions son escollir la dificultat del joc, fàcil o be difícil. Deperent de l'opció escollida quan es carregi l'escena de joc apareixeran mes o menys enemics.

La última opció que es pot escollir es la de sortir del joc, aquesta opció farà que surtis de l'aplicació.

El menú de joc s'ha fet mitjançant un *script*, el nom d'aquest *script* es MenuPauseAparition. Hi tindrem dues opcions, continuar el joc o sortir de l'aplicació.

4.4. Hud

El hud es la informació que es mostra en tot moment a la pantalla durant la partida, generalment en forma de icones i números. El hud mostra el número de vides, punts, nivells de salut, mini mapa... depenent del tipus de joc.

El hud que s'ha fet, s'ha basat en el del joc *Baldur's Gate*, i la informació que disposa es la següent:

- Vida dels personatges: Imatge que mostra els PG dels personatges
- *Portrait* dels personatges: Imatge q mostra un retrat del personatge, quan el personatge mor el retrat apareixerà en escala de grisos
- Vida dels enemics: Imatge que mostra els PG dels enemics
- Habilitats i encantaments dels personatges: Mostra les habilitats i encanteris disponibles dels personatges i el numero de cops que poden ser utilitzats. Els encanteris i habilitats es mostren en una imatge i el numero de cops i habilitats
- Informació de esdeveniments i combats: Mostra informació dels combats i esdeveniments que succeeixen en el joc
- Selecció múltiple: Quadre de selecció per poder moure els personatges
- Personatge seleccionat: Icona que apareix als peus del personatge seleccionat
- Descans: Icona, permet recuperar la vida, màgies i encantaments dels personatges.
- Icona d'atac: Apareix el icona de una espasa quan passem el punter del ratolí sobre un enemic.

A la imatge 11 es podem veure la totalitat dels elements del hud descrits anteriorment excepte el icona d'atac.

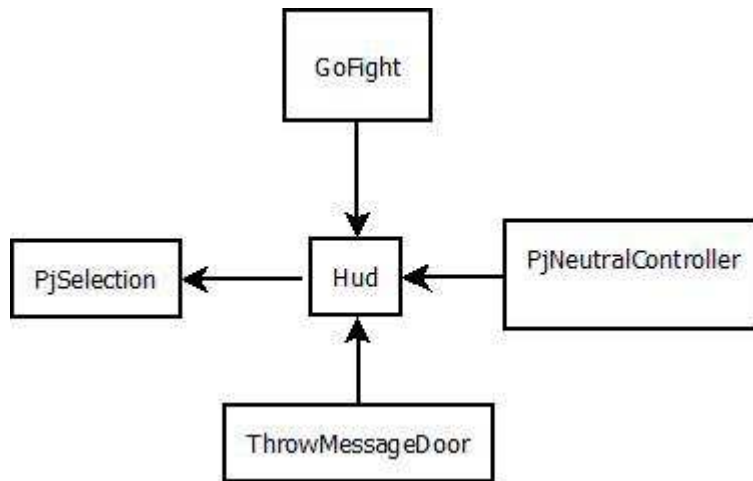


Imatge 11: Visualització del hud del joc

El Hud s'ha creat mitjançant un *script* anomenat Hud. En aquest *script* es defineix el *layout* utilitzat, també fa de intermediari entre les accions que es defineixen a continuació:

- Escollir als personatges polsant els *portraits*.
- Seleccionar els personatges mitjançant un quadre de seleccio.
- Activar els diferents encantaments o habilitats dels personatges seleccionats.
- Descansar, es adir recuperar el número de encantaments o habilitats inicials, així com la vida total.

Els *scripts* en el que fa de intermediari el Hud son GoFight per tal de que els *textArea* mostrin els missatges de les lluites; ThrowMessageDoor per que es mostri el missatge que retorna aquest *script*; PjNeutralController per que es mostri el missatge d'alerta quan aquest personatge et trobi; PjSelection per tal de saber quin es el personatge seleccionat com a destinatari del encantament ArmorSpell. A la imatge 12 podem veure el diagrama dels *scripts* relacionats amb el Hud.



Imatge 12: Diagrama de *scripts* relacionat amb el Hud

Per mostrar els estats dels personatges el *script* Hud podrà accedir a la llista de personatges instanciats al *script* SearchCharacters. D'aquesta forma també podrem tenir informació dels encanteris i habilitats dels personatges.

4.5. Habilitats y Encanteris

Els nostres personatges principals a part de lluitar, tenen habilitats pròpies per poder lluitar contra els enemics. Aquestes habilitats permeten pujar les característiques del personatges principals i algunes altres treuen vida als enemics.

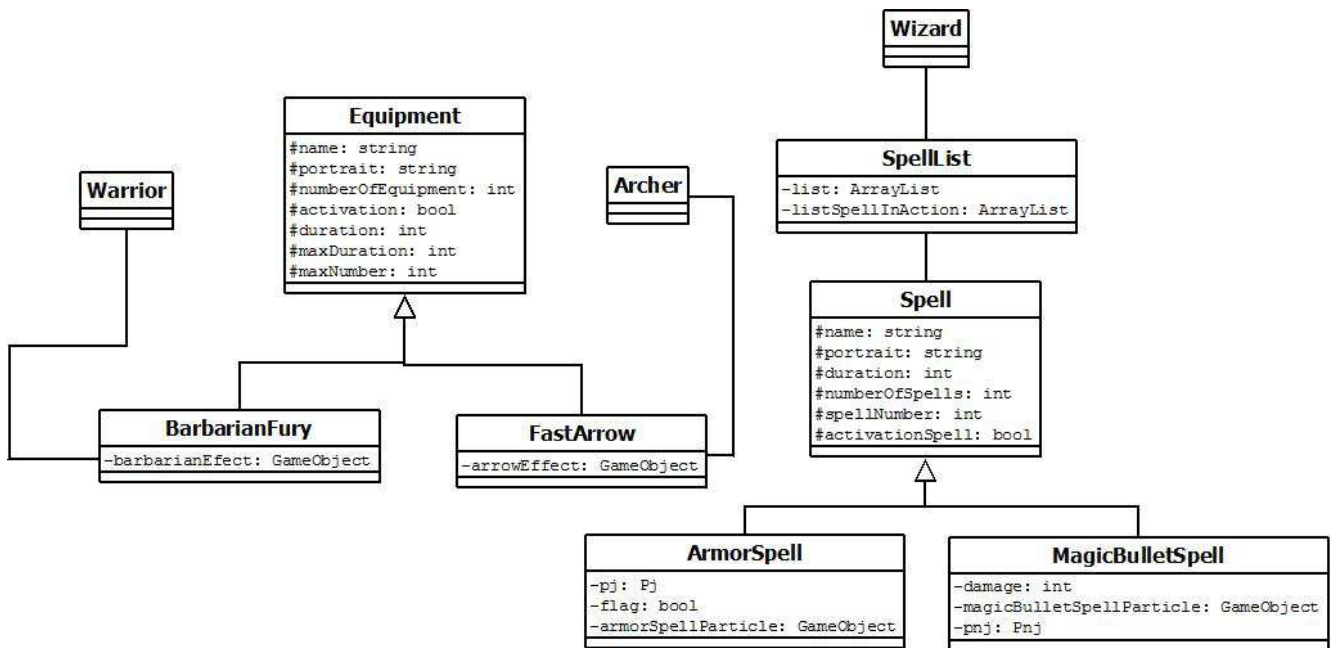
En concret les habilitats son aquestes:

- BarbarianFury: Puja 4 punts de la característica força. Solament la utilitza el guerrer.
- Fast Arrow: Puja 2 punts de la característica destresa. Solament la utilitza l'arquer
- ArmorSpell: Puja dos punts de la característica destresa al personatge receptor. Solament la utilitza el mag.
- MagicBulletSpell: Atac especial del mag, treu 2D6*^{*} de PG, si l'encantament impacta contra l'enemic.

* Consultar al glossari de vocabulari al final del document

Les habilitats pròpies en cas del mag són els encanteris i en cas de ser un arquer o guerrer són les habilitats especials. Per tal de gestionar aquestes habilitats s'ha utilitzat programació orientada en objectes.

Podem veure a la imatge 13 el diagrama de classes utilitzat, per gestionar les habilitats i encanteris.



Imatge 13: Diagrama de classes corresponent les habilitats especials dels personatges.

Com podem veure al diagrama, el guerrer i l'arquer tindran un atribut de la classe BarbarianFury i FastArrow respectivament, aquestes dues classes heretaran de Equipment. La classe Equipment disposa dels atributs protegits portraits, nom, número de habilitats, màxim número de habilitats, duració, activació, màxima duració amb els respectius getters i setters, i dos mètodes abstractes effect i cancelEffect.

La classe Barbarian Fury i FastArrow disposa de la implementació dels mètodes abstractes de la seva classe mare.

El mètode effect de les dues classes crea un *GameObject* a partir del *prefab*, aquest *GameObject* és el sistema de partícules que es mostrarà mentre la habilitat existeixi. Aquests sistemes de partícules han sigut creats mitjançant l'editor de Unity.

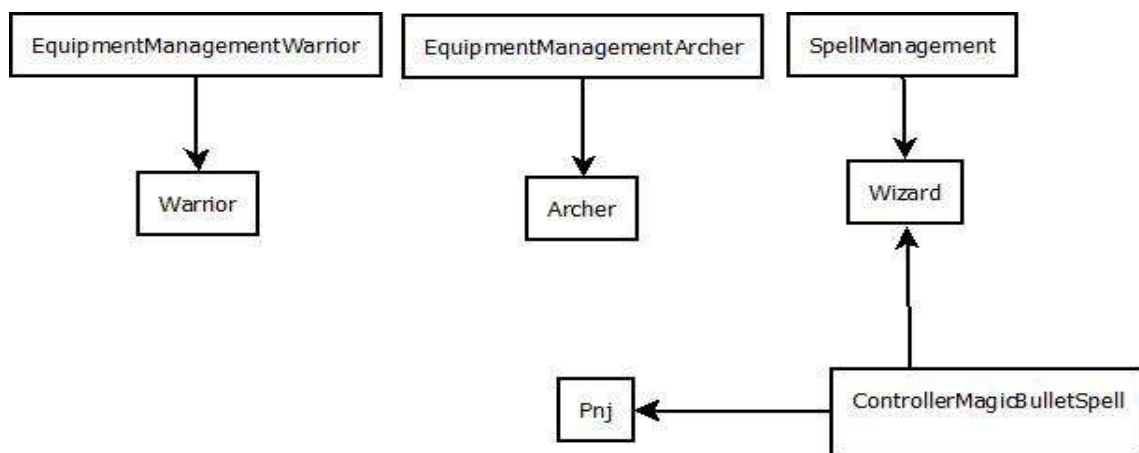
El mètode cancelEffect destruirà el *gameObject* quan passi el efecte de la habilitat.

En quan els encanteris, el mag disposarà d'un atribut SpellList per tal de poder gestionar els diferents encantaments. En aquesta llista, hi ha dos atributs, els *arrayList* : list i listSpellInAction. El primer d'ells es una llista dels encanteris que existeixen i el segon es una llista dels encanteris que estan activats durant el temps que duri el seu efecte.

La classe Spell es una classe que defineix els encantaments, te els atributs tals com el *portrait*, la duració, el numero de encantaments i el *activationSpell*. La classes ArmorSpell i MagicBulletSpell hereten d'aquesta classe i disposen dels mètodes *effectSpell* per activar el funcionament dels encantaments, es adir generar el sistema de partícules associat i canvia les característiques dels personatges afectats per l'encanteri en cas de ser l'encanteri ArmorSpell o be calcular els PG que reduiran vida del enemic en cas d'activar el magicBulletSpell. El *script* ControllerMagicBullet reduirà la vida del enemic, controlarà el projectil que es llança cap al enemic. Per controlar el projectil s'utilitzarà un *rigibody* i per detectar que el projectil topa contra un element s'utilitzarà un *collider*. El sistema de partícules podrà xocar contra qualsevol *collider*, si es així evidentment no treu vida al enemic.

Per tal de que els sistemes de partícules associats als encanteris segueixin als personatges s'ha creat varis *scrips*, per duar a terme aquesta tasca, aquets *scripts* son FollowArmorSpell i FollowHabilitats que seran components dels *prefabs* corresponents a ArmorSpell i tant FastArro i BarbarianFury respectivament.

Per tal de gestionar els encantaments i habilitats dels diferents personatges disposem dels *scripts* SpellManagement ,EquipmentManagementWarrior,EquipmentManagementArcher. SpellManagement sera component del gameobject wizard i EquipmentManagementWarrior,EquipmentManagementArcher seran components del *gameobject* warrior i wizard. A la imatge número 14 podem veure el diagrama que mostra la relació entre els *scripts* de control de encanteris i habilitats i els objectes que s'utilitzen.



Imatge 14: Diagrama de relació entre els *scripts* que controlen el moviment i control dels personatges

4.6. Combats

Les lluites entre els personatges del joc segueixen unes regles bàsiques del joc de rol D&D, en el punt 3.7 s'expliquen aquestes regles.

DISTANCIES DE COMBAT

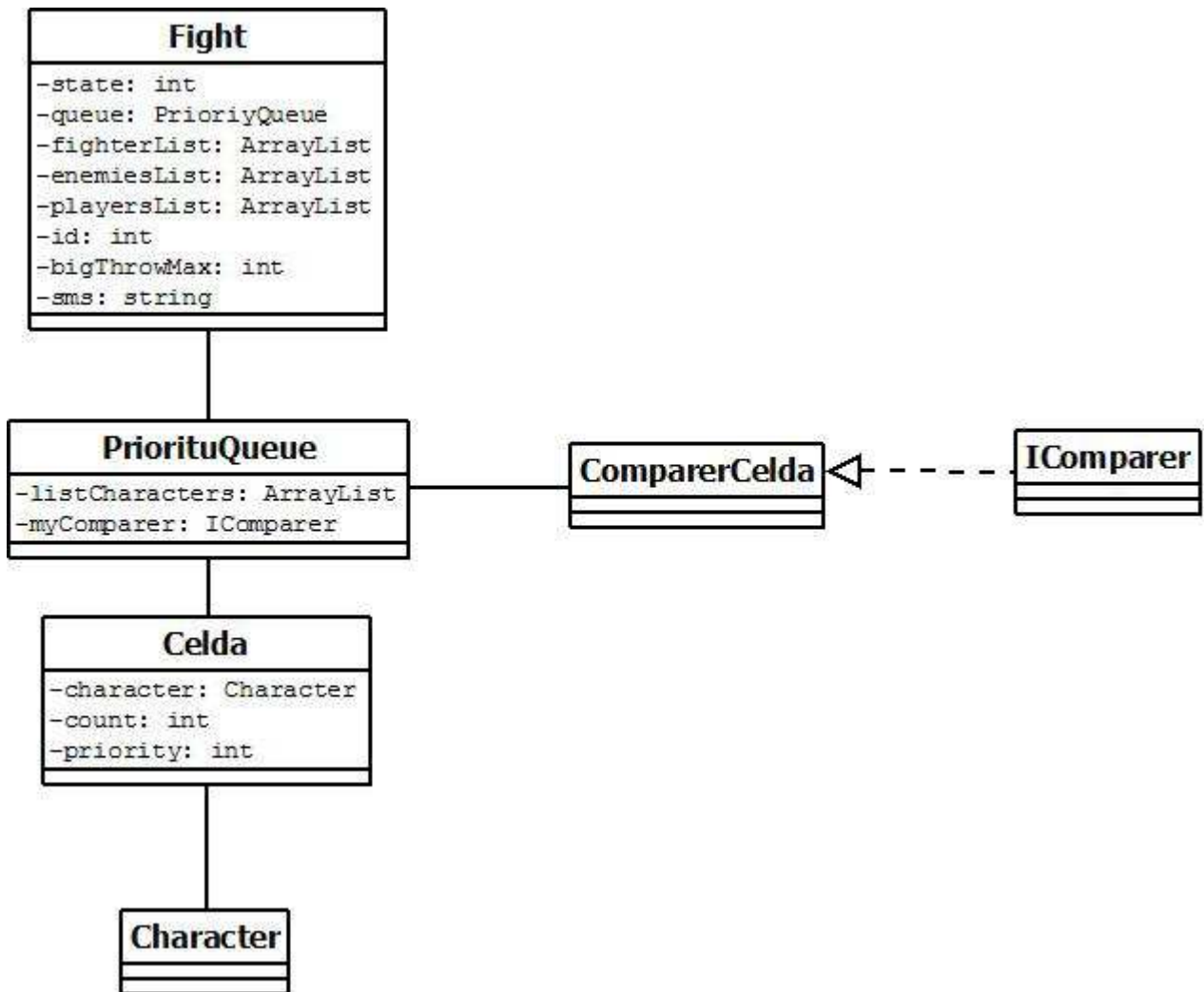
Per que s'iniciï un combat, els personatges hauran d'estar a una certa distància, aquesta distància depèn del tipus de personatge. És a dir l'arquer podrà lluitar des de una distància més llunyana que un guerrer o un mag. Igual passa amb els enemics, la distància de lluita d'un enemic de la classe PnjRange serà més llarga que la distància d'un PnjShort. Per tal de establir les diferents distàncies de combat cada un dels personatges té com a atribut una distància de lluita.

Per saber la distància que hi ha entre un personatge i els seus enemics, s'ha utilitzat un diccionari, aquest diccionari és atribut de tots els personatges (Character Class). Es calcula aquesta distància de forma consecutiva actualitzant els valors dels diccionaris de cada personatge. Això està implementat al *script* UpdateDistances, que és component del *gameobject* Cube. Els enemics dels personatges s'estableixen en el *script* SearchCharacter.

Lavors es començarà una lluita quan la distància del diccionari que hi ha entre un personatge principal i un enemic és igual que la distància de lluita de cada personatge. Quan això succeeixi s'inicialitzaran els combats i s'instanciarà un objecte Fight.

DESCRIPCIO DELS COMBATS

Per gestionar els combats s'ha utilitzat programació orientada a objectes. A la imatge número 15, es pot veure el diagrama de classes.



Imatge 15:Diagrama de classes corresponent a la gestió dels combats.

El *script* que gestiona les lluites es l'anomena't GoFight, aquest *script* es component del gameobject Cube.En aquest *script* s'instancien les lluites quan las distancies dels personatges ho requereixen, a mes si una lluita entre personatges existeix i un altre personatge vol lluitar, aquest personatge s'incorporarà a la lluita existent. En aquest *script* també es gestiona el final d'una lluita.

Cada lluita tindrà una cua, aquesta cua esta formada per una llista de Celdas,objecte format per un Character, una prioritat i un comptador. La cua serveix per gestionar els torns d'atac entre els diferents personatges. S'han gestionat els torns en funció d'una prioritat que es calcula en funció de la característica de destresa dels personatges.

Un personatge tindrà el torn quan l'atribut count de la classe Celda sigui 0, aquest comptador es va reduint al mateix temps per a cada celda de la cua, llavors el que passa es que qui te una prioritat de torn mes alta podrà atacar mes cops que un altre personatge. A la imatge 16 corresponent al videojoc FFXII es pot veure que quan la barra de atac arriba al final, el personatge tindrà el torn per atacar.



Imatge 16: Hud de gestió de torns d'atac del joc FFXII

Quan al personatge tingui el torn i l'utilitzi, el atribut comptador de la celda de la cua tornarà a tenir el valor de la seva prioritat.

El funcionament de la las lluites es el següent, quan els personatges que son enemics es troben a la distancia d'atac, es realitzaran las tirades de prioritat del personatges, quan arribi el torn d'atac del personatge amb la prioritat mes alta es farà una tirada d'atac, aquesta tirada serà enfrontada contra la tirada de defensa del personatge a qui s'ataqui, si la tirada d'atac es mes alta que la de defensa, el atacant farà una tirada de dany, per restar la vida del enemic. Això es farà consecutivament fins que un dels personatges mori, llavors es treu de la cua aquell personatge que ha mort i també es treu de la llista de enemics del personatge que ha matat al enemic. Al punt 5.4.7 s'explicarà com s'ha implementat les tirades quan hi ha una lluita i els demés scripts que intervenen als combats.

Cal esmentar que s'ha creat la classe ComparerCelda que implementa el mètode compare de la interfície IComparer, per ordenar les celdas segons el atribut prioritat del la classe Celda.

4.7. La Càmera

Per fer una càmera de vista isomètrica s'ha creat un *script* anomenat CamFollowPJ, aquest *script* farà que es mostri la escena des de una altura i amb un zoom determinat i seguirà al personatge seleccionat.

4.8. Fi del joc i morts

El joc finalitzarà en dos casos, guanyes o perds. Es guanya si es troba l'espasa màgica, es a dir si es maten a tots els enemics de la cripta, es perd si els enemics maten als personatges principals.

Si es guanya apareixerà un sistema de partícules en front de la tomba que hi ha al saló i la espasa màgica. Per tal de sortir s'haurà d'utilitzar el menú de pausa. A la imatge numero 17 podem el succés del que passa si guanyes.



Imatge 17: Esdeveniment en cas de que es guanyi la partida

Si es perd apareixerà un *GuiTexture*, es a dir una imatge de fi del joc i es sortirà al menú principal polsant el ratolí. A la imatge 18 es pot veure la imatge de fi del joc.



Imatge 18: Esdeveniment en cas de que es perdi la partida

Pel que fa a les morts dels personatges, quan es morin es destruirà el *gameobject* corresponent al personatge i s'instanciaran *gameobjects* cadàvers i un *decal** que representa sang. A la imatge 19 es pot veure els cadàvers i el sagnat un cop mort un enemic.



Imatge 19: cadàver d'un enemic i el *decal* que representa el sagnat

En el cas de que es mori el mag i hagin encantaments activats, els encanteris deixaran de tindre efecte. En cas de que es mori un personatge que tingui activat un encanteri de mag o una habilitat especial deixaran de tindre efecte. Al *script* anomenat *ControllerDead* es gestiona aquestes funcionalitats. Aquest *script* es un component del *gameobject* Cube.

5. Implementació

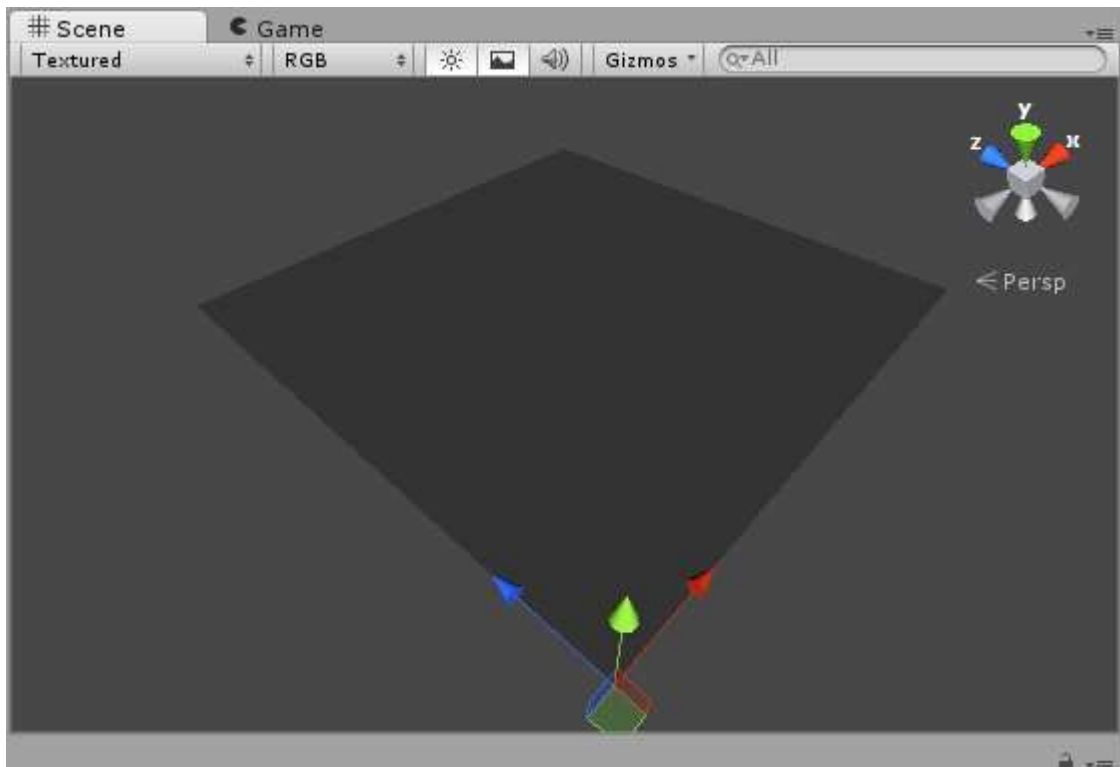
5.1. Escenari

5.1.1. Terreny

Per crear el terreny he utilitzat una eina de Unity anomenada *Terrain*, aquesta eina es molt útil per la creació d'escenaris¹. Al iniciar un projecte de Unity, per defecte ja es crea un *terrain*, però si es vol crear un de nou, es podrà afegir mitjançant el menú, *GameObject-> Create other->Terrain*. A la imatge 20 podem veure un *terrain* creat sense haver modificat cap paràmetre d'aquest.

* Consultar al glossari de vocabulari al final del document

¹[6] <http://desnovato.blogspot.com.es/2012/04/como-crear-una-escena-en-unity3d.html>



Imatge 20: Terrain creat de forma automàtica per Unity


Per modificar l'aspecte del terreny haurem de modificar les opcions d'aquest a partir del menú de *terrains* del que disposa Unity.



Imatge 21: barra de ferramentes de terrain

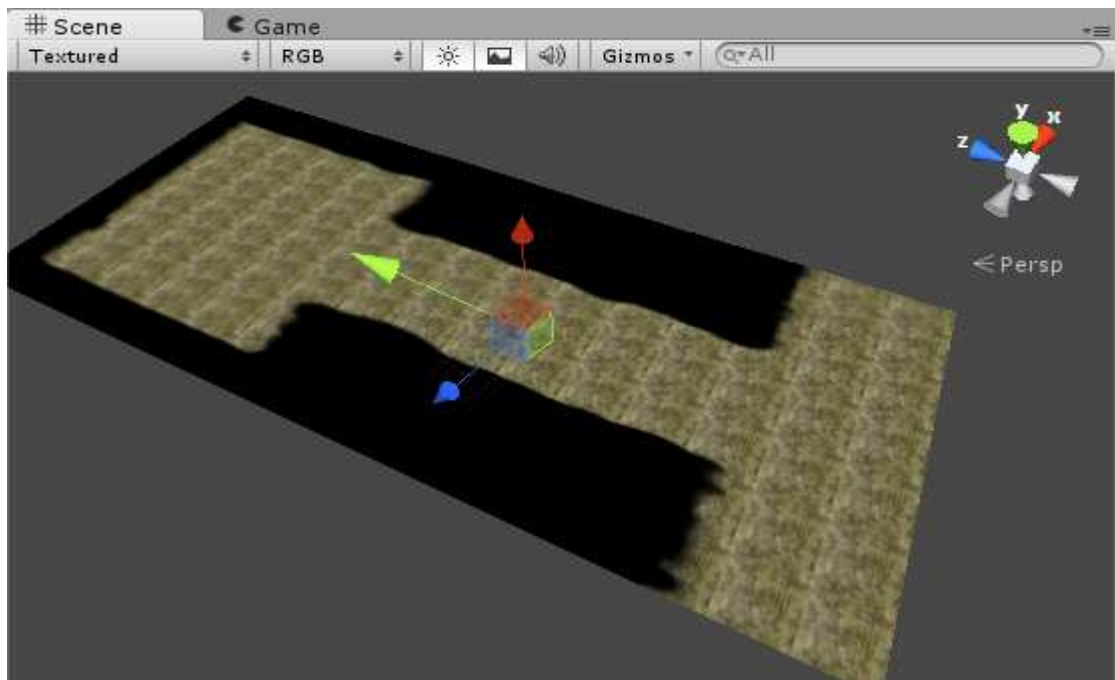
Aquestes opcions són elevar terreny, elevar o disminuir el terreny a una altura concreta, suavitzar la zona desitjada del terreny que es vol aplicar, aplicar textures, afegir arbres, afegir flors i opcions del terreny

Les opcions que he utilitzat per tal de donar l'aspecte final al escenari han sigut l'opció de afegir textures i de canviar opcions del *terrain*.

Per aplicar textures es pulsarà  i després *edit textures* i escollirem les textures desitjades, en el meu cas m'he decantat per un terreny de pedra. Al aplicar la textura podrem veure que Unity ho fa de forma automàtica. També es pot pintar altres textures sobre la que està aplicada, això es fa afegint un altre textura, seleccionar un dels pinzells que ens ofereix Unity i començar a pintar on es desitja.

He utilitzat aquesta opció aplicant una textura de *background** negre, per tal de que no es visualitzí la textura inicial fora de la cripta.

He utilitzat les opcions del terreny per tal de modificar la allargada i la amplada del terreny inicial. A la imatge 22 podem visualitzar el terreny final creat.



Imatge 22: Escenari un cop aplicat les opcions comentades anteriorment

5.1.2. L'arquitectura

La creació de la cripta a estat feta amb Blender, a continuació s'explicaran els passos que s'han dut a terme per tal de crear-la.

MODIFICADORS UTILITZATS

Per a fer aquestes parts, es van utilitzar dos modificadors, el modificador *array* i el modificador *boolean*.

- MODIFICADOR ARRAY

El modificador array bàsicament el que fa es copiar un objecte i desplaçar-lo a un offset respecte a un eix. Això ho fem per que l'escenari sigui simètrica.

* Consultar al glossari de vocabulari al final del document

Aquest modificador te 3 operacions bàsiques.

Fit Curve: Genera còpies del objecte fins a omplir tota la corba. Es necessita haver definit abans una corba.

Fit Length : Genera còpies dels objectes fins a la longitud desitjada

Fixed count: Genera el numero de còpies del objecte que s'hagi definit en el comptador.

- MODIFICADOR BOOLEAN

El modificador boolean ens permet fer les operacions de intersecció, unió o diferència entre dos objectes.

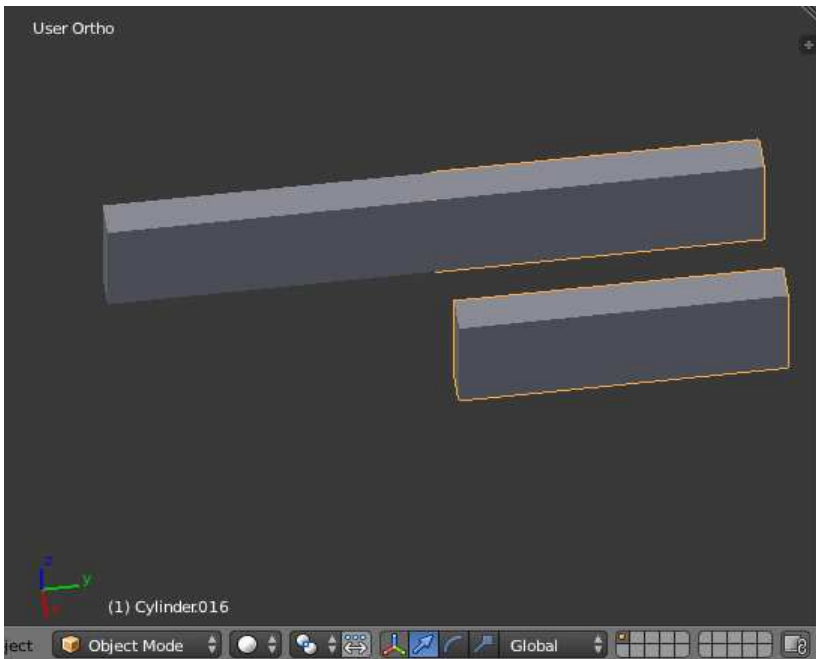
La diferència : L'objecte destí es resta l'objecte que el modifica

La unió: L'objecte objectiu es suma al objecte destí

La intersecció: aplicant aquesta opció el resultat es la part comú dels dos objectes.

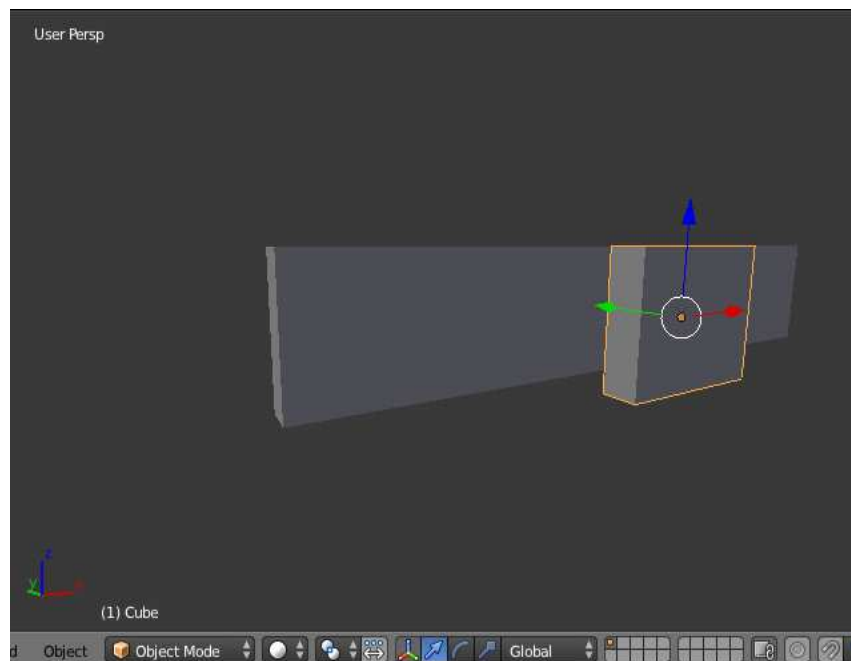
CREACIO DE LA CRIPTA

Per fer el saló principal primer vaig començar fent una de les parets laterals de la sala, la paret no es mes que un cub escalat, i per fer el conjunt de parets vaig utilitzar el modificador *array*, de forma que tingues una còpia d'aquesta paret al costat de la inicial i en front, a partir de la paret clonada al costat de la inicial vaig fer la que esta al front d'aquesta. Per la paret situades a la part nord i sud de la sala vaig seguir el mateix procediment. A la imatge 23 podem veure el resultat d'aplicar el modificador *array* per fer unes parets simètriques a una distància en funció del offset.

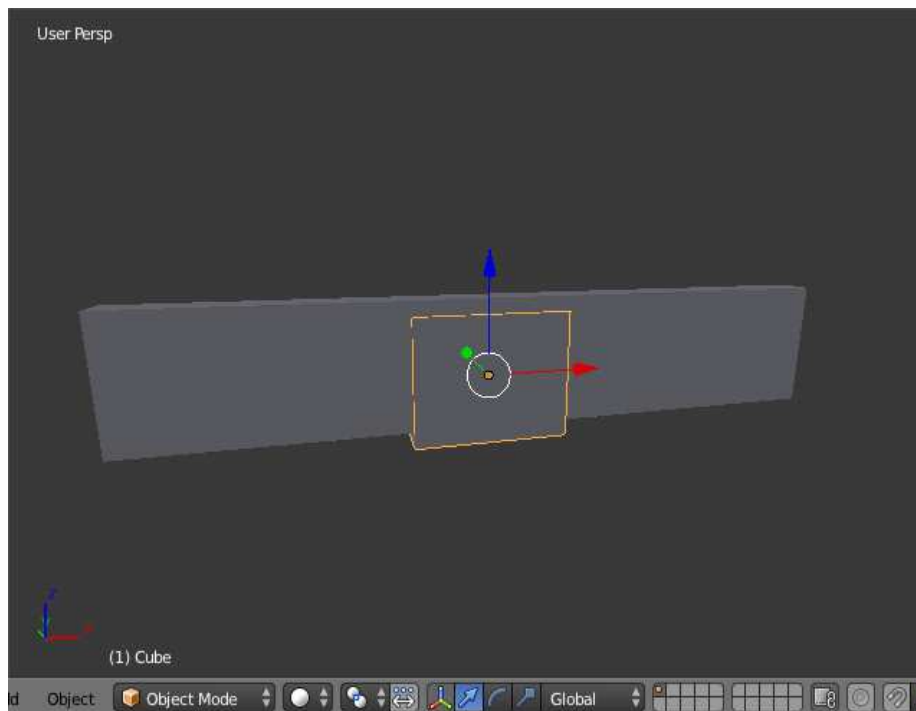


Imatge 23: Aplicació del modificador array

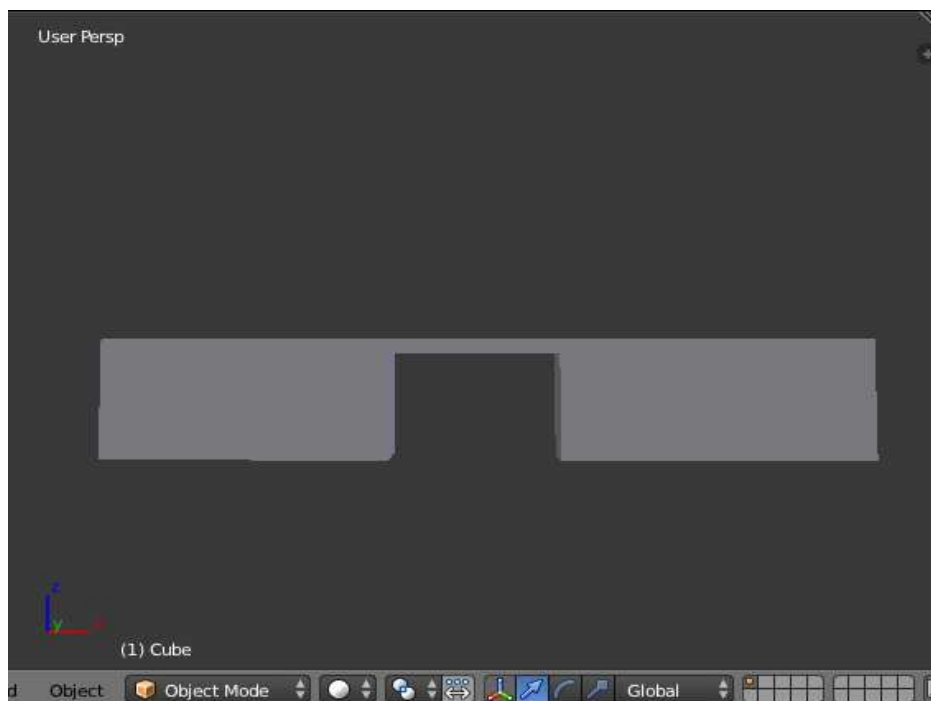
El buit de la porta es va crea utilitzant el modificador *boolean* contra les dues parets nord del salo, amb l'operació diferencia. Las portes son dos cubs, per fer-les vaig crear un dels cubs i després vaig aplicar un altre cop el modificador *array*. A les imatges 24,25 i 26 es pot veure el procediment de crear el buit de la porta del salo.



Imatge 24: creació d'un cub abans d'aplicar el modificador boolean



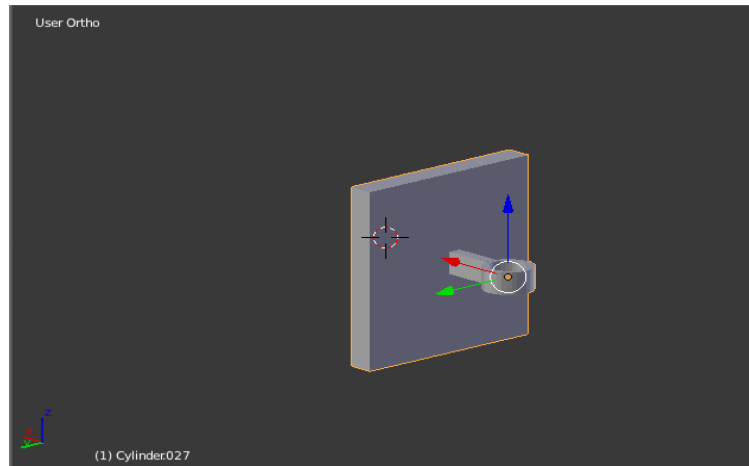
Imatge 25:Correspondència dels objectes per aplicar el modificador boolean



Imatge 26:Resultat donal despres d'aplicar el modificador

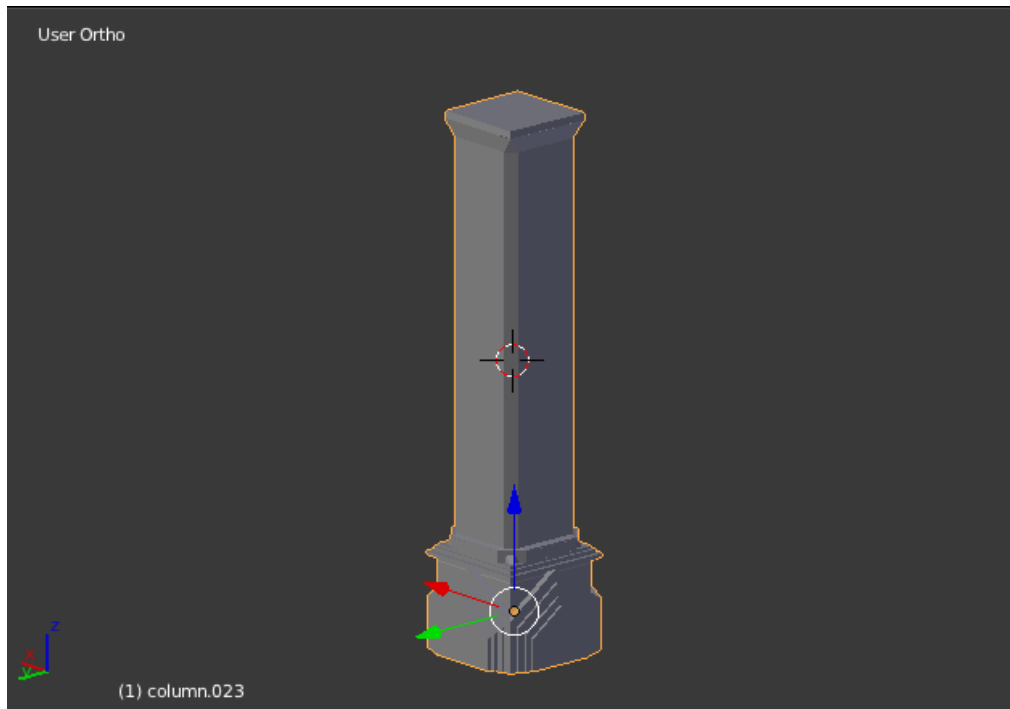
Per finalitzar , es van crear dos petits cubs invisibles, per tal de que funcionin com a frontisses, això s'ha fet per poder obrir les portes del salo en el joc mitjançant un *script*.

Els acoblaments de les torxes es van crear mitjançant un cub escalat com a base, un altre cub escalat i un cilindre. Per fer el cilindre on va la torxa també es va utilitzar el modificador *boolean*, per tal de fer un forat. A la imatge 27 podem visualitzar el model dels acoblaments utilitzats. Es va utilitzar el modificador *array* per distribuir-les per la cripta.



Imatge 27: Model del acoblament de les entorxes

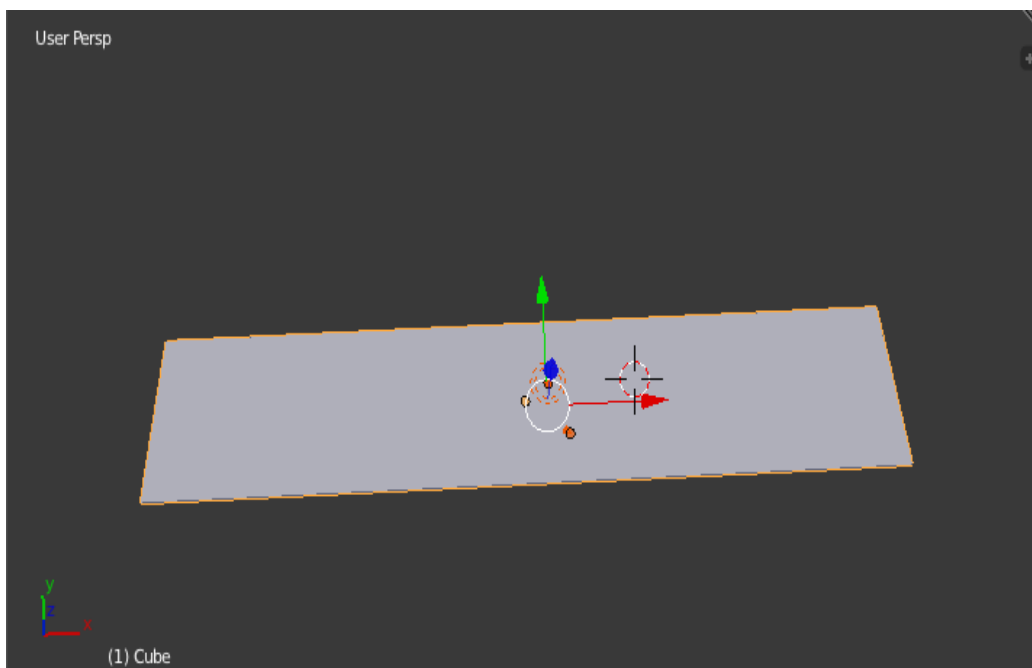
Per fer el conjunt de columnes es va utilitzar el modificador *array* per distribuir-les simètricament. Cal dir que la columna va ser cedida, solsament van ser modificades posant un cub per tapar el forat de la columna situat a la part de dalt, ja que existia un buit a la part superior, això es va solucionar escalant un cub. A la imatge 28 podem veure el model de la columna.



Imatge 28: Model utilitzat per les entorxes

Un cop fet el salo vaig fer el passadís i la entrada principal, aquestes dues parts es van fer utilitzant el mateix procediment que el utilitzat per fer el salo.

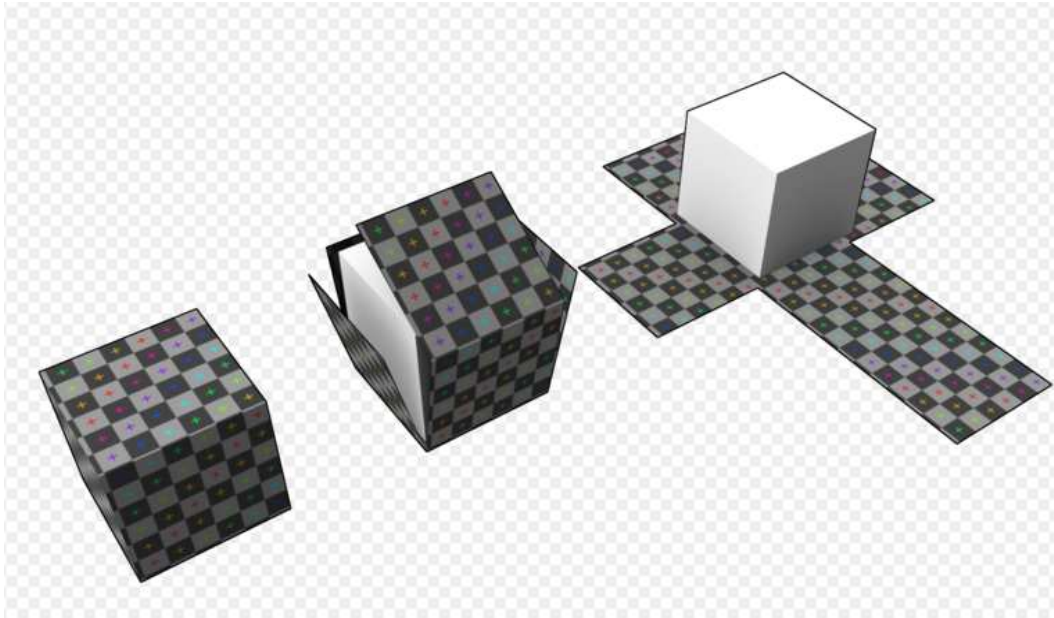
Vaig crear una catifa a partir d'un cub, la catifa te una llargada que va des de el inici de la porta fins als peus de la tomba. A la imatge 29 podem veure el model de la catifa.



Imatge 29: Model per a la catifa

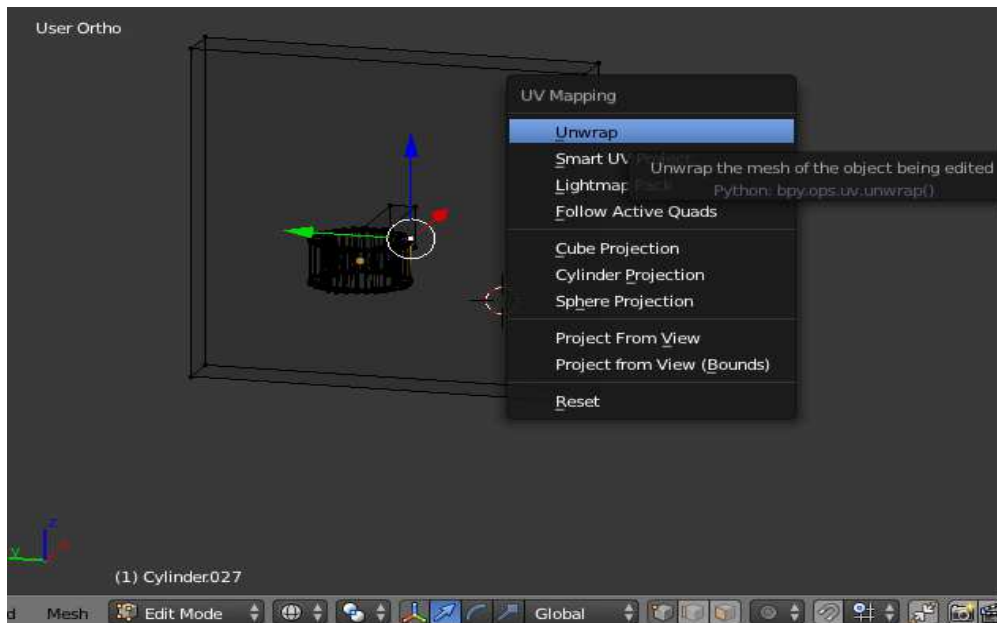
APLICACIO DE LES TEXTURES(UV-MAPPING)

El UV Mapping es el procés per mapejar una textura 2D en un objecte 3D, es a dir embolicar un objecte amb una imatge. El UV Mapping descriu quina part de la textura es aplicada a cada polígon del objecte. cada vèrtex de cada un dels polígons tenen unes coordenades 2D, anomenades U i V.



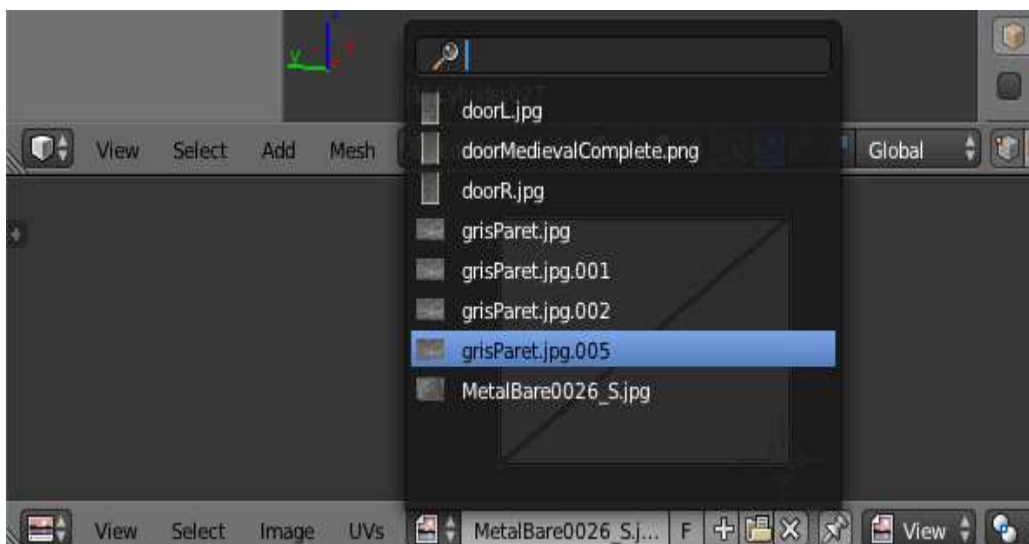
Imatge 30:Técnica Uv Mapping aplicada a un cub

Per acabar la cripta es van aplicar les textures des de Blender, es tria l'objecte que es vol aplicar la textura en mode edició i realitzes un *Unwrap*, polsant la tecla "U", això farà que la figura 3D es torni en un pla en 2D.



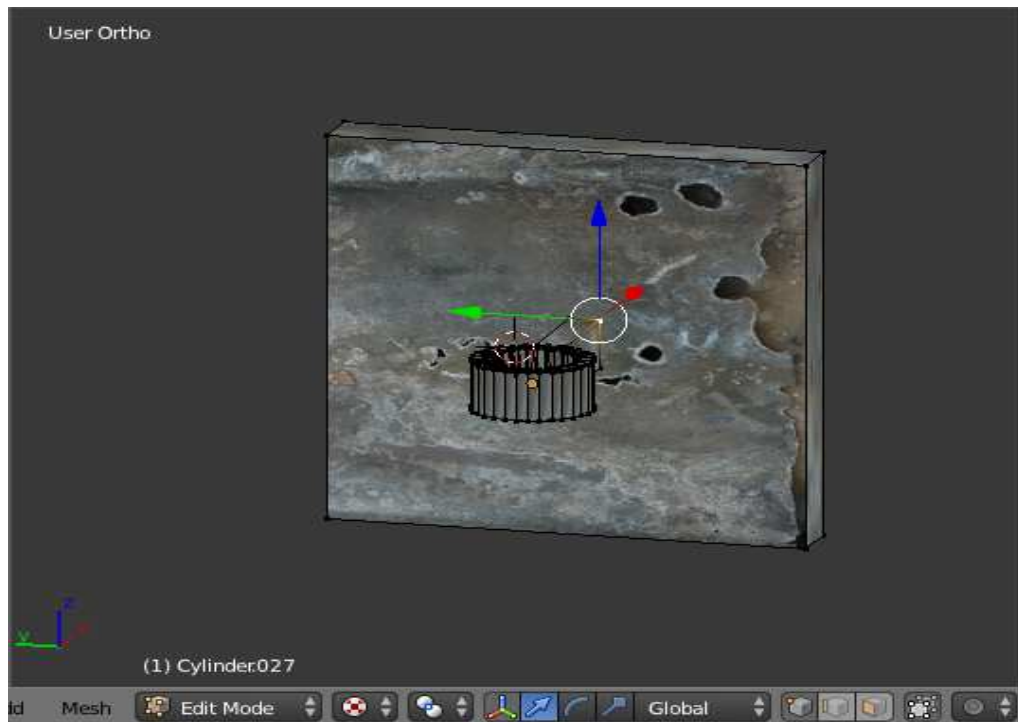
Imatge 31: Aplicació de unwrap al model en blender

Aquesta tècnica s'anomena *UV Map*, obres una finestra del editor en mode *UV/image editor* i es selecciona la textura desitjada. A la imatge 32 es mostra com escollir la textura que es voldrà aplicar al model. Es mostrarà la imatge en *UV Map* de forma automàtica



Imatge 32: Selecció de la textura

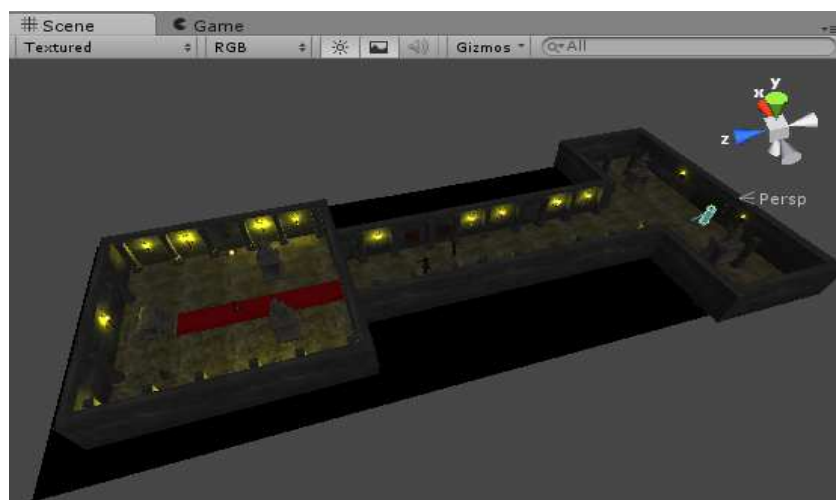
Per visualitzar com queda has de seleccionar el mode textura en la pestanya *viewport shading*. A la imatge 33 podem visualitzar l'aplicació de la textura a la base del acoblament.



Imatge 33: Visualització final de l'aplicació de la textura a la base del objecte

Totes les textures han sigut extretes de la pagina web <http://www.cgtextures.com/>.

Un cop fet això, exportem el nostre model en format .fbx*. Aquest model l'utilitzarem a Unity. Després importarem aquest model creat, i uns altres que s'han descarregat de la web que serveixen com a ornaments de la cripta, aquets models son les estàtues, les torxes i la tomba. A la imatge 34 podem observar l'escenari final amb els diferents models un cop importat a Unity.



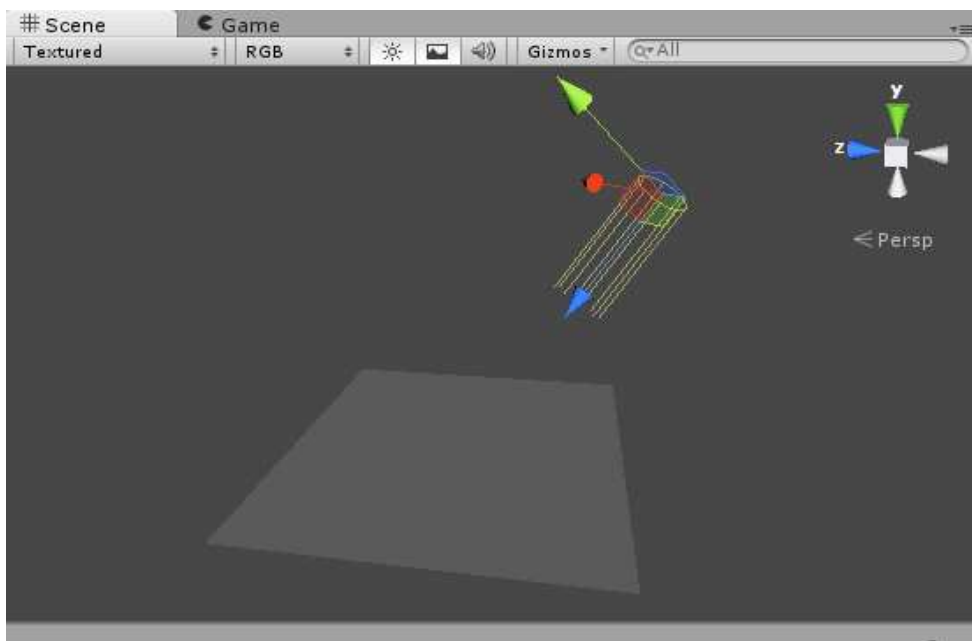
Imatge 34: Visualització del Escenari final amb il·luminació

* Consultar al glossari de vocabulari al final del document

ILUMINACIO

La il·luminació va ser creada des de Unity mitjançant *scripting* i utilitzant dos tipus de llum, la llum direccional aplicada directament a l'escena i els punts de llum aplicats a cada una de les torxes de las que disposa el escenari.

Per a crear la llum direccional es fa des de el menú de Unity, *GameObject-> Create other->Directional Light*, vaig decidir baixar d'intensitat d'aquesta llum, per donar la sensació de estar a les fosques. En concret a 0.1.



Imatge 35: Llum direccional

Unity incorpora un menú per poder personalitzar les llums, com el color, la intensitat de llum, etc.



Imatge 36: Menú del GameObject llum direccional

Abans d'explicar els punts de llum, s'ha de comentar l'estructura del objecte torxa, la torxa es un *prefab*, un *prefab* es un tipus de *GameObject* reutilitzable, es a dir, que el pots inserir a qualsevol escena i al número de cops que vulguis. Quan s'afegeix un *prefab* a una escena, es crea una instància del mateix. Totes aquestes instàncies del *prefab* son vinculades al original y son còpies d'aquest.

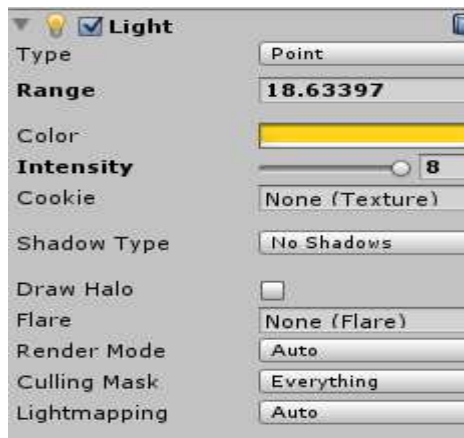
Després d'aplicar els punts de llum a la torxa, vaig fer que aquest objecte punt de llum (anomenat *torch_light*) fosi fill del objecte *Torch*. Es fa arrossegant l'objecte *torch_light* fins al objecte *Torch* de la finestra de jerarquies. Pels sistemes de partícules del foc i del fum s'ha fet el mateix.

Així doncs, la torxa es un *prefab*, la seva estructura la podem observar a la finestra de jerarquies de editor de Unity. A la imatge 37 pot observar aquesta estructura. *Torch_light* es el punt de llum creat, *torche* es l'objecte en si mateix que té com a fills els diferents sistemes de partícules, *fire*, *smoke* i *etincelles* que després explicaré.



Imatge 37: Finestra de jerarquies dels diferents GameObjects

Inicialment he creat el punt de llum amb les característiques que es poden veure a la imatge 38. Mes concretament he modificat el rang fins on es veurà la llum i la intensitat de la llum dins d'aquest rang. Aquestes característiques seran modificades via *script* en temps d'execució del joc.



Imatge 38: Menu del GameObject llum puntual

Al *script* TorcheLight a gran trets el que es fa es variar la intensitat de la llum i canviar el color de la llum respecte la intensitat calculada, comentar que es varia la propietat del sistema de partícules *eticelles* en funció d'aquesta intensitat de llum.

SISTEMA DE PARTICULES

Per fer un sistema de partícules anem al menú *GameObject>create other>Particle System* i a la escena podem visualitzar una pluja de partícules rodones i blanques, ara al inspector podem veure totes les opcions per a canviar les nostres partícules. A la opció *Renderer* podem canviar la nostra textura, la mida màxima de les partícules utilitzades entre d'altres opcions, però no es suficient per tindre un efecte realista. Fa falta canviar la configuració de les nostres partícules.

A la Api de Unity pot veure detalladament les diferents opcions de configuració dels sistema de partícules.

A la imatge 39 es pot veure l'efecte del sistema de partícules creat per el foc i el fum de la torxa.



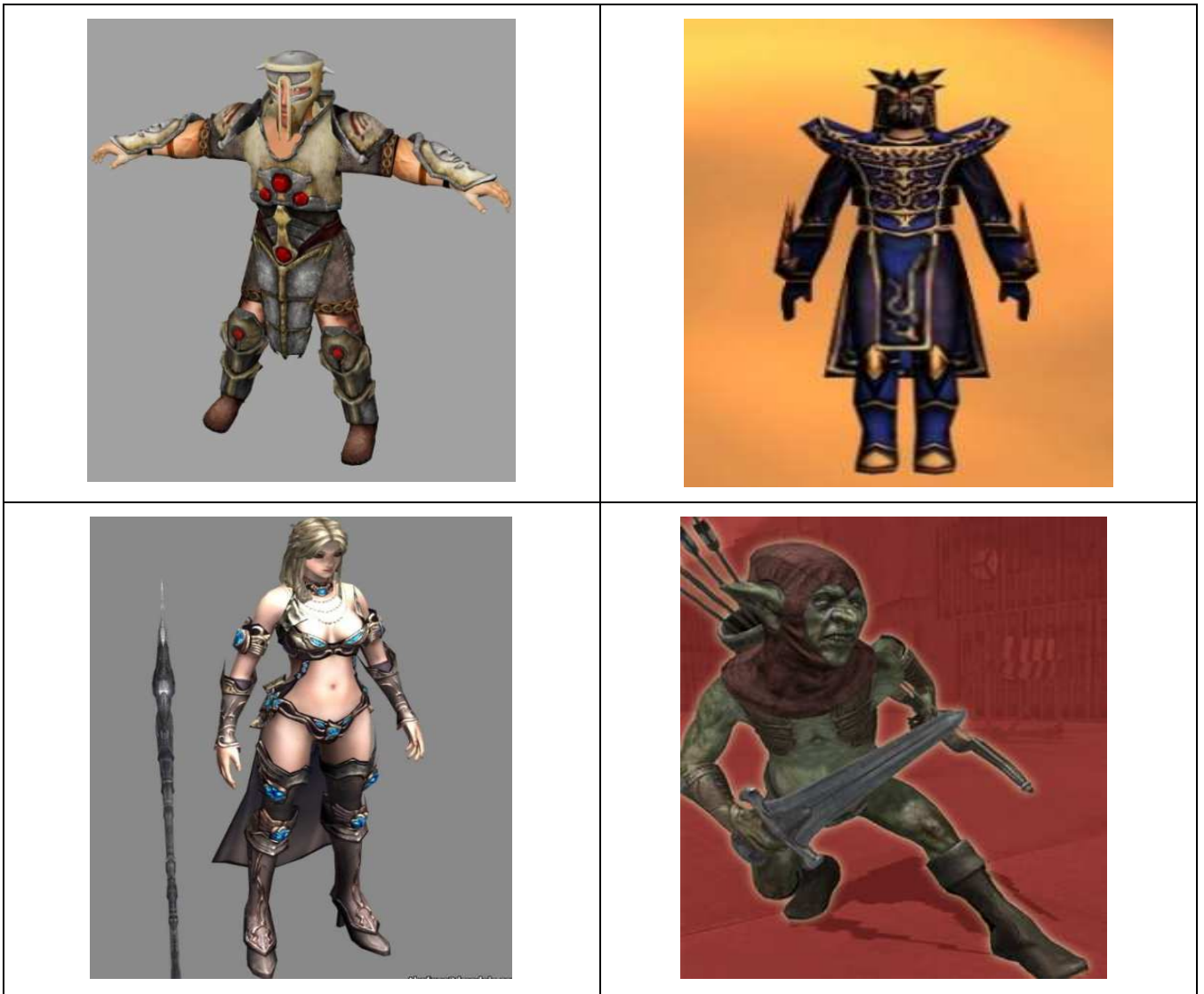
Imatge 39: Visualització del model torxa, amb els sistemes de partícules i llum

L'efecte del sistema de partícules *etincelles* no es pot veure amb claredat a la imatge, però al joc, es pot veure, aquest efecte són punts que van sorgint al voltant de la entorxa que simulen la cendra incandescent que surt a vegades d'una foguera.

Abans de parlar sobre el comportament en el joc dels diferents elements i sobre la interacció entre aquests elements parlaré sobre la importació dels models creats des de Blender, que es convertiran en objectes que utilitzarem en aquest motor gràfic i sobre els components tant d'aquests objectes importats com dels que crearem al propi *engine*.

5.2. Creació dels personatges

Els personatges jugables utilitzats es poden veure a la imatge 28,29,30. Es van descarregar de la pàgina <http://tf3dm.com/>, el model que representa els enemics va ser descarregat de l'*asset Store* de Unity. Utilitzar un model ja fet i amb textures va tindre avantatges, ja que el nombre de polígons que tenien els models no eren elevats i per tant no es perd rendiment en el joc. L'únic inconvenient ha sigut escalar els tres personatges aproximadament a les mateixes mesures. Amb els tres personatges ja escalats solament faltaria aplicar les textures, crear l'esquelet i fer les animacions, tasca que s'ha fet amb Blender. Les textures s'apliquen de la mateixa forma explicada en el apartat Aplicació de textures del punt 5.1.2. A la imatge 40 es poden veure els diferents models utilitzats com a personatges del joc.

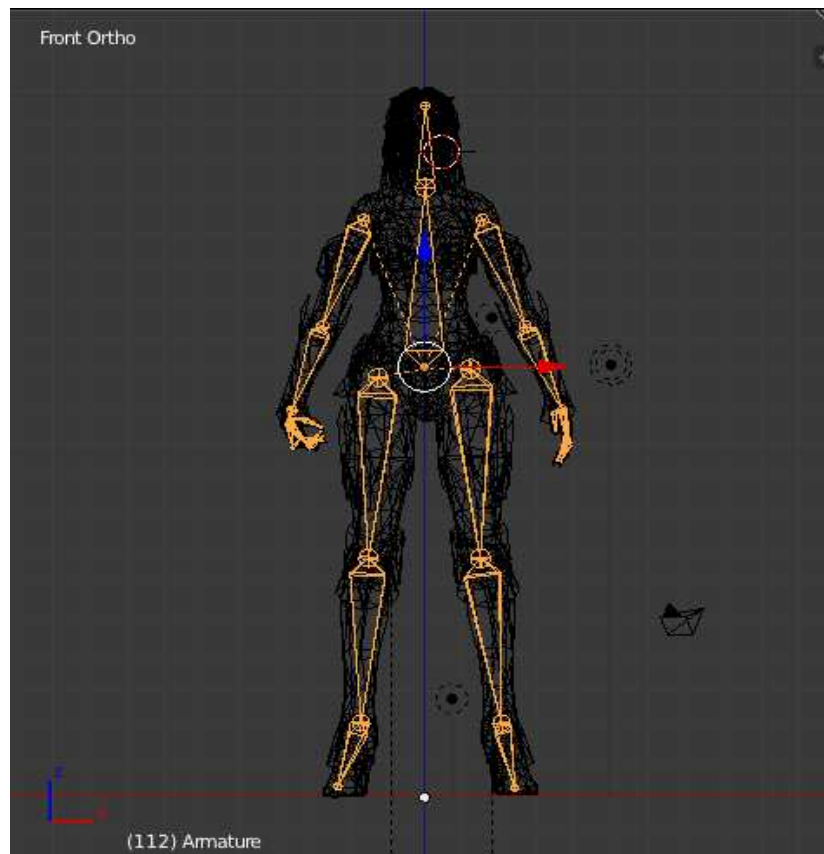


Imatge 40:Models utilitzats en el joc

5.2.1. L'esquelet(Armature)

CREACIO DELS ESQUELET

Per poder dotar de moviment als personatges, animar-los, prèviament s'ha de crear un esquelet (*armature*). Per poder animar el model s'hauran de posar en correspondència els ossos del esquelet amb les diferents parts d'aquest model. Cada un dels ossos aniran d'una articulació del cos a una altre, així podrem rotar les parts del cos correctament per després animar-les. A la imatge 41 podem veure l'esquelet que s'ha fet a un dels personatges principals.



Imatge 41: Esquelet del model de la arquera

Per crear un esquelet haurem de pulsar `shift+a ->armature->single bone`, aquest primer os,el principal, mourà tot el model, es situa a la part inferior del tronc i l'escalem a unes dimensions petites que a prou feines ocupi un espai significatiu entre el troc i les cames. Seguidament haurem de crear els ossos a partir del principal, això es fa entrant en mode edició, seleccionar el cercle superior del os creat i pulsar `e+z` per expandir un os que hereti del os principal i desplaçar-lo sobre el eix de les z, fins a la següent articulació, això s'ha fet fins omplir tot el model. Per tal de crear els ossos sense que estiguin connectats seguidament dels ossos pares es desactiva la casella de *connected* de la barra de *Bone menu outliner*.

UNIO DELS ESQUELET AMB EL MODEL

Un cop creat l'esquelet s'haurà d'aplicar *skinning*, es adir, associar els ossos amb les diferents parts del model. Per fer això es selecciona en mode objecte el model i després l'esquelet mantenint polsant `shift`, i a continuació pulsar `ctrl+P->automatic weight painting`. Blender ho fa de forma automàtica.

WEIGHT PAINTING

Un cop feta la unió , pot passar que l'efecte no sigui el desitjat degut a la complexitat del model. Per tant s'hauran de comprovar els pesos de les diferents zones del model que es corresponen amb els

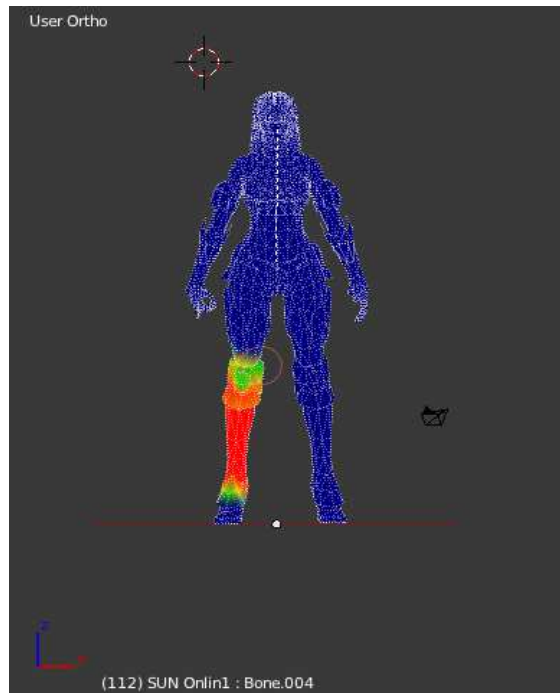
ossos, i si no es corresponen de forma correcte realitzar el pintat de forma manual. Aquest punt es crític a l'hora d'animar els personatges.

Els efectes no desitjats poden ser que una part del model no es mogui, això es per que aquesta zona del model no ha sigut influenciada per cap os o be que diferents zones del model han sigut influenciades per un mateix os. Per comprovar que tot es correcte el que s'ha fet es mirar la correcta correspondència entre model y esquelet os per os. A la imatge 42 es pot observar que la unió entre model i esquelet no es correcta, i per tant haurem de fer un *weight painting* de forma manual.



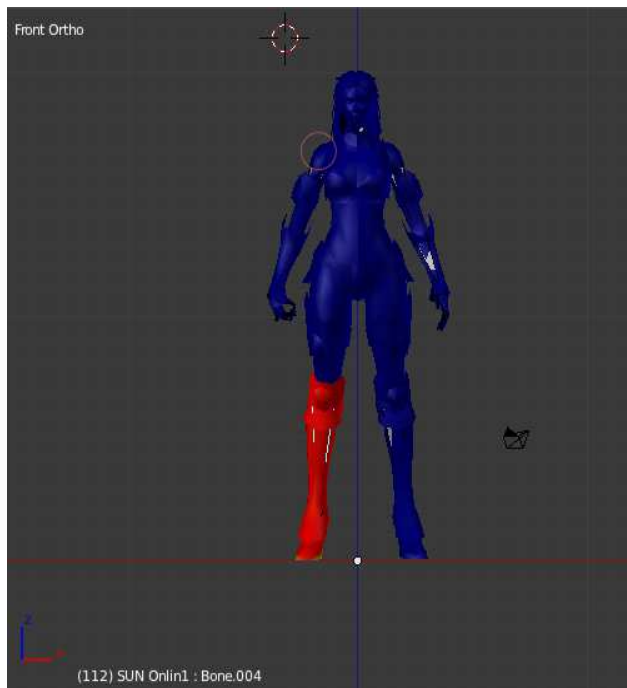
Imatge 42: Skinning fet de forma errònia

Si l'efecte no es el desitjat aplicarem el *weight painting*, ara veurem con es fa. Seleccionarem un os en *pose mode*, després seleccionarem el model i el mode *weight paint*. A la imatge 43 podem observar la zona de influencia del os de la cama a la zona on es l'os, després de haver fet l'*skinning* de forma automàtica.



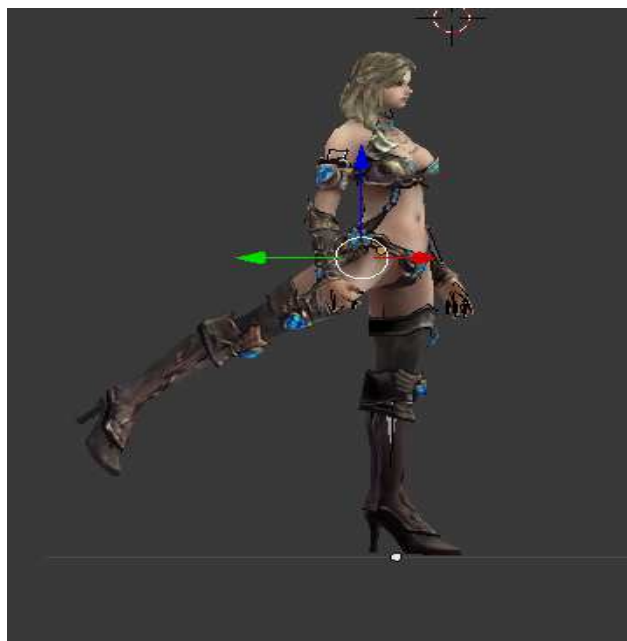
Imatge 43: Pintat automàtic fet per Blender de forma incorrecta

El model del cos canviarà per tonalitats, i es podrà veure la influència de l'os en les zones del model. Per modificar la influència disposem d'un panell on existeixen les funcions per realitzar el *weight paint*. Les opcions que he utilitzat son la de personalització de pinzell i el mode de pintat. En quant a la personalització del pinzell , he anat variant la mida per poder pintar correctament el model, i pel que fa al mode de pintat, he utilitzats les opcions de *add* i *subtract*. La funció *add* redueix la influència del os a la zona afectada i la de *subtract* fa el contrari. Per saber la zona de influència dels ossos a les diferents parts de os disposem de les diferents tonalitats a l'hora d'entrar en el mode *weight paint*. On el color vermell es la màxima influència i el color blau fosc la mínima.



Imatge 44: Modificació del pintat de pesos per la correcta posterior animació

A la imatge 44 i 45 es pot visualitzar con ha estat aplicat el pintat de pesos a la cama i el resultat d'aquesta modificació. Com podem veure a la imatge 39 he aplicat la màxima influència a l'os així els vèrtexs dels polígons que formen part d'aquesta zona no es veuran desplaçats tal com hem vist a la imatge 40.



Imatge 45: Resultat de moure la cama després de modificar el pintat de pesos

5.2.2. Creació d'una animació

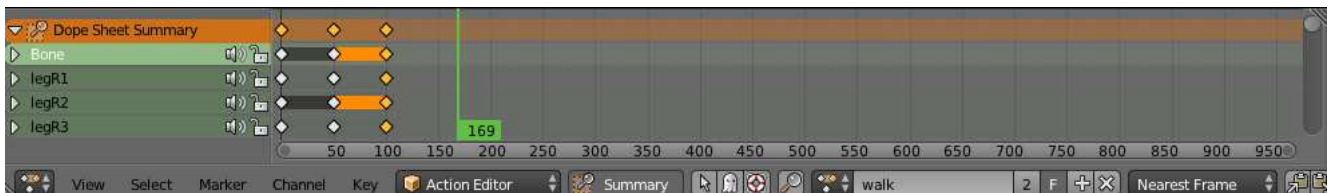
Per a crear una animació hem d'assignar la finestra de *dopeSheet*, en aquesta finestra s'ha de localitzar l'opció *Action Editor*, fent això podrem trobar el icona per donar nom a les animacions que crearem.

A la imatge 46 podem observar en el requadre vermell el boto *dopesheet* que ajustarà els temps dels *keyframes*^{*}, en el requadre verd podem veure l'opció *action editor* que editarà els *keyframes* corresponents al ossos i en requadre blau el nom de la animació que estem editant.



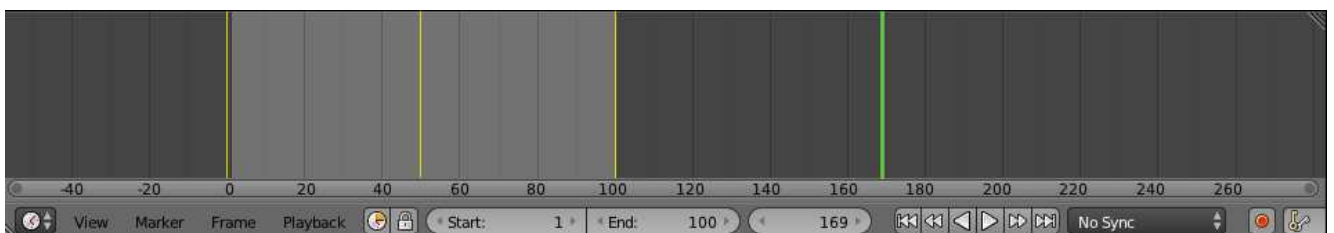
Imatge 46: Visualització de opcions de l'eina *dopeSheet*

Abans de començar a animar es necessari conèixer les diferents funcionalitats de les finestres que intervenen en el procés d'animació. En la imatge 47 es pot veure la finestra *DopeSheet*. En aquesta finestra es mostren els diferents ossos del esquelet del model en un instant determinat.



Imatge 47: Visualització dels *keyframes* corresponents al ossos

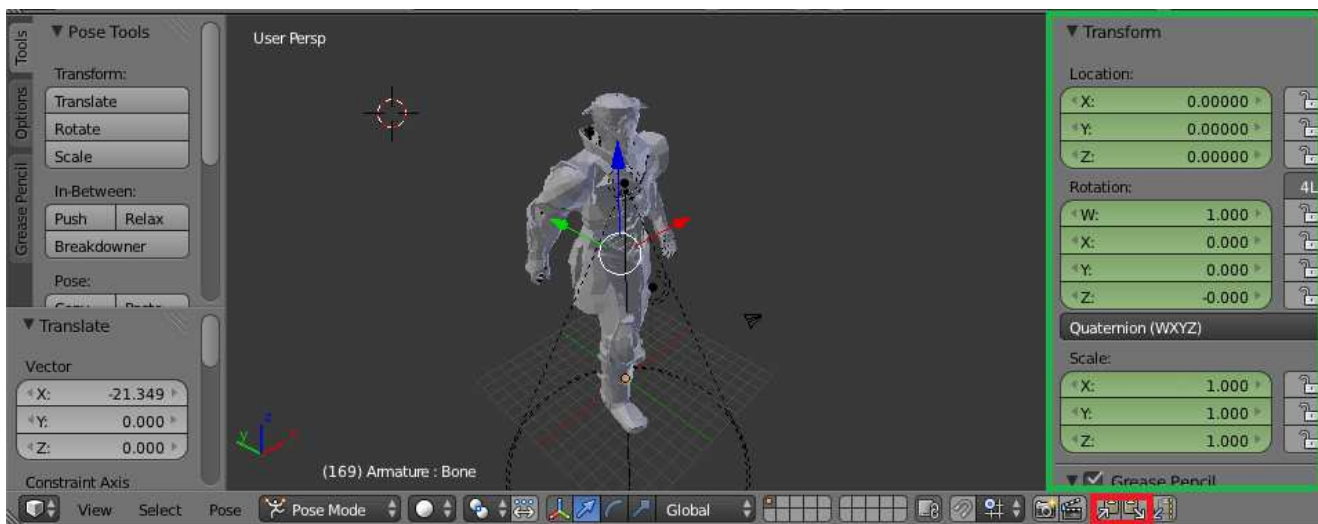
A la imatge 48 es pot veure la finestra de *timeline*, aquesta serà l'encarregada de definir els diferents *keyframes*. També es reproduiran les animacions i es podran variar els *frames* de començament i final de les animacions.



Imatge 48: Visualització de la finestra *timeline*

* Consultar al glossari de vocabulari al final del document

A la imatge 49 es pot veure la finestra de *3dview*, aquí es podran visualitzar les animacions, a més te una opció molt interessant, els botons de copiar i enganxar(requadre vermell), la funcionalitat d'aquest botons va molt bé ja que copia i enganxa las posicions dels ossos seleccionats als diferents *frames* de l'animació. Es a dir estalviarem temps en crear les nostres animacions. També es molt important l'opció *transform*(requadre verd) ja que ens permet variar les posicions dels ossos, posicions que es guardaran en els diferents *keyframes*.



Imatge 49: Visualització del model del guerrer a la finestra *3dview*

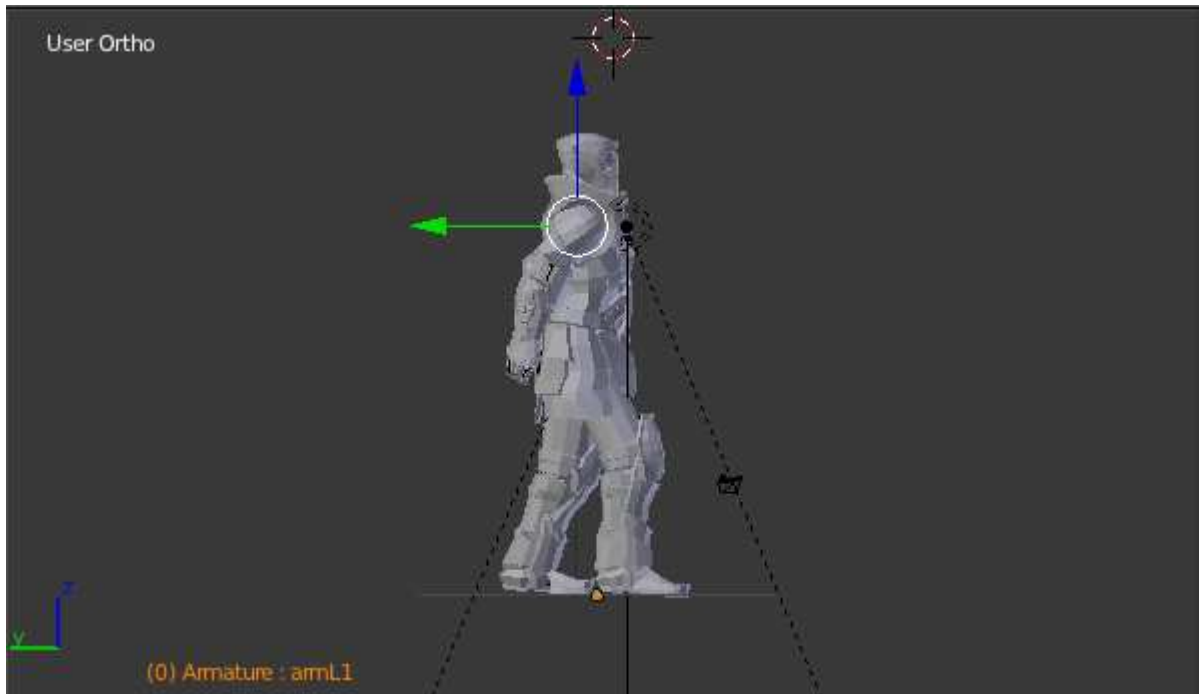
Ara ja podem crear les nostres animacions, una animació esta composta de *keyframes*. Un *keyframe* es una imatge fixa en una seqüència animada que es produeix en un punt important en un ordre donat. Per crear una animació hem de crear un numero determinats de *keyframes*, cada *keyframe* es una posició del personatge en un determinat moment de l'animació.

La primera cosa que s'ha de fer es assignar l'instant de temps en el *timeLine* corresponent als *keyframes*.

El pas següent es assignar la posició que se li vol donar al nostre model, per fer això hem d'establir el *pose mode* i seleccionar l'os o ossos als que volem variar la seva posició en aquest instant, es fa variant les coordenades de posició, rotació o escalat de cada un dels ossos del menú *transform*. Per finalitzar hem de desar els canvis de posició que s'han fet en aquest *keyframe* polsant la tecla "A", per seleccionar tots els ossos, i després polsar la tecla "I" on apareixerà un desplegable amb las opcions disponibles. Tot aquest procés es fa repetidament en cada instant de temps que es requereixi per la animació que es vol crear.

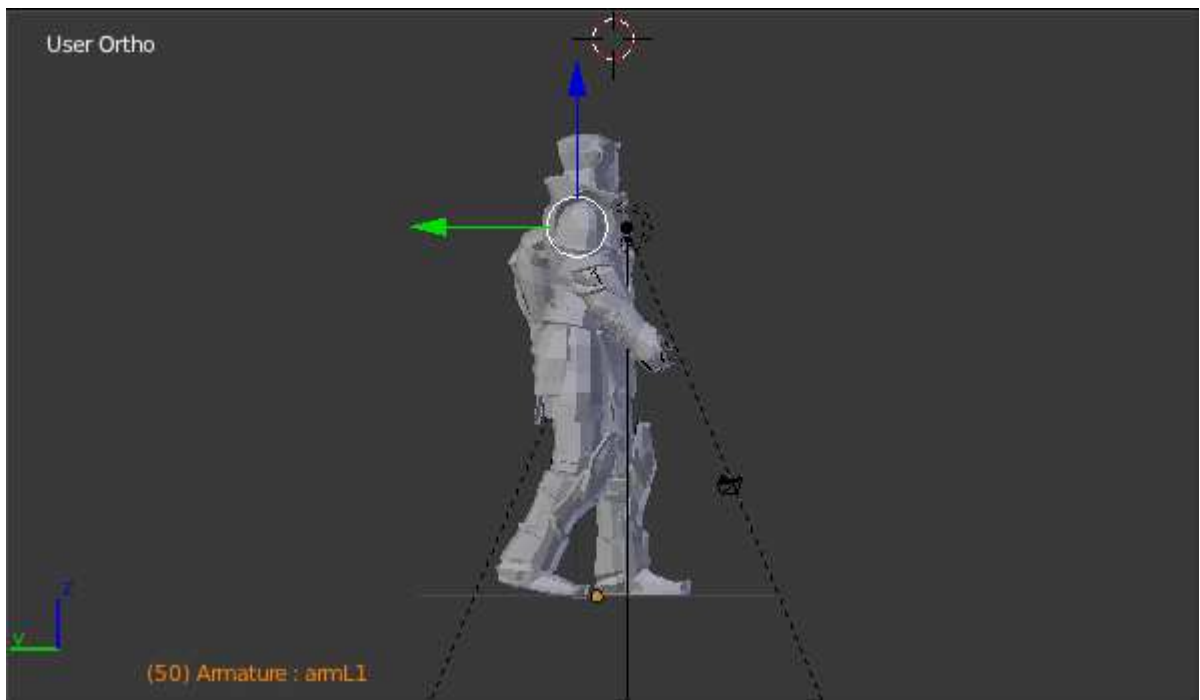
Exemple animació caminar

En aquesta animació hem assignat 3 *keyframes* en el *timeLine*, el primer *keyframe* ha estat assignat al instant 0. A la imatge 50, podem veure la posició dels ossos que li hem assignat a un dels personatges per aquest instant.



Imatge 50: Visualització del *keyframe* 1 i 3 de l'animació caminar de model del guerrer

El segon *keyframe* s'ha assignat al punt mig de l'animació al frame 50 i es invers al primer *keyframe*. A la imatge 51 es pot observar el resultat.



Imatge 51: Visualització del *keyframe* 2 de l'animació caminar de model del guerrer

El tercer y últim *keyframe* esta ubicat al frame 100, Es idèntic al primer *keyframe*. Aquesta animació dura 100 *frames*, te una bona proporció perquè l'animació sembli realista.

5.2.3. Importació dels models creats a Blender

Per importar els models es necessari haver fet la exportació dels models creats desde blender. Aquests models han sigut importats en format fbx.

Per fer la importació s'ha de crear una subcarpeta dins de la carpeta *assets** de nostre projecte, on guardar els teus objectes fbx. També s'haurà de crear una altre amb les textures dels models.

Després ens situarem dins de la subcarpeta creada i amb el boto dret del ratolí busques l'opció *import package*. Apareixerà una finestra de navegació on has de buscar el teu model, el selecciones i començarà la importació a Unity. Jo he creat varies carpetes segons el objectes creats, per exemple per els personatges principals, he creat una carpeta PJ amb subcarpetes que corresponen als diferents personatges principals. A la imatge 52 es pot veure l'estructura utilitzada dels *assets* corresponents als meus models.

* Consultar al glossari de vocabulari al final del document

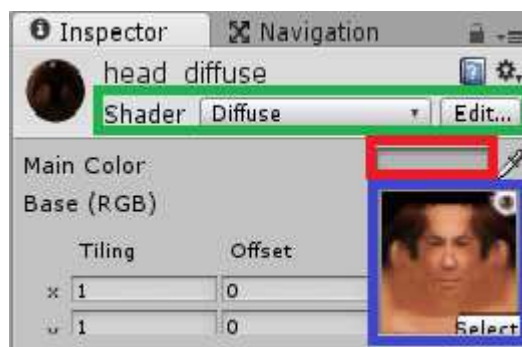


Imatge 52: Estructura dels assets

Després de que Unity hagi finalitzat la importació s'haurà creat una carpeta materials amb els materials importats del objecte i l'objecte del model que s'utilitzarà a Unity.

APLICACIO DE TEXTURAS A UNITY

Unity no ens assignarà les textures al models de forma automàtica, ho tindrem de fer manualment. Anirem a la carpeta de materials que s'ha creat prèviament de forma automàtica, seleccionarem un material de la llista, i després ens situarem al inspector. En el inspector tenim les opcions de canviar el color del material, escollir la textura desitjada per aquest material i modificar el tipus de efecte del material. A la imatge 53 podem observar en el requadre verd l'opció del tipus d'efecte pel material, en el requadre vermell l'opció de canviar el color del material i en el requadre blau l'opció de escollir la textura desitjada.



Imatge 53: Visualització del keyframe 1 de l'animació caminar de model del guerrer

Un cop fet això per tots els material del que estan compostat l'objecte importat, podem veure l'estat final a la finestra *preview* de la pestanya model del inspector corresponent al objecte importat. A la imatge 54 podem veure aquest estat final corresponen al objecte warrior1.

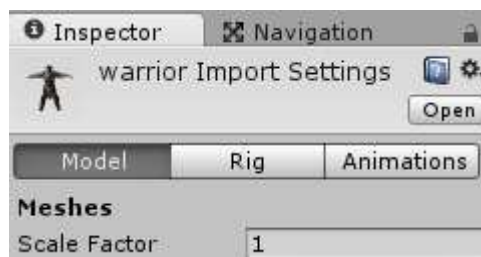


Imatge 54: Estat final del objecte després de aplicar tots els passos de la importació

MODIFICACIO DE LA MIDA DEL MODEL Y APLICACIO DE ANIMACIONES A UNITY

Es seleccionará l'objecte importat ,en la finestra inspector apareixeran tres pestanyes. Aquestes pestanyes son model, *rig* i *animations*.

A la pestanya model haurem de modificar el valor de la propietat *scale factor* a 1.Després de la importació Unity per defecte ens posa aquest valor a 0.01, el que fem canviant aquest valor es augmentar la mida del model 100 cops mes gran, d'aquesta forma la mida del model es la mateixa que en Blender. A la imatge 55 podem veure les pestanyes corresponents al objecte seleccionat i la modificació del *scale factor*.



Imatge 55: Visualització de les diferents opcions del inspector corresponents a un model

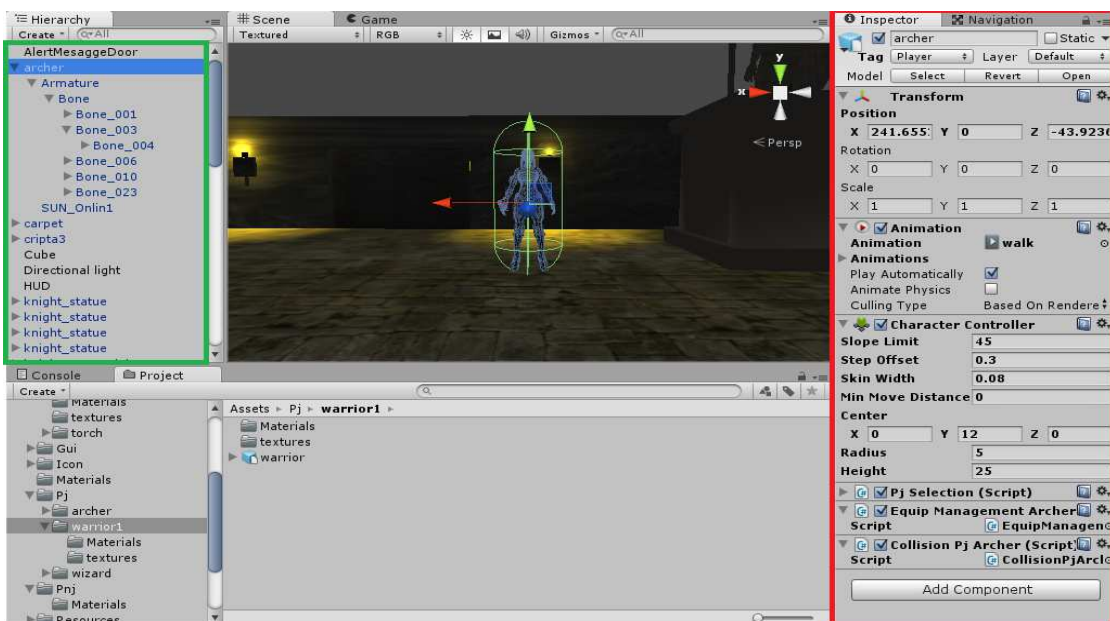
Per poder utilitzar les animacions fetes a Blender en els objectes importats haurem d'aplicar els següents passos:

A la pestanya *Rig* haurem d'escollir el tipus de animació del rig del model, la opció que s'ha de escollir es la de *legacy*, per tal de poder utilitzar les animacions del objecte importat

A la pestanya *Animations*, s'observaran totes les animacions disponibles del model, aquestes animacions son les que hem creat a Blender. Podrem canviar el temps d'inici i final de les animacions, es podrà canviar l'animació que s'executarà per defecte un cop afegit l'objecte a l'escena. També es podrà veure l'execució de l'animació a la finestra *preview* d'aquesta pestanya.

5.3. Utilització dels objectes a unity

Un cop ja tenim els nostres objectes importats llestos per poder utilitzar en el *engine*^{*}, el que fem es arrossegar l'objecte cap a l'escena. Al arrossegar els objectes a l'escena, es visualitzaran a la finestra de jerarquies i des de aquí poder accedir al inspector per modificar els diferents valors dels components que posseeixen. Això també es pot fer mitjançant *scripts* per dotar als objectes de les funcionalitats que desitgem en el nostre joc. A la imatge 56 podem veure al requadre de color verd la finestra de jerarquies dels diferents *GameObjects* que son a la escena i de color vermell els diferents components del *GameObject* seleccionat.



Imatge 56: Visualització de la finestra de jerarquies i inspector

* Consultar al glossari de vocabulari al final del document

A part de poder importar models que després es podran utilitzar com a objectes a Unity, anomenats *GameObjects*, també podem crear *GameObjects* des del editor de Unity accedint a la barra de menú *GameObject->createOther*.

A Unity la utilització dels *GameObjects* i els components que formen es bàsica per dotar de comportament al joc. Els *GameObjects* i els seus components poden ser modificats, creats i destruïts mitjançant scripts o bé mitjançant l'editor de Unity. És necessari que els objectes tinguin aquells components que seran modificats per un *script*, si no, no es podrà accedir a aquest component.

Per poder accedir a un *GameObject* i poder modificar les components d'aquest objecte es necessari que tinguin un nom i un tag*. A la imatge 57 podem veure que el nom del *GameObjects* *archer* i el seu *tag* es *Player*, el demes personatges principals com son el guerrer i el mag també tindran el mateix *tag*. Per accedir des de *script* a un *GameObject* pel seu nom s'utilitza:

```
private GameObject neutral;  
neutral = GameObject.Find("pnj_neutral");
```

Imatge 57:codi per accedir un *GameObject* mitjançant el mètode Find

També es pot accedir pel seu *tag* amb el mètode *FindObjectWithTag* o si es vols accedir a varis *gameObject* que tenen el mateix *tag* i guardar aquests *GameObjects* en un vector es fa pel mètode *FindObjectWithsTag*.

A part de els objectes importats a unity hem utilitzat altres *GameObjects* com els *ParticleSystem*, la *Camera*, *GuiTextures* i *Lights*. En el punt 3.3.2.1 s'ha explicat la utilització dels *GameObjects* sistema de partícules i *Lights* en l'escenari. En punts posteriors explicaré l'us del *GameObject* *Camera* i *GuiTextures*.

5.3.1. Components utilitzats

Collider

Un *collider* es la estructura que fa sòlids als objectes, un objecte que tingui un *collider* toparà amb els elements de l'entorn. També es poden utilitzar com a *triggers*, es necessari que el *collider* que s'utilitzi tingui la forma mes semblant possible al *GameObject* al que se li aplica.

Existeixen diferents tipus de *Colliders*, a la imatge NUM podem veure els diferents tipus.

* Consultar al glossari de vocabulari al final del document

S'ha utilitzat els *colliders* en la arquitectura del joc, així com als *GameObjects* *swith A i B*, que funcionaran com a *triggers* per poder obrir les portes que donen accés al salo. Aquets *triggers* si son activats faran que s'obri la porta de la cripta. També hi ha un *trigger* a un *GameObject* ocult que quan els personatges entren dins d'aquest surt un missatge pel *TextArea*.

Un *trigger* es una propietat dels *colliders*, Per a que un *collider* sigui un *trigger* s'ha d'activar en el editor de Unity o be fer-ho mitjançant *scripting*. Els *triggers* son activats quan un *collider* topa amb ells. A la imatge 58 podem veure com el *CharacterController* del arquer al entrar al *collider* del objecte *AlertMessageDoor* s'activa el *trigger* fent que surti un missatge pel *textArea*.



Imatge 58: Mostreig d'un missatge al log després d'activar un *trigger*

Els *triggers* tenen 3 formes per ser activats quan un *collider* entra en contacte amb ells, es necessari fer-ho mitjançant *scripting*.

- *OntriggerEnter*: Aquest missatge s'activa quan un *collider* entra en contacte amb el *trigger*.
- *OntriggerStay*: Aquest missatge s'activa quan un *collider* es dins del *trigger*.
- *OntriggerExit*: Aquest missatge s'activa quan un *collider* surt del *trigger*.

A la imatge numero 59 podem veure un exemple d'us amb el mètode *OnTriggerEnter*.

```
void OnTriggerEnter(Collider other){
    if(transform.collider.name == "switchB"){
        switchBTouch = true;
    }
}
```

Imatge 59:Exemple d'us de la activació d'un *trigger*

Per a mes informació consultar l'API de Unity corresponent a [Collider](#).

Rigidbody

Un *rigibody* es un conjunt de dades que permet calcular las conseqüències que tindrà una col·lisió per el *gameObject* al cual s'assigna un determinat *rigibody*.

Per a que Unity calculi las conseqüències d'una col·lisió necessita tindre en conta factors com la massa del *GameObject* al que se li vincula la velocitat a que es produeix la col·lisió, si existeix fregament i en quina mida, si hi ha o no gravetat,etc. Resumidament, un *rigibody* dota a un *GameObject* de propietats físiques com la gravetat, massa i pes. Ja que si no es sap la massa d'un objecte no es pot calcular el resultat d'una col·lisió.

S'ha utilitzat un *rigibody* al *GameObject* MagicBullet. Aquest *GameObject* es un sistema de partícules que representa l'encanteri d'un projectil màgic llençat pel mag, utilitzo el *rigibody* per moure aquest *GameObject* per l'escena fins a topar amb el *collider* d'un enemic. En apartats posteriors explicaré amb mes detall el funcionament d'aquest encanteri.

Per a mes informació consultar l'API de Unity corresponent a [Rigidbody](#).

CharacterController

El *CharacterController* actua com un *collider* que s'aplica als personatges del joc. El *CharacterController* no es veu afectat per las forces i solsament es mourà quan es cridi a la funció *move*. El *CharacterController* pot detectar col·lisions entre altres *colliders*.

Hem utilitzat una configuració diferent en els diferents *CharactersControllers* dels personatges en el que es refereix al radi del *CharacterController*.

S'ha utilitzat aquest *collider* per moure els personatges principals i als enemics per l'escenari. Mes endavant s'explicarà com.

Per a mes informació consultar l'API de Unity corresponent a [CharacterController](#).

Transform

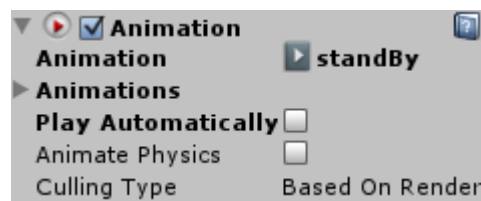
Aquest component ens indica els valors de rotació, posició i escalat d'un *GameObject*. El *transform* contindrà la rotació, posició i escalat actual d'un *GameObject* a l'escena.

Aquest component es essencial per efectuar els moviments i rotacions dels diferents objectes que hi ha a la escena.

Per a mes informació consultar l'API de Unity corresponent a [Transform](#).

Animation

Aquest component controla las animacions del *GameObject* si aquest te alguna animació. A la imatge 60 es pot veure les característiques del *animation*. Aquestes característiques es poden observar al editor de Unity en el inspector dels *GameObjects*.



Imatge 60:Opcions del component Animation corresponent a la finestra de inspector

En els *scripts* les animacions solsament podran ser cridades per un *GameObject* sempre que desponguin d'elles. Si aquest es el cas es pot reproduir una animació de la forma que es veu a la imatge numero 61.

```
Animation animation = pj.getTarget().animation;  
animation["walk"].speed = 5;  
animation.Play("walk");
```

Imatge 61:Codi per reproduir una animació amb una determinada velocitat

En aquest codi a la primera línia s'assigna l'animació d'un *gameObject* a la variable de tipus *Animation*, `getTarget()` es un mètode que retorna el *GameObject* assignat al objecte anomenat `pj`, A la segona línia es modifica la velocitat de l'animació *walk* i a la darrera es reproduceix l'animació.

Per a mes informació consultar l'API de Unity corresponent a [Animation](#)

AudioSource

Aquest component permet reproduir un clip d'àudio, afegir un *audio source* a un *GameObject* converteix a aquest objecte en un emissor de so. S'ha afegit un so a la càmera perquè reproduïxi el so ambiental de la cripta, de forma constant, això es fa activant la casella de *loop* en el inspector del objecte, si aquesta casella no fos activada es reproduiria solsament un cop. També s'ha fet que es reproduïxi res més carregar l'escena activant la casella *play awake*. A la imatge 62 es pot veure com s'ha introduït sons des de l'editor de Unity.



Imatge 62:Opcions del component *audioSource*

S'ha posat sons als personatges principal a l'hora de realitzar els encanteris i a l'explosió de l'encanteri de projectil màgic, aquests sons s'activaran de forma automàtica cada cop que s'instancia el nou efecte de partícules corresponent al encanteri i a l'explosió, per tal de aconseguir-ho en aquest cas s'activa la casella *play on awake*.

S'ha afegit so al personatge neutral i a una de les portes, aquets dos sons son controlats per *.script*. El personatge neutral emet el so d'un cop de porta quan et localitza, el so que reproduïx una de les portes es la porta mentre s'obre. A la imatge 63 es pot veure com es pot reproduir un so per *scripting* utilitzant el mètode *Play*.

```
ch.getTarget().audio.Play();
```

Imatge 63:codi per reproduir un determinat so

Per a més informació consultar l'API de Unity corresponent a [AudioSource](#).

NavMeshAgent

Aquest component permet que un objecte navegui per l'escenari utilitzant *NavMeshAgent*. S'ha utilitzat aquest component per moure el personatge neutral per l'escenari. La funció *setDestination* permet fer que els objectes puguin dirigir-se als punts assignats.

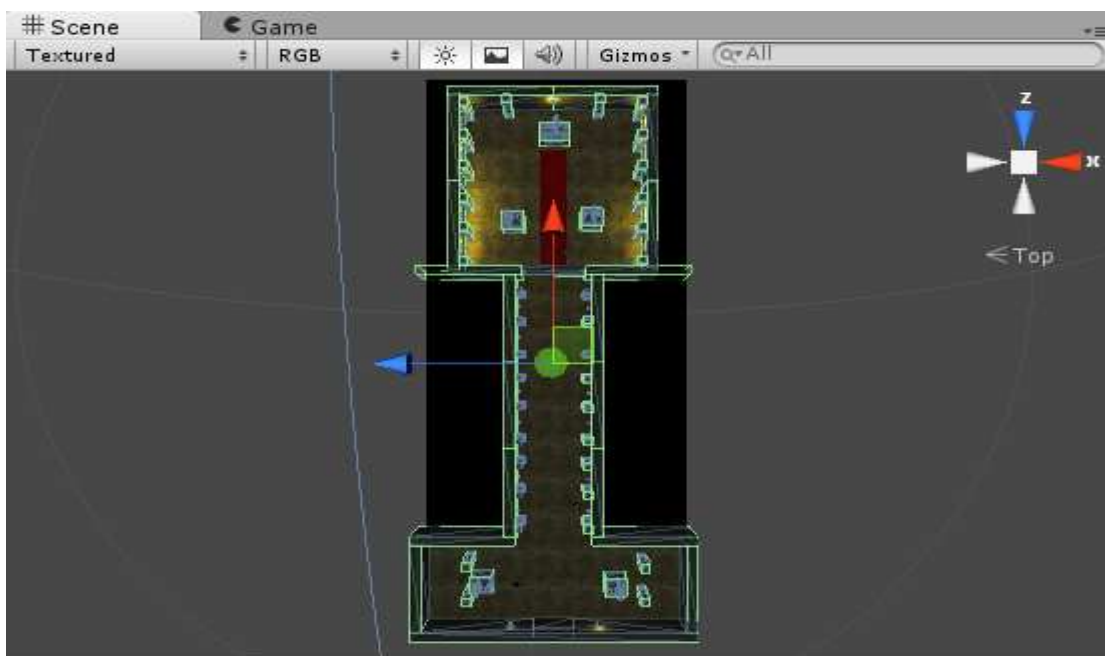
Per a més informació consultar l'API de Unity corresponent a [NavMeshAgent](#).

5.4. Funcionament dels GameObjects al joc

En aquest punt es parlarà de com funcionen els diferents *GameObject*s creats en el joc i dels *scripts* assignats a ells.

5.4.1. Escenari Principal

El terreny com ja he dit, es un *GameObject Terrain*, aquest terreny te un *terrain Collider*. Per tal de limitar els moviment dels personatges a dins de la cripta, he afegit *box colliders* als diferents elements que la formen, es a dir, a les parets, portes, estàtues,etc. Així els personatges no traspassaran aquests elements. A la imatge 64 es pot veure la disposició dels *colliders* de la cripta i del altres *GameObject*s que formen part del escenari.



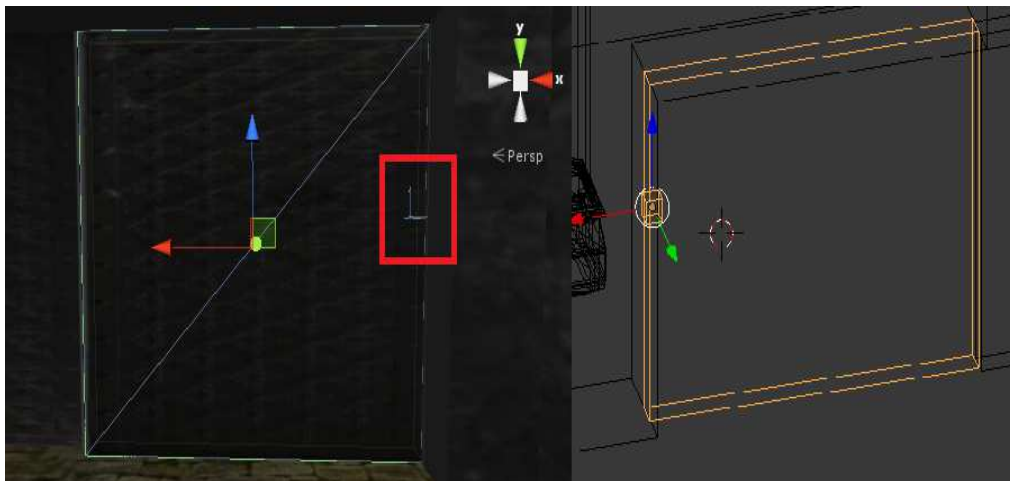
Imatge 64: Disposició dels *colliders* de la cripta

Gracies al *collider* del terreny i als diferents *colliders* podem moure els nostres personatges per l'escenari ja que utilitzo la tècnica de *Raycasting*. El que es fa es tirar un raig des de la càmera contra

un punt de la pantalla, aquest punt de la pantalla es la posició del ratolí, per tant en tot moment dispo de la posició a la que vull moure els personatges pel terreny. Amb aquesta tècnica dispo de la informació de tots els altres *colliders* de la escena. Així si es vol anar fins una posició on esta ubicada una columna es podrà anar. Els *scripts* que intervenen en el moviment dels personatges son: *MotionControllerPj*, *GoControllerPj* i *PjSelection*, després seran explicats amb mes detall.

En quant a les portes que formen part del *GameObject* cripta, per tal de que s'obrin i poder accedir al salo principal, el que s'ha fet mitjançant *scripting* es obrir les portes si es te contacte amb els *GameObjects* *SwithA* i *SwithB*, aquests dos panells, son dos *triggers* que faran que s'executi l'*script* que obre la porta. Els *scripts* *openDoor* i *ActivateSwithA* donen lloc a aquesta funcionalitat. El funcionament es el següent:

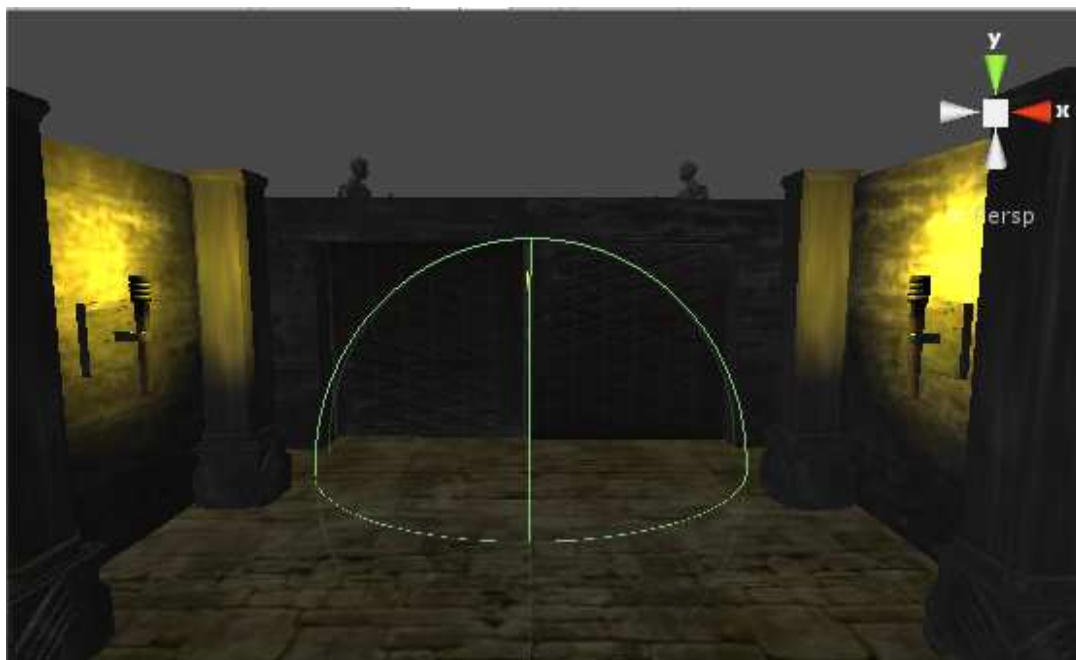
En el *script* *ActivateSwithA*, mitjançant la funció *OnTriggerEnter* es canvia el valor d'una variable booleana a cert, aquesta variable es correspon al fet de que s'hagi activat el *SwithA*. El *script* *openDoor* fa el mateix que el *script* *ActivateSwithA*, però amb la diferencia que en el mètode *update* es va mirant si el valor de la variable booleana associada al *SwithA* esta a cert i si la variable booleana associada al *GameObject* *SwithB* es certa s'obriran les portes, les portes s'obren mitjançant una rotació aplicada a un extrem de la porta mitjançant la funció *Rotate* del component *Transform* propi de cada *GameObject* corresponen a les portes (element de la cripta anomenat *bisagra1_000* i *bisagra1_002* i). A la imatge 65 podem observar, l'element que estem rotant.



Imatge 65: Element utilitzat per obrir la porta tant en Unity com en Blender

S'ha creat un *GameObject* buit anomenat *AlertMessageDoor* que dona informació al usuari de com obrir la porta, aquesta informació evidentment la dona mitjançant *scripting*, aquest *script* s'executa mitjançant un *trigger* i donarà la informació als personatges si la seva tirada es major a una valor per defecte. El *script* *ThrowMessageDoor* dona lloc a aquesta funcionalitat, bàsicament el que fa es mirar si el *collider* que entra en contacte amb el *trigger* correspon a un personatge principal, en aquest cas el personatge o personatges que hi siguin al *trigger* faran una tirada, si aquesta tirada supera un valor

determinat es mostrarà per *textarea* un missatge, aquest missatge s'envia al *script* Hud. A la imatge 66 podem veure el *collider* corresponent al *gameObject* *AlertMessageDoor*.



Imatge 66: Collider corresponent al *gameObject* *AlertMessageDoor*

5.4.2. Inici del joc

Un cop carregada l'escena principal el *script* *SearchCharacter* s'encarregarà de instanciar cada un dels objectes que pertanyen a la classe *Character*, per fer això en el cas dels personatges principals es buscaran els *GameObjects* en l'escenari, s'instanciaran els objectes a partir d'aquests *GameObjects* i es ficaran dins d'un *arrayList* de *Characters*, els enemics es ficaran també en el mateix *arrayList* però es carregaran de forma dinàmica a partir dels prefabs *pnj* i *pnj_range* de la carpeta *Resources*, s'ha optat per carregar-los dinàmicament ja que el número d'enemics depèn de la dificultat del joc que ve donada per la selecció del menú de opcions, al punt 3.3.6.7 es parlarà de com s'assigna la dificultat en el joc. En quan al personatge neutral també es buscarà el seu *GameObject* associat en l'escena i es ficarà en un *arrayList*.

5.4.3. Moviment y Controls dels personatges principals

Als *scripts* que tenen que veure amb la interacció dels personatges amb l'escena son *GoController* *PJ*, *MotionControllerPJ*, *PJSelection*, *Hud* i *StopTime*.

La interacció entre aquests *scripts* es la següent, PJSelection, establirà els diferents personatges seleccionables per teclat si es volen seleccionar tots els personatges, o mitjançant el ratolí polsant sobre els *CharactersCollider* de cada personatge si cap personatge ha estat escollit com a receptor d'algun encanteri del mag. Amb el *script* GoControllerPJ el personatge o personatges que s'hagin seleccionats podran ser moguts per les diferents parts del escenari que farà el usuari mitjançant *Raycast* com s'ha explicat anteriorment . Aquest *script* també es el encarregat de fer el quadre de selecció. Posteriorment el *script* MotionControllerPJ s'encarregarà de dotar de moviment els personatges. Per el que fa el Hud s'encarregarà de seleccionar els personatges si es seleccionen els *portraits* de cada personatge, així com seleccionar els diferents encanteris o habilitats del personatge seleccionat, GoControllerPj també controlarà quin personatge ha estat seleccionat en cas d'haver escollit un encanteri que necessiti d'un *collider* receptor per tal d'executar l'encanteri. El *Script* StopTime serà el encarregar d'aturar el temps, si el temps ha estat parat es podrà fer totes les accions descrites. A continuació explicaré amb mes detalls aquests *scripts*, excepte del HUD que ja parlaré mes endavant.

Script PJSelection

Mitjançant aquest *script* podem saber quin personatges han sigut seleccionats, aquesta informació serà enviada al *script* GoControllerPJ i MotionControlerPj per saber fins a on es mouran els personatges i moure'ls respectivament, mitjançant un enter sabrem quin personatge ha sigut seleccionat. Si s'han seleccionat tots els personatges polsant la tecla "A" s'enviarà un 0, en cas d'haver fet la selecció mitjançant el quadre de selecció s'enviarà un -1 i en cas de seleccionar els *colliders* dels personatges s'enviarà un 1,2 o 3 depenent de personatge seleccionat. Aquí també es fa aparèixer el *GameObject* SelectorPjs que correspon a la aureola que rodejarà al personatge o personatges un cop seleccionats.

Des d'aquí també es seleccionaran els personatges receptors del encanteri ArmorSpell, que el Hud processarà per activar l'encanteri al personatge seleccionat.

Per saber si s'han seleccionat tots els personatges, utilitzarem una funció de Unity per detectar una entrada per teclat.

```
Input.GetKeyDown(KeyCode.A)
```

Imatge 67:Codi utilitzat per saber si s'ha polsat la tecla "A"

En quant a la selecció individual dels personatges s'ha utilitzat la funció OnMouseUpAsButton. Aquesta funció es cridada quan el ratolí es alliberat sobre un *collider* o *GuiElement*. Per saber quin *GameObject* ha sigut seleccionat utilitzem :

```
(transform.gameObject.name == "warrior")
```

Imatge 68:Codi per saber el nom d'un objecte

I assignem a una variable l'enter corresponent al personatge seleccionat, per fer aparèixer la aureola corresponent al personatge seleccionat utilitzarem

```
pj.getSelectionPj().renderer.enabled = true;
```

Imatge 69:Codi per generar la imatge corresponent a la seleccio d'un personatge

`getSelectionPj` es un mètode que retorna el *GameObject* corresponent a la aureola de cada instància de *Pj*, mitjançant `renderer.enabled` fem que aquest objecte es torni visible o no. Així doncs un cop seleccionat un personatge fem desaparèixer les aureoles dels altres personatges i fem aparèixer la del Personatge seleccionat. Utilitzem un mètode anomenat `caughtPj` per saber a quina instància ens referim segons el personatge seleccionat.

Per saber si un personatge a estat seleccionat per rebre un encanteri o no utilitzem un mètode anomenat `spellActivate`, aquest mètode retorna un boolea, aquest boolea es el retorn del mètode `getStop`, mètode propi de la classe *Wizard*, el atribut de classe que retorna correspon a si el usuari selecciona un encanteri.

Script GoController PJ

Aquest *script* s'encarregarà d'assignar les posicions on aniran els personatges seleccionats i de calcular les rotacions dels personatges respecte el punt seleccionat. També en aquest *script* es generarà el quadre de selecció i s'establirà quins personatges han sigut seleccionats en aquest quadre.

Per a calcular els angles de rotació el que s'ha fet es el següent, cada objecte *Pj* tindrà un *GameObject* anomenat `internalCompass`, aquest objecte funciona com una brúixola que apunta sempre a la direcció on es troba el ratolí quan el personatge esta seleccionat. Per saber el angle que s'assignarà a la brúixola, generem un pla virtual al centre del personatge sobre l'eix y.

```
Plane plano = new Plane(Vector3.up, pj.getCompass().transform.position);
```

Imatge 70:Codi que instancia un pla

Generem un raig des de la càmera fins a la posició del ratolí i mirem si aquest raig interseca amb el pla virtual creat.

```
if(plano.Raycast(ray_,out hitDist))
```

Imatge 71:Codi utilitzat per generar un raig

En cas positiu obtenim las coordenades de la posició del ratolí sobre el pla virtual i gracies a aquestes coordenades podem calcular la rotació que farà la brúixola utilitzant la direcció de la posició del plànol virtual i la direcció que apunta la brúixola virtual :

```
Vector3 posPlano =ray_.GetPoint(hitDist);  
rotarWarrior = Quaternion.LookRotation(posPlano - pj.getCompass().transform.position);
```

Imatge 72:Codi que calcula la coordenades per aplicar un rotacio

Després apliquem la rotació a la brúixola mitjançant una interpolació:

```
pj.getCompass().transform.rotation = Quaternion.Slerp(pj.getCompass().transform.rotation,rotarWarrior,Time.deltaTime * 10);
```

Imatge 73:Codi que calcula la rotació que farà la brúixola interna dels personatges

Abans de establir el punt on volem moure el nostre personatge, haurem de veure si volem fer una selecció mitjançant un quadre de selecció i si esta parat el temps per poder generar el quadre de selecció un cop parat. Per tal de poder fer això hem de veure si el ratolí ha estat posat,deixat de polsar o be esta sent polsat. Per mirar si el ratolí ha estat polsat,deixat de polsar o s'està polsant utilitzem les funcions de Unity GetMouseButtonDown, GetMouseButtonUp i GetMouseButton respectivament.

Si hem polsat el ratolí i el temps esta parat calculem el tems real de joc, aquest temps seguirà contant encara que el joc sigui en pausa, també calculem la posició del ratolí un cop polsat, aquesta posició serà un dels vèrtexs del quadre de selecció. Si el temps no esta parat el que calculem es el temps de joc, aquest temps si que deixa de contar un cop parat el temps, també calculem el punt del vèrtex inicial del quadre de selecció.

A la imatge 74 utilitzem la funció time de la classe time per mirar el temps real, aquest temps parará si es pausa el joc, en canvi a la funció realTimeSinceStartUp el temps no parará de comptar si el joc s'atura.

```

if(Input.GetMouseButtonDown(0)) {
    Cursor.SetCursor(null, Vector2.zero, cursorMode); //res
}

if(Time.timeScale != 0f) {
    oldTime = (float)Time.time;
    initSelectionSquarePosition = Input.mousePosition;
}
else{
    oldTimeInPause = Time.realtimeSinceStartup;
    initSelectionSquarePosition = Input.mousePosition;
}
}

```

Imatge 74:Utilització dels mètodes de la classe Time pròpia de Unity

Ara el que es comprova es si el ratolí esta sent polsat, aquí el que fem es calcular un nou temps tant si el joc esta parat com si no. Si el temps esta parat mirem si la diferencia entre els temps calculat quan hem polsat el boto del ratolí i el temps calculat mentre l'estem polsant es major que una determinada unitat de temps, es dibuixarà el quadre de selecció i s'actualitzaran els personatges seleccionats que son dins del quadre.

Per últim mirem si el boto del ratolí s'ha deixat de polsar, en aquest cas s'elimina el quadre de selecció i es reinicialitzen els valors del diferents temps. Es comprova que la diferencies dels temps siguin menors que una unitat de temps i es mira quins personatges estan seleccionats. Utilitzem *raycast* per saber a quin punt volem moure els personatges, aquest punt es guarda en variables tipus *Vector3* si s'han escollit independentment els personatges o en vectors si s'han escollit tots els personatges o una part d'ells, i mourem aquells personatges que han estat seleccionats. Pot ser que s'hagi seleccionat un enemic, si ha passat això mirem si el *collider* seleccionat es un *GameObject* anomenat *SeguimientoPnj*, aquest *gameObject* es atribut de la classe *Pnj*, llavors el que es fa es buscar el *Pnj* que te aquest atribut i s'assigna la posició d'aquest *Pnj* com a punt al que anirà el nostre personatge, això en cas de que sigui un guerrer a o un mag, l'arquer no anirà a la posició del enemic ja que atacarà a llarga distancia, també s'assignarà els personatges principals que hagin seleccionat al *collider* del enemic com a personatge que seguirà el enemic per a lluitar,aquests personatges es guardaran a una llista de enemics pròpia dels objectes de la classe *Pnj*.

Per mirar quins colliders han sigut seleccionats s'utilitza el mètode *colliderSelected*, a part de mirar si el *collider* seleccionat ha sigut el d'un enemic, mirem si el seleccionat ha sigut el de un dels nostres personatges principals i posem el atribut *stop* del *wizard* a fals, això vol dir que ja ha fet un encantament i ja pot realitzar el seu moviment de forma normal. També es fa el mateix pel terreny en cas d'haver seleccionat una habilitat o un encanteri de qualsevol personatge per tal de que es puguin moure després.

El quadre de selecció forma part de HUD de joc, per fer-ho necessitem de las posicions dels quadres calculats quan es polsa el boto del ratolí, si es mante polsat el ratolí, es quan cridem als mètodes `drawSelectionBox` i `updateSelectionBox`.

El mètode `drawSelectionBox` dibuixa el quadre de selecció, el que es fa es comprovar primer si s'ha instanciat un *GameObject* anomenat `SelectionBox`, en cas negatiu, es crea aquest *GameObject*, aquest *GameObject* conte un *GuiTexture*, la mida del *GuiTexture* es modifica utilitzant la variable `pixelInset`. Utilitzem las posicions del quadres i una mida de 1x1, per començar a dibuixar el quadre de selecció.

Si ja s'havia instanciat el *GameObject* el que es fa es modificar la grandària d'aquest mitjançant la posició del ratolí, m'entres el boto esta sent polsat.

El mètode `updateSelection` el que fa es incloure els *GameObjects* que son dins del quadre de selecció. Això es fa mitjançant `raycast`. En el cas de que el quadre de selecció no seleccioni cap personatge del joc s'eliminaran els *GameObjects* de la llista. També es visualitzaran les aureoles dels personatges seleccionats en el quadre. Per mirar si els personatges son al quadre de selecció s'haurà de transformar les coordenades de mon real de la posició dels personatges seleccionades a coordenades de pantalla ja que la *GuiTexture* funciona amb coordenades de pantalla. A la imatge X es pot veure com es canvia de coordenades de mon a coordenades de pantalla l'eix x de la posició del personatge.

```
(Camera.main.WorldToScreenPoint (pj.transform.position) ).x
```

Imatge 75:Codi que transforma les coordenades de mon a les de pantalla

Script MotionControllerPJ

En aquest *script* es gestiona els moviments dels personatges, bàsicament s'actualitza la posició a la que es vol moure, posició que s'assigna al *script* `GoController`. Es te en compte que si es selecciona a un personatge i aquest no ha acabat el seu moviment l'acabi tot i que estigui un altre personatge seleccionat, això s'aconsegueix mirant si els personatges han acabat el seu moviment sense que estiguin seleccionats.

S'ha de tindre en compte varis factors en el moviment, es a dir, un personatge que hagi seleccionat a un enemic per atacar-lo es moure fins a arribar a ell, excepte si un arquer que es para una determinada distancia segons el seu rang d'atac. També es te en compte que un personatge es parará de moure's si activa un encanteri o habilitat.

En aquest *script* també es fan les rotacions, en el cas de que un personatge segueixi a un enemic. Varia una mica la forma en que s'aplica la rotació perquè en aquest cas no es llança cap raig,aquí l'angle que rotarà el nostre personatge es basa en la posició del enemic del nostre personatge.

Per moure als nostres personatges utilitzem la funció Move pròpia del *CharacterController* de la següent forma, calculem la diferència entre la posició on volem anar i la posició en la que estem ubicats, si el resultat d'aplicar el mètode magnitude de Unity a aquesta diferència es menor que 1el nostre personatge es mourà.

En aquest mateix *script* també es posen les animacions al estat d'animació 0. Aquest estat correspon a la animació d'aturada dels personatges. Aquesta variable estat de animació es atribut propi dels personatges controlables i dels enemics. Es posaran a estat 0 quan els personatges arribin a la posició seleccionada i que no se estigui reproduint cap animació en el moment de estar a la posició seleccionada.

Scripts CollisionPjWarrior, CollisionPjWizard, CollisionPjArcher

En aquestes *scripts* es mira que els *CharacterController* topin contra els *colliders* dels escenaris mitjançant el mètode de Unity *OnControllerColliderHit*. Si això passa es posarà la variable corresponent a les *collision* de la classe *MotionControllerPj* com a certa. En el mètode *Move* de la classe *MotionControllerPj* es farà que s'aturin els personatges al veure la variable a certa. Es tornaran a posar a *true* quan es mogui el personatge de nou.

5.4.4. Els enemics

El *script* que controla als enemics es el *pnjController*. En aquest *script* es controla el moviment dels enemics. Els enemics inicialment seguiran una ruta predefinida, aquestes rutes estan predefinides en el *script* *SearchCharacters* mitjançant punts, es a dir els enemics es mouran entre dos punts, i deixaran de seguir aquesta ruta quan visualitzin a un enemic, per anar cap a ells en el cas de ser instancies de la classe *PnjShort* o be començar a atacar des de una posició llunyana si son instancies de la classe *PnjRange*.

Script PnjController

El funcionament d'aquest *script* és el següent:

Les instancies de la clase *Character* que estan a la llista creada en el scrip *SearchCharacter* que són *Pnj* o *PnjRange* seguiran tota la estona una ruta, en el cas de no veure cap enemic. El mètode que s'utilitza per fer aquestes rutes es el mètode *monitoring*.

En aquest mètode si s'ha arribat a un punt de la llista pròpia de cada *Pnj* que especifica els punts de la ruta, s'actualitza la llista amb el següent punt. En cas contrari es crida al mètode *move* del *CharacterController* del enemic.

Els comportaments dels enemics de curt i llarg rang difereixen entre ells. Primer parlaré del comportament dels enemics de curt rang.

Per detectar als personatges es crida al mètode `overlapSphere`, aquest mètode detecta els *colliders* a una cert radi respecte una posició, així doncs els arguments utilitzats són la posició del *GameObject* associat a la instància de la classe *PnjRange* i la distància que caracteritza al atribut *sight* multiplicat per 10 d'aquesta instància. En el cas de que es detecti el *collider* d'un enemic s'assignarà el personatge com a enemic a un atribut de la classe *Pnj* que és una llista de enemics, i el enemic es mourà cap el primer personatge principal que hagi detectat, en cas de que no estigui lluitant contra un altre enemic. En cas de que el personatge principal mori, es borrarà de la llista de personatges a seguir, i es destruirà el *GameObject*, el sistema de partícules de sang corresponent a aquest personatge i es borrarà de la llista de personatges.

En cas de que els enemics no hagin detectat als personatges principals, pot passar que si els enemics han sigut seleccionats amb el ratolí ja siguin a la llista de enemics dels *Pnj*. En aquest cas es mira si el personatge principal que és a la llista és un arquer o un mag. En cas de ser un arquer i la distància entre el enemic i el personatge sigui inferior al atribut *DistanceFight* propi de cada *character*, el enemic es mourà cap a ell. I en cas de que el personatge sigui un mag, mirarem si el mag a tirat un encanteri de llarga distància, en aquest cas que així sigui el enemic anirà fins al personatge principal.

En quant el enemic de llarg rang farà el mateix que el enemic de distància curta, però amb l'única diferència que s'aturarà per atacar a una distància més llarga.

Per tal de que el *GameObject* *SeguimientoPnj* segueixi al seu corresponent enemic s'utilitzarà el *script* *FollowPnj*. Aquest *GameObject* es crea quan s'instància un enemic, i per tal de seguir-lo es fa el següent, es busca a la llista de caràcters els enemics i mitjançant la variable posició del component *transform* de *GameObject* s'assigna la posició del objecte *SeguimientoPnj* a la posició del enemic.

Script *LifeFollowPnj* i *DeathFollowPnj*

Aquests dos *scripts* faran que els *GameObjects* *lifeTexture* i *deathTexture*, la barra de vida i de mort respectivament, segueixin al seu respectiu enemic. Per fer això es calcula la posició del enemic mitjançant el mètode `getTransformPj` utilitzat en aquest *script* i s'assigna la posició de la barra de vida a la posició del enemic però en coordenades de càmera perquè les barres de vida són *GuiTextures* i és més fàcil treballar amb aquests tipus de *GameObjects* utilitzant aquest tipus de coordenades.

També cal dir que al *script* *lifeTexture* es calcula la mida de la barra de vida per saber el dany del enemic.

Script FollowPnj

En aquest *script* es calcula la posició del *GameObject* SeguimientoPnj, s'utilitza la component transform del *GameObject* SeguimientoPnj, a cada moment la posició d'aquest objecte serà la mateixa que la el enemic.

5.4.5. Encantaments y Habilitats

- Habilitats

Com s'ha dit en punts anteriors, els guerrers i arquers disposen de habilitats especials. Aquestes habilitats son gestionades per el *script* EquipManagementWarrior i EquipManagementArcher respectivament i activades pel Hud.

El mètode *update* del *script* EquipManagementWarrior mirarà si la habilitat del guerrer ha sigut activada, habilitat que serà activada des del *script* Hud quan es polsi un determinat *GuiButton* i augmentarà la característica de força al guerrer. Mentrestant aquest mateix *script* cridarà a cada segon a la funció ImFurius mitjançant *invokeRepeating*. A la funció imFurius es comprovava que una variable booleana sigui certa, aquesta variable es posarà a cert al *update*, quan es comprovi que la habilitat es activada. També es comprovarà a cada segon que la duració de la habilitat hagi acabat, si es el cas, es restarà al guerrer la força que li proporcionava la habilitat, apart de destruir el sistema de partícules mitjançant la funció *cancelEffect* que s'ha utilitzat per aquesta habilitat.

Des de el Hud es cridarà al mètode *effectSpell* de la classe BarbarianFury per crear el sistema de partícules associat a aquesta habilitat. A la imatge 76 es pot veure l'efecte de la habilitat BarbarianFury.



Imatge 76: Efecte de la habilitat Barbarian Fury

El *script* EquipManagementArcher funciona de la mateixa forma, però en comptes de reduir la força del personatge principal al que la habilitat esta associat, doncs redueix la destresa. La destresa també serà augmentada al pulsar el *GuiButton* del Hud. A la imatge 77 es pot veure l'efecte de la habilitat FastArrow.



Imatge 77: Efecte de la habilitat Fast Arrow

- Encantaments

Els encantaments es gestionen a partir del *script* SpellManagement. En aquest *script* es gestionen els dos encantaments, ArmorSpell i MagicBulletSpell.

En quan l'encanteri ArmorSpell en aquest *script* mitjançant invokeRepeating es va cridant a cada segon el mètode imArmor. Aquest mètode s'assigna a una variable els SpellList del mag mitjançant el mètode getSpellListOfWizard; i a un *arrayList* tots els encantaments que han sigut sol·licitats per l'usuari als diferents personatges mitjançant el mètode getSpellinAction propi de la classe SpellList que te el Wizard. S'iteren tots els encantaments d'aquest *arrayList* i es crida al mètode managementArmorSpell, passant per argument els corresponents encanteris i la spellList . En el mètode managementArmorSpell, es mira si els encantaments que hi ha a la llista han sigut activats pel guiButton del Hud, si es el cas es va reduint la duració del encantament, i quan s'acaba el temps del encantament es cancel·la l'activació d'aquests, es destrueix el sistema de partícules associat, es torna a reduir la característica de la destresa a la destresa inicial i es treu el encantament de la llista de encantaments en acció.

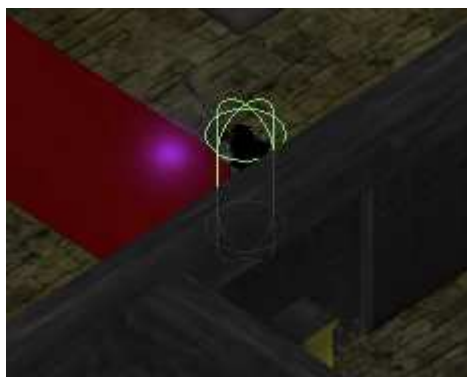
Quan s'activa el encantament des de el Hud, es crida a la funció effectSpell de la classe ArmorSpell, aquest mètode puja la característica de destresa al personatge seleccionat i crea el sistema de partícules associat a aquests encanteri.

En quant l'encanteri MagicBulletSpell el que es fa en el *update* del *script* SpellManagement es assignar a una variable els SpellList del mag mitjançant el mètode getSpellListOfWizard; i a un

arrayList tots els encantaments propis de la llista *list* de la classe *SpellList*, es miren els encanteris d'aquesta llista i per els que tinguin l'identificador 1, que es el que identifica un *MagicBulletSpell*, es crida a la funció *managementBulletMagicSpell* passant per paràmetre l'encanteri i el *spellList*, aquesta funció el que fa es posar a fals el atribut *activationSpell* per poder fer mes encanteris d'aquests tipus.

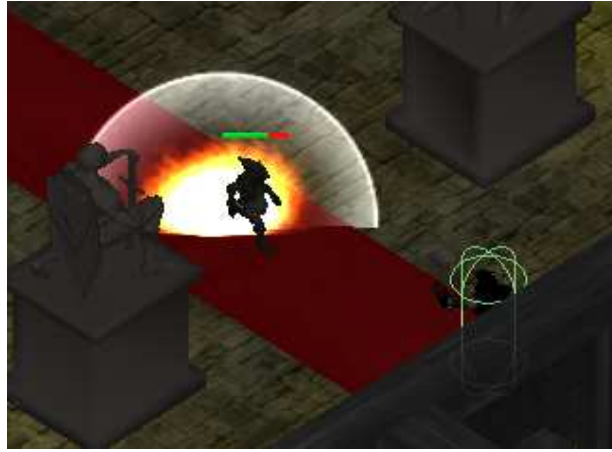
El Hud activarà el mètode *efectSpell* de *MagicBulletSpell* polsant el *GuiButton* corresponent. Aquest mètode instanciarà el sistema de partícules, a la posició del mag, associat al encanteri i calcularà els danys que causen al enemic, per calcular aquests danys es prendrà un valor aleatori entre un i sis, multiplicat pel nivell mes un del mag i s'assignarà aquest valor al atribut *damage* de l'encanteri, així com el enemic seleccionat.

En quan al *script* *ControllerMagicBulletSpell* el que fa es agafar el objecte *spellList*, atribut del mag, iterar per la llista de encanteris d'aquests objecte. Agafar l'encanteri corresponen al *MagicBulletSpell*, aquests encantaments ja han sigut activats pel Hud, i el mouen utilitzant la funció *MovePosition* del *rigibody* del *GameObject* associat al encanteri. A la imatge 78 es pot veure l'efecte del encanteri.



Imatge 78: Efecte del encantament Magic Bullet

Gracies al *Collider* que te aquest sistema de partícules es pot utilitzar la funcio *OnCollisionEnter* pròpia dels *colliders*, aquesta funció permet saber si un *collider* xoca contra un altre. Si topa contra un *collider* anomenat *SeguimientoPnj* es busca el *MagicBulletSpell* de la llista de encanteris que correspon amb el que topa contra el enemic i s'agafa el valor del atribut dany i el atribut *Pnj* calculats en el mètode *efectSpell* de la classe *MagicBulletSpell* i es resta aquest dany de la vida del enemic. Acte seguit es destrueix el sistema de partícules del encantament i s'instancia el de una explosió. A la imatge 79 es pot veure el sistema de partícules de l'explosió i com el encanteri treu vida al enemic.



Imatge 79: Efecte de la explosió al impactar l'encateri

5.4.6. El Hud

Com ja he explicat en el punt 3.2.8, el Hud es la informació que es mostra en la pantalla durant la partida. Per fer el Hud he creat un *GameObject* buit on seran els *scripts* principals que tenen que veure amb aquesta informació.

El Hud principal del joc s'ha fet a partir de *Guis*. Unity per crear les *Guis* fa us d'un mètode especial anomenat *OnGui()*. Aquesta funció es crida a cada frame i es cridada cada cop que es vulgui renderitzar o gestionar events dels diferents components *Gui*.

Així doncs el que s'ha fet en aquest script es crear tots elements *GUI*, aquests elements son:

- *GuiBox*: Utilitzat com a contenidor dels diferents *GuiComponents*
- *GuiLabel*: Mostra el número de encantaments o habilitats disponibles.
- *GuiButton*: Executa els diferents encantaments i habilitats, també permet la selecció dels personatges.
- *GuiTexture*: Mostra la vida de enemics i personatges
- *GuiText*: Mostra missatges de lluita i missatges
- *GuiLayout*: Utilitzat per que la text Area estigui centrat en pantalla.

Script Hud

El *script* funciona de la següent manera:

Es crearan dos Gui.Box a les dues bandes de la pantalla, amb una amplada determinada. Al Gui Box de la dreta es crearan els GuiButtons corresponents als *portraits* dels personatges, les barres de vida i el boto de descans.

Al GuiBox de la esquerra es crearan els botons de encantament o habilitat corresponen al personatge seleccionat.

En el mètode onGui s'iteren els diferents personatges principals del joc mitjançant el arrayList que conte a tots els personatges i es mira que aquests personatges siguin els personatges principals.

Per a cada personatge que sigui guerrer, mag o arquer es carregarà el seu *portrait*, en cas de que sigui mort es carregarà el *portrait* però en escala de grisós. Aquests *portraits* son imatges que estan ubicats a la carpeta Resources i es carregaran com Textures2D. Aquestes imatges es carregaran dins del component GuiButton i s'ubicaran a una determinada posició depenent del personatge.

Es carregaran las imatges de vida i mort i s'ubicaran sota del *portrait* de cada personatge. La imatge corresponent a la vida s'anirà reduint quant el personatge perdi vida, per fer això es va calculant la vida del personatge principal de la següent forma. Es mira lla vida del personatge, mitjançant el mètode getPg i es divideix entre el màxim punts de vida que te el personatge, després d'aquest càlcul es multiplica per l'amplada que te la aquesta imatge i es dibuixa amb la seva nova amplada.

En quan al boto del *portrait* farà que si l'usuari el polsi, el personatge seleccionat canviarà, també tornarà a aparèixer la aureola corresponent al personatge seleccionat.

Si polses el boto de descans, els personatges descansaran, aquest boto s'ubicarà a sota dels *portraits*, el seu funcionament es el següent:

El que es fa es mirar si es troben tots els *GameObject* que pertanyin al personatges principals, si hi son es mira la distancia entre cada un d'ells, per tal de que pugin descansar s'han de trobar a una distancia concreta. Si algun dels *GameObject* no hi es, es adir es mort, es mira la distancia entre aquests *GameObject* i si solsament queda un personatge amb vida podrà descansar .

En tot cas si es troben a la distancia en que poden descansar o be solsament queda un personatge amb vida es crida al mètode sleeping. Aquest mètode el que fa es mirar si els personatges que son a la llista de personatges del joc, si no son morts o be no estan en una lluita, podran descansar.

Abans de descansar es mira si el personatge es un guerrer, un arquer o un mag.

Si es un guerrer o un arquer es comprovarà que la habilitat associada a aquest haguí finalitzat, en cas de que així sigui, descansaran.

En cas de que sigui un mag es comprova si la llista de encanteris associada al mag es buida(aquesta llista es un atribut propi del mag), en concret la llista de encanteris en acció, això vol dir que els encanteris que pugen certes característiques durant un temps han finalitzat. Si es dona el cas de que hagin finalitzat, els personatges descansaran.

S'ha creat un *timer* per mirar quan poden descansar, per crear un *timer*, el que es fa es utilitzar el mètode de Unity `InvokeRepeating`. Aquest mètode rep com a paràmetres el nom de la funció que es cridarà a cada unitat de temps, per a mes informació sobre el mètode consultar la api: [InvokeRepeating](#).

El mètode timer que es el nom que invocarà la funció `InvokeRepeating`, el que fa es mirar si s'ha polsat el boto mitjançant una variable booleana i mirar si una variable entera es superior a un enter, en cas de ser cert es podrà descansar, si no, no es podrà i a mes es penalitzarà, ja que al mètode `sleep` farà que s'inicialitzi de nou el comptador.

Pel que fa als botons de las habilitats o encanteris el procediment es el mateix que la carrega del *portrait* i de la vida però amb la diferencia de que es mira quin personatge ha estat seleccionat, per a que apareguin la habilitat o encanteris del personatge escollit al `GuiBox` corresponents.

Per carregar la imatge de l'habilitat del guerrer o arquer al `GuiButton` corresponen es necessari carregar la imatge associada a la habilitat, això es fa mitjançant el mètode `Load`, el nom de la imatge que es carregarà es una atribut de la classe `Equipment`, aquest objecte `Equipment` al mateix temps es atribut de la classe `Warrior` o `Archer`,depenent de quin personatge sigui seleccionat. Per carregar el número de habilitats que disposa es fa el mateix, però mirant el atribut corresponent al numero de encantaments. A la imatge es pot veure la línia de codi que carrega una imatge `Texture2D`.

```
textureBf = Resources.Load(bf.getEquipment().getPortrait()) as Texture2D;
```

Imatge 80:Codi que carrega una imatge

Pel que fa el mag es fa el mateix però es crean dos `GuiButtons`, ja que tenim dos encanteris.

Per activar les habilitats en cas de ser arquer o guerrer, s'ha de mirar si es pren el boto de habilitat corresponen ,si el numero de habilitats no es zero i si el menú de pausa no esta activat podem activar la habilitat ,així com activar l'efecte de la habilitat, reduir el numero de habilitats associada a la habilitat del personatge corresponent i augmentar la característica d'aquest personatge, això s'aconsegueix accedint al objecte de la habilitat que pertany a la instancia de la classe del personatge seleccionat i accedint als mètodes corresponents.

En el cas de seleccionar un encanteri del mag, si el encanteri es el que correspon a `ArmorSpell` el que es fa es aturar el moviment del mag mitjançant la crida del mètode corresponent, guardar el personatge que ha sigut seleccionat des de la classe `pjSelection`, es crea una instancia del

encantament `ArmorSpell`, passant per argument aquest personatge seleccionat. Es mira que el numero de encants sigui major que zero i s'afegeix a la llista de encants en acció del mag ,propi de la calsse `SpellList` del mag.

Acte seguit s'activen els encants que no estiguin activats, es crida al mètode `effectSpell` de la classe `ArmorSpell` passant com arguments el personatge seleccionat augmentant el numero del encantament `ArmorSpell` i s'activa l'animació d'aquest encanteri.

Si es selecciona el boto corresponent al encanteri anomenat `MagicBulletSpell`, el que es fa es assignar el enemic al que li llançarem el encanteri, mitjançant la variable estàtica `assignPnjForSpell` de la classe `PJSelection`.

Acte seguit es buscarà el encanteri en la llista de encants de la classe `SpellList` pròpia del mag i l'encantament no esta activat i el numero de encants disponibles es mes gran que zero doncs activem l'encanteri i activem l'animació corresponent a aquest encanteri.

Amb els *scripts* `FollowHabilities` y `FollowArmorSpell` fem que els sistemes de partícules segueixin als personatges mitjançant les variables de la classe *transform* pròpies de Unity., cercant abans els *GameObjects* que volem que segueixin.

5.4.7. Combats

En aquest apartat veurem que fan els diferents *scripts* utilitzats per gestionar combats

Script SearchCharacter

Al *script* `SearchCharacter` es buscaran els enemics dels personatges i dels enemics, es a dir els nemesis dels corresponents personatges i s'afegiran a una llista de enemics,que serà atribut dels diferents personatges.

Script UpdateDistances

En aquest *script* el que es fa es mitjançant la llista de tots els personatges, es busca la distancia entre aquests i els enemics, utilitzant la llista que funciona de atribut de cada personatge. Aquestes distancies s'actualitzaran en el atribut diccionari de cada personatge, aquest diccionari te com a clau, els enemics d'un personatge i com a valor la distancia.

Script GoFight

En aquest *script* es creen las diferents lluites i s'afegeixen nous lluitadors a les lluites que encara existeixen, també es finalitzen aquestes.

Per gestionar totes les lluites, es crea una llista de tipus Fight, aquí es guardaran totes les lluites que es facin,

Bàsicament el que es fa es mirar si els enemics dels personatges principals son a la distancia designada per lluitar segons el atribut distandeFight i si els personatges principals han seleccionat un enemic(això es fa en el *script* GoController, si es selecciona un enemic) o be que els enemics son a la distancia designada per lluitar. Si passa això es crearà una nova lluita en el cas de que cap personatge ni enemic siguin en una lluita, per mirar si son en una lluita utilitzem el atribut de la classe Character goFight, li donem al personatge que intervé en la lluita un identificador de lluita, aquest identificador va augmentat sempre que es creí una lluita.

En cas de que el personatge principal no sigui en una lluita i el enemic al que es vol atacar si que i sigui, o al cas oposat ,es buscarà el identificador de la lluita que esta succeint, es posarà el atribut del personatge goFight a cert, se li assignarà el identificador de la lluita a la que acaba d'entrar i s'afegirà a la llista de lluitadors de la lluita a la que s'acaba de ficar.

Per controlar ja els estats de les lluites es mira si els personatges principals, estan lluitant a traves del atribut goFight i es mira si existeix un identificador de lluita que no sigui -1, si es el cas i es van controlant aquests estats, el estat un correspon a la tirada de iniciatives, l'estat dos es correspon al desenvolupament de la lluita si no ha mort cap personatge i el estat tres es correspon a mirar si un personatge ha mort o no. Si es troba un identificador de lluita -1 vol dir que la lluita ha finalitzat, es a dir que els enemics han mort, es treura al personatge de la lluita i es posarà l'atribut de goFight a fals.

Els cassos que tenen a veure a mirar si un personatge esta lluitant i el seu enemic no o al revés es cridarà al mètode findFightBegin, en aquest mètode es busca l'identificador de la lluita a la que un personatge vol entrar, el que es fa es iterar els lluitadors de la lluita corresponent i retornar l'identificador d'aquesta lluita.

El cas que te a veure en si un personatge principal esta lluitant es busca l'identificador de la lluita on esta intervenint aquest personatge principal.

Script ParticleBloodFollow

En aquest *script*, que utilitzat com a component del *prefab* particleBloodFollow, el que s'ha fet es que els *gameObjects* instanciats quan s'executa el joc segueixin als personatges que posseeixen dits *gameObjects*.

Classe Fight

En aquesta classe es gestiona una lluita. En el constructor posem l'estat de la lluita a un, que correspon a la tirada de iniciatives. Afegim a una llista de lluitadors els dos personatges que es passen per paràmetres al constructor i a una variable el identificador de lluita. Als personatges que son enemics els afegim a una llista de enemics i als que son personatges principals a una llista de personatges.

Des de el *script* GoFight es mira l'estat de la lluita i si aquest es un s'iniciarà els torns de iniciatives.

En el torn de iniciatives es crea una cua, i els personatges que formen part de la llista de lluitadors faran una tirada, aquesta tirada es correspon al mètode initiativeBigThrow de la classe Characters i es guardarà en un vector. En el mètode initiativeBigThrow es calcula la iniciativa del personatge a partir de la suma del modificador de destresa, 20 i el nivell que te el personatge dividit per dos. Agafem la tirada mes gran de totes las fetes d'aquets vector.

Després el que es fa es crear una instancia d'una celda passant per argument cada un dels personatges lluitadors i la diferencia de la tirada mes alta i una nova tirada, aquesta nova tirada es igual que la anterior però en comptes de sumar 20, es sumarà un número aleatori entre 1 i 20, això es fa perquè els resultats baixos seran els que tinguin millor prioritat, ja que si un personatge te una prioritat pròxima a zero podrà tenir el torn aviat. Després el que fem es assignar la llista de enemics com atribut als personatges i al revés perquè tant personatges com enemics tinguin la seva llista de enemics respectiva. Aquest torn de iniciatives solament es farà un cop en tota la lluita a no ser que un altre personatge s'afegeixi a la lluita.

La gestió del torn d'atac correspon al estat dos. Per gestionar els torns d'atac es fa el següent:

Es va reduint el comptador de cada celda de la cua de prioritats mitjançant el mètode reduceCount de la classe PriorityQueue i quan el comptador de qualsevol celda arriba a 0 s'agafa el personatge(principal o enemic) corresponent a la del comptador que ha arribat a zero mitjançant el mètode strain i firstQueue de la classe QueuePriority, i si aquest personatge te mes de zero PG i es troba a una distancia que pugui atacar al seu contrincant es calcula la tirada de atac mitjançant el mètode attackThrow de la classe Character. Aquesta tirada es la suma d'un numero aleatori entre 1 i 20, el seu atac base i el seu modificador de força. Després es calcula la tirada de defensa corresponen al mètode caThrow. Aquesta tirada la farà el primer enemic de la llista de enemics del personatge que te el torn, ja que s'ha decidit atacar sempre al primer enemic que et vol atacar. La tirada de defensa es la suma de 10, el atribut armorCa de la classe Character i el modificador de destresa. Si la puntuació de la tirada d'atac es mes gran que la puntuació de la tirada de defensa, el personatge atacant farà una tirada de dany contra el defensor, la tirada de dany es la suma de un numero aleatori entre 1 i l'atribut atac base de la classe Character i el modificador de força. Amb el

resultat d'aquesta tirada es treura PG al personatge que s'estava defenent, i a part es reproduirà el efecte del sistema de partícules de sang. A la imatge 81 podem veure una lluita entre el guerrer i un enemic.



Imatge 81: lluita entre dos personatges

Després d'aquest intercanvi de tirades es mirarà si el personatge que s'estava defenent a mort, per fer això es fa el mateix que en el estat dos, s'agafa el primer enemic del personatge atacant, llavors es mira els Pg d'aquest i si es menor o igual a zero es borra de la llista d'enemics de la classe Character, la celda que pertanyi a aquest personatge de la cua de prioritats i de la llista de lluitadors. També es carregarà el prefab que representa el cadàver de un dels personatges, així com un *decal* de sang i es posarà a la ultima posició on aquest personatge i es posarà a cert el atribut *death* corresponent a la classe Character. Per finalitzar s'esborrarà el personatge corresponen als diccionaris i també s'esborrarà el personatge de la llista d'enemics.

En el cas de que el personatge defensor no hagi mort s'ordenarà la cua de prioritats posant la prioritat inicial a la celda en el que el atribut era zero.

5.4.8. La càmera

El *script* anomenat CamFollowPj s'encarregarà de seguir els personatges des de una altura i amb un zoom determinat. Aquest *script* estarà ubicat al *GameObject* MainCamera.

Per tal de seguir al personatge seleccionat es mira quin personatge ha estat seleccionat mitjançant la variable de la classe PjSelection. Al inici s'utilitza una distancia ,una altura i una rotació predefinida, per tal de moure la càmera m'entres que el personatge es mou el que es fa es calcular l'angle corresponen al eix y del *GameObject* main.camera. Per a calcular aquest angle s'ha de calcular una interpolació entre el valor de la component y de la posició de la càmera i dos variables amb valors predeterminats. Un cop calculat aquest valor, s'ha de calcular una rotació sobre l'eix y de la càmera. Llavors per que la càmera es mantingui amb l'angle que es vol es calcula de la següent forma:

```
tmp= new Vector3(target.position.x,target.position.y,target.position.z);  
tmp -= currentRotation * Vector3.one* distance;  
tmp.y = currentHeight;
```

Imatge 82:Codi corresponent a la posicio i rotacio de la camera en el joc

I per tal de que et segueixi, s'ha d'assignar la posició de la càmera al valor calculat anteriorment.

Per a que la càmera roti segons la posició del personatge s'utilitza :

```
transform.LookAt(target);
```

Imatge 83:Mètode utilitzat per seguir la posició del personatge

5.5. Menús

5.5.1. Menú Principal

El menú principal es va crear amb l'editor de Unity, es un menú 2D. Es divideix en dos escenes, a la primera escena, anomenada menú, es la que es mostrarà res mes executar el joc, apareix el menú amb les opcions de jugar, opcions de joc i sortir del joc. El primer que es va fer es el fons del menú, el fons es un pla escalat al que se li va afegir la imatge que es pot veure a la imatge 84.



Imatge 84:Background utilitzat al menú de joc

Després es van crear les imatges corresponents al títol del joc i a les diferents opcions del menú mitjançant Gimp. Aquestes imatges van ser afegides a quatre plans mes, que varen ser escalats, a

aquests plans tenen *collider*, per poder gestionar les accions dels usuaris quan els polsin. A la imatge 85 podem observar l'estat final del menú.



Imatge 85: Estat final del menú principal

Al *script* LoadGame es creen les funcionalitats del boto jugar, amb el mètode OnMouseExit canviem la imatge del pla, aquesta imatge serà la m'esma però la vora canviarà de color. En quant al mètode OnMouseEnter, fem el mateix però carregant la imatge del boto inicial, el mètode OnMouseDown carregarà l'escena del joc. A la imatge 86 es pot veure com carregar una escena.

```
Application.LoadLevel("tfg scene")
```

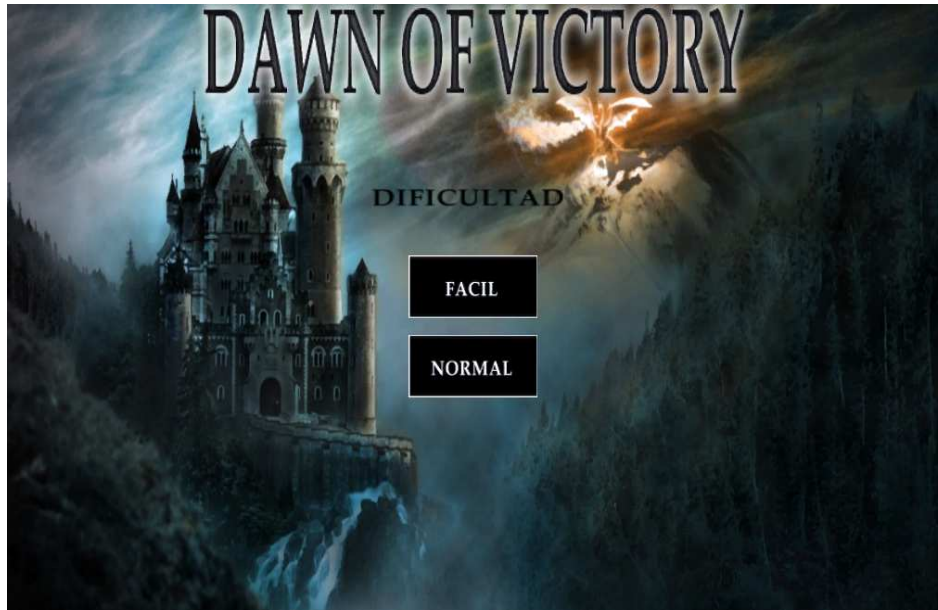
Imatge 86: Codi corresponent a la carrega d'una escena

En quant als botons de opcions i sortir dels *scripts* OptionsGame i ExitGame, els mètodes OnMouseExit i OnMouseEnter faran el mateix que el script LoadGame. La única diferencia es en el mètode OnMouseDown, al *script* OptionsGame en aquest mètode es carregarà la escena del menú de opcions, en quant al *script* ExitGame es cridarà al mètode per sortir del joc. A la imatge 87 es pot veure la línia de codi per sortir del joc.

```
Application.Quit();
```

Imatge 87: Codi corresponent a la sortida del joc

A la escena anomenada, MenuOptions, es troba una única opció de joc, que es la dificultat, que pot ser fàcil i normal. Aquest menú esta fet de la mateixa forma que el principal, amb diversos plans, on cadascun tenen diferents imatges. A la imatge 88 podem veure el menú de les opcions de joc.



Imatge 88: Codi corresponent al estat final del menú de opcions

Per escollir les opcions de joc s'ha utilitzat el *script* ChangeDifficulty per inicialitzar la variable de dificultat, aquesta dificultat serà canviada en els *scripts* EasyMode i NormalMode. En aquests *scripts* es canvia la variable de la dificultat a través dels mètodes OnMouseDown, on també es carregarà l'escena del menú principal. A partir de l'assignació de la variable de la dificultat en el *script* searchCharacter es carregaran un determinat número d'enemics o uns altres.

5.5.2. Menú de pausa

S'ha creat un menú de pausa del joc, aquest menú serà accessible quan el jugador sigui a una partida, es podrà accedir mitjançant la tecla "P" del teclat, això farà que el joc pari posant la variable TimeScale de Time a 1 i donarà al usuari les opcions de sortir del joc i de continuar jugant. S'ha creat mitjançant un *script* anomenat MenuPauseAparition, en aquest *script* es creen els botons de continuar i sortir mitjançant dos GuiButons que es definiran al mètode OnGui. Quan es polsi al boto de continuar es reiniciarà el temps posant la variable TimeScale a 0. Si l'usuari polsa el boto sortir es tancarà l'aplicació mitjançant el mètode Application.quit. A la imatge 89 podem veure el menú de pausa.



Imatge 89: Codi corresponent al menú de pausa del joc

5.6. Fi del joc y morts

En el script `ControlWarriorDead` es controla els esdeveniments que succeiran quan morin els personatges. Per controlar si s'ha perdut o guanyat en el joc utilitzarem una variable booleana, aquesta variable comprovarà si els `GameObjects` que representen els personatges existeixen o no. En cas de no existir aquesta variable serà certa i per tant es carregarà la textura corresponent a la imatge de fi del joc. En aquest cas s'esborraran de la llista les instàncies de la classe `Character` de la llista de personatges i el component de la llista de personatges neutral per tal de carregar l'escena del menú principal. Per carregar l'escena de menú un cop es mostri la imatge del fi del joc s'ha utilitzat el mètode `GetMouseButtonDown`. La carrega de la imatge, així com la gestió de les llistes en cas de que morin els personatges principals està ubicada a la funció `OnGui`.

En el `update` d'aquest script es controla la destrucció dels sistemes de partícules dels encantaments i habilitats especials dels personatges un cop han mort i aquests encara eren activats. Pel que fa a les habilitats del arquero i del guerrer es comprova si el `GameObject` corresponent als personatges principals existeixen, si no existeixen es destrueix el sistema de partícules associat a la habilitat.

En el cas del encanteri del `ArmorSpell`, com es un encanteri i es propi del mag i es la classe `Wizard` qui te la llista de encanteris s'ha decidit que quan mori el arquero o el guerrer els encanteris llançats sobre ells desapareixeran. Per això s'ha de mirar si aquets personatges son morts i destruir el sistema de partícules creat. Per fer això en el cas de que sigui llençat un encanteri a un arquero o a un guerrer es mira si aquests son morts i que el mag no sigui mort, si es el cas el que es fa es destruir el efecte de partícules que segueixen als personatges morts.

Des de aquest script també s'eliminaran de la llista de personatges que s'han seleccionat amb el quadre de selecció una cop morts.

6. Proves i resultats

En aquesta part s'analitzaran els resultats final dels diferents elements creats en el videojoc, així com les seves possibles millores.

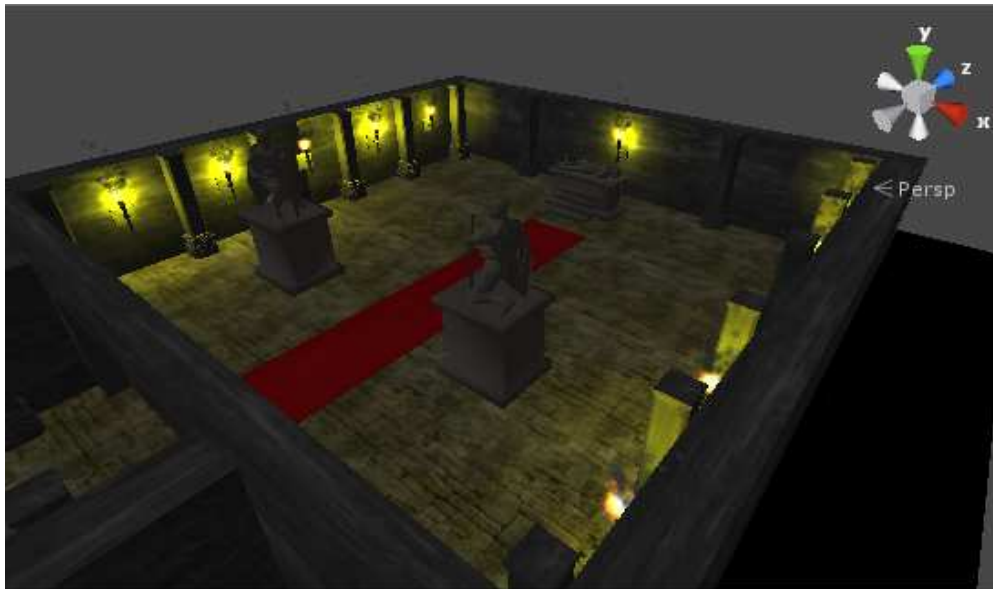
6.1. L'escenari

L'escenari inicial va ser reduït respecte als esbossos inicials. Al final els passadissos laterals de la cripta varen ser eliminats i solsament existeix un passadís que va de la entrada al salo principal. A la imatge 90 es pot veure la distribució del escenari final. En quadre color vermell es pot veure la entrada, en el quadre taronja el passadís i en el quadre groc el salo principal.



Imatge 90: Distribució final del escenari

A la imatge 91 es pot veure el salo, amb els diferents elements incorporats creats amb Blender .



Imatge 91: Visualització del estat final del salo

En quant al passadís es va tindre que modificar la mida de la porta, per tal de que els personatges poguessin passar be tots a l'hora si es seleccionaven a tots per anar a una posició del escenari sense provocar cap embut a l'hora d'entrar al salo. Els panells que activaven la porta, en un principi eren cada un a un costat, es van tindre que desplaçar a una de les parets del passadís com a conseqüència de la modificació de la mida de la porta. A la imatge 92 es pot veure l'estat final del passadís.



Imatge 92: Visualització final del passadís

A la imatge 93 es pot veure la disposició final dels panells.



Imatge 93: Visualització dels panells en el passadís

En quan a la entrada la podem veure a la imatge 94



Imatge 94: Visualització final de la entrada

En quan a la il·luminació del escenari crec que ha estat un gran encert. Disminuint la llum direccional i creant els punts de llum ha estat un total encert per a la ambientació del escenari, a mes queda molt realista el fet d'apropar el personatge a una de les torxes i que la llum quedi reflectida en el cos d'aquest. A la imatge 95 podem veure aquest efecte.



Imatge 95: Visualització del reflex de la llum de la torxa al model

6.2. Personatges

6.2.1. Personatges principals

6.2.1.1. Els personatges al joc

La importació dels models dels personatge de Blender a Unity no ha donat cap problema. A la imatge 96 podem veure el estat finals dels 3 personatges principals al joc.



Imatge 96: Estat final dels personatges principals al escenari

6.2.1.2. Accions dels personatges al joc

SELECCIO I MOVIMENT

Es la acció de seleccionar els diferents personatges del joc i que es moguin per l'escenari. Al vídeo 1 es pot veure com es mouen utilitzant la selecció individual, múltiple i per selecció de quadre.



Vídeo 1:Vídeo que mostra els moviments dels personatges

Es pot veure que tot funciona correctament, l'única pega es el fet de quan dos o mes personatges es seleccionen per anar a una mateixa posició del escenari, al topar entre ells, el primer en arribar a la

posició seleccionada para el seu moviment, els demés no paren ja que encara no han arribat a la posició, si desplaçem el primer personatge que arriba, els altres personatges arribaran ja a la posició seleccionada al inici. En tot cas el fet de que això sigui d'aquesta forma no resta jugabilitat.

HABILITATS ESPECIALS

Habilitat Fúria Bàrbara: Es la habilitat especial del guerrer, al vídeo 2 es pot veure que seleccionant la habilitat en el Hud, de forma automàtica el personatge executa la corresponent animació i els crea el sistema de partícules, al acabar el temps de la habilitat, el sistema de partícules es destrueix.



Vídeo 2:Vídeo que mostra la habilitat Barbarian Fury del guerrer

Habilitat fletxa rapida: Igual que la habilitat especial del guerrer, al vídeo 3 es pot veure que seleccionant la habilitat en el Hud, de forma automàtica l'arquer executa la corresponent animació i els crea el sistema de partícules, al acabar el temps de la habilitat, el sistema de partícules es destrueix.



Vídeo 3:Vídeo que mostra la habilitat Fast Arrow del arquero

ENCANTAMENTS

Encantament Armadura Màgica: Aquest es un dels encantaments que pot executar el mag, en el vídeo 4 es pot veure com es selecciona l'encantament i acte seguit es selecciona al personatge que rebrà el encantament.



Vídeo 4:Vídeo que mostra l'encanteri Armor

Encantament projectil Màgic: Aquest es un altre dels encantaments que pot executar el mag, en el vídeo 5 es pot veure com es selecciona l'encantament i acte seguit es selecciona al personatge que rebrà el encantament, al primer cop l'encantament no ha funcionat perquè el enemic esta fora del rang d'aquest encantament. però en el segon intent el enemic es a una distancia on es pot fer l'encantament de forma correcta



Vídeo 5:Vídeo que mostra l'encanteri MagicBullet

DESCANSAR

Acció que s'executa quan es selecciona el boto de descans. En el vídeo 6 es pot veure aquesta acció, es pot contemplar com el guerrer te poca vida després d'una dura baralla, en el textarea es veu els torns de la lluita, quan polsem al boto de descans la vida del personatge ferit es restableix.



Vídeo 6:Vídeo que mostra el descans dels personatges

LLUITAR

Acció que poden executar els personatges contra un o diversos enemics. En el vídeo 7 es pot veure una lluita entre el guerrer i el mag i un enemic de curta distancia, parem el temps en un moment determinat, per pensar l'estratègia idònia i m'entres el guerrer esta lluitant, llencem l'encanteri magicBullet contra l'enemic. Al final hem guanyat aquesta lluita.



Vídeo 7:Vídeo que mostra una lluita

6.2.1.3. Interacció dels personatges amb l'escenari

Seguidament s'analitzaran els diferents resultats accions que fan els personatges amb els elements del escenari.

MISSATGE D'ALERTA PER OBRIR LA PORTA DEL SALO

Quan el personatge sigui a la vora de la porta del salo principal, si el personatge ha superat amb èxit una tirada de daus a amb èxit, es mostrarà el corresponent missatge al log, es a dir al textArea. Si un dels personatges ha superat amb èxit la tirada, els demes ja no faran la corresponent tirada. Al vídeo 8 es pot veure com apareix el missatge d'alerta en el cas de tenir èxit a la corresponent tirada. Aquest missatge dona una pista de com obrir les portes.



Vídeo 8:Vídeo que mostra el missatge d'alerta als personatges

OBRINT PORTES

Per obrir les portes els personatges hauran d'accionar els dos interruptors que hi ha a una de les parets. Aquests panells son dos pentagrames. Al final els interruptors s'han posat a una de les parets, en comptes de fer-ho a cada costat de la porta pel fet de que es va tenir que augmentar la mida d'aquesta. El motiu va ser que el furat de la porta era massa petit perquè passessin els personatges seleccionats a l'hora, fent aquesta modificació la jugabilitat no s'ha vist afectada. En el vídeo 9 es pot veure com s'obre la porta després de accionar els interruptors.



Vídeo 9:Vídeo que mostra com s'obren les portes del salo principal

6.2.2. Els enemics

Els enemics al joc

La importació dels models dels enemics de Blender a Unity no ha donat cap problema. A la imatge 97 podem veure el estat finals dels dos tipus d'enemics al joc. L'enemic de curt rang i l'enemic de rang llarg.



Imatge 97: Estat final dels enemics al escenari

6.2.2.1. Accions des enemics al joc

RUTAS

Els enemics per defecte fan una vigilància per l'escenari en la que van d'un punt a un altre del escenari. El moviment serà caminant, amb una velocitat normal .

ATACAR

L'acció d'atac consta de tres fases, la primera d'aquestes fases es la de córrer fins a un dels personatges, depenent del tipus d'enemic, aquest s'aproxima fins a una determinada distancia.

La segona fase consta del atac, el enemic efectuarà el seu atac en el corresponent torn. En aquets dos vídeos es poden veure les accions d'atac dels diferents enemics. En el vídeo 10 es veure la ruta i l'acció d'atac del enemic de curt rang i en el vídeo 11 es pot veure com s'efectua la ruta i l'atac del enemic de llarg rang.



Vídeo 10:Vídeo que mostra la ruta d'un personatge de curt rang



Vídeo 11:Vídeo que mostra la ruta d'un personatge de llarg rang

En el cas de haver guanyat el combat, el enemic seguirà la ruta especificada abans d'haver localitzat un dels enemics. Al vídeo 12 es pot veure aquest fet.



Vídeo 12:Vídeo que mostra la ruta d'un enemic després de guanyar un combat

6.2.3. El personatge Neutral

El personatge neutral, farà una ruta d'un punt del passadís a un altre, un cop el personatge neutral, hagi vist als personatges, aquest mostrarà un missatge pel log alertant als seus companys i fugirà direcció al salo principal, abans d'arribar a la porta del salo desapareixerà. En el vídeo 13 es pot veure la acció del personatge neutral.



Vídeo 13:Vídeo que mostra la acció del personatge neutral

6.3. El joc

6.3.1. Inici del joc

Un cop executat el joc, apareixerà el menú principal mostrant el títol del joc, "Dawn of victory", i les diferents opcions que disposa.

- Jugar: Podrem iniciar el joc.
- Opcions: Entrarem al menú de secció de dificultat.
- Salir: Sortirem del joc.

En cas d'entrar en el menú d'opcions, escollirem la dificultat i automàticament tornarem al menú principal.

Un cop inicialitzat el joc apareixerà l'escenari principal. Al vídeo 14 es mostra l'inici del joc.



Vídeo 14: Vídeo que mostra el inici del joc

6.3.2. Èxit al joc

En el cas que matem a tots els enemics del escenari, apareixerà de forma màgica la recompensa, una espasa. En el vídeo 15 es mostra la derrota dels enemics en el mode fàcil i l'aparició de la recompensa.



Vídeo 15:Vídeo que mostra una partida guanyada

6.3.3. Joc Fallit

En el cas de que tots els personatges principals siguin derrotats, apareixerà un missatge de fi del joc, i es tornarà a començar la partida des de el principi. Al vídeo 16 es pot veure la mort de l'últim personatge en peu i la finalització del joc.



Vídeo 16:Vídeo que mostra la partida fallida

A la carpeta vídeos que ve amb aquest document es poden veure eles diferents probes, sense tenir accés a Internet.

7. Conclusions i treball futur

La valoració del projecte ha estat positiva, però hi ha coses que s'han de destacar. Inicialment aquest projecte era bastant extens per el temps de que es disposava, degut això, a mesura que s'anava desenvolupant i aprenent a utilitzar les eines per a poder dur a terme la realització d'aquest, es va optar per reduir les funcionalitats d'aquest però conservant l'essència dels objectius al que es volia arribar.

Un dels aspectes positius ha sigut aprendre a utilitzar varies tecnologies com son Unity3D i Blender per el desenvolupament del videojoc, així com el adquirir els coneixements necessaris per fer aquest projecte.

S'ha de dir que el videojoc es podria millorar de varies maneres, però crec que es compleixen les característiques que des de un primer moment s'havien proposat.

Les possibles millores són la creació duna millor forma per a realitzar els *paths* dels personatges utilitzant *pathfinding*; la millora de les transicions entre animacions dels personatges; la incorporació d'inventari en els personatges; la inserció de punts d'experiència cada cop que els personatges realitzin una tasca, així el seu nivell i característiques evolucionarien i podrien aprendre noves habilitats i la personalització dels personatges un cop s'inicialitza la partida.

8. Referències Bibliogràfiques

[1] <http://www.blender.org/>

[2] <http://docs.unity3d.com/>

[3] <http://answers.unity3d.com/index.html>

[4] <http://docs.unity3d.com/ScriptReference/>

[5] <http://unityspain.com/>

[6] <http://desnovato.blogspot.com.es/2012/04/como-crear-una-escena-en-unity3d.html>

[7] <http://unity3d-es.blogspot.com.es/p/recursos.html>

[8] [http://msdn.microsoft.com/en-us/library/vstudio/System.Collections\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/vstudio/System.Collections(v=vs.110).aspx)

[9] <http://www.gamedev.es/>

9. Manual de usuari

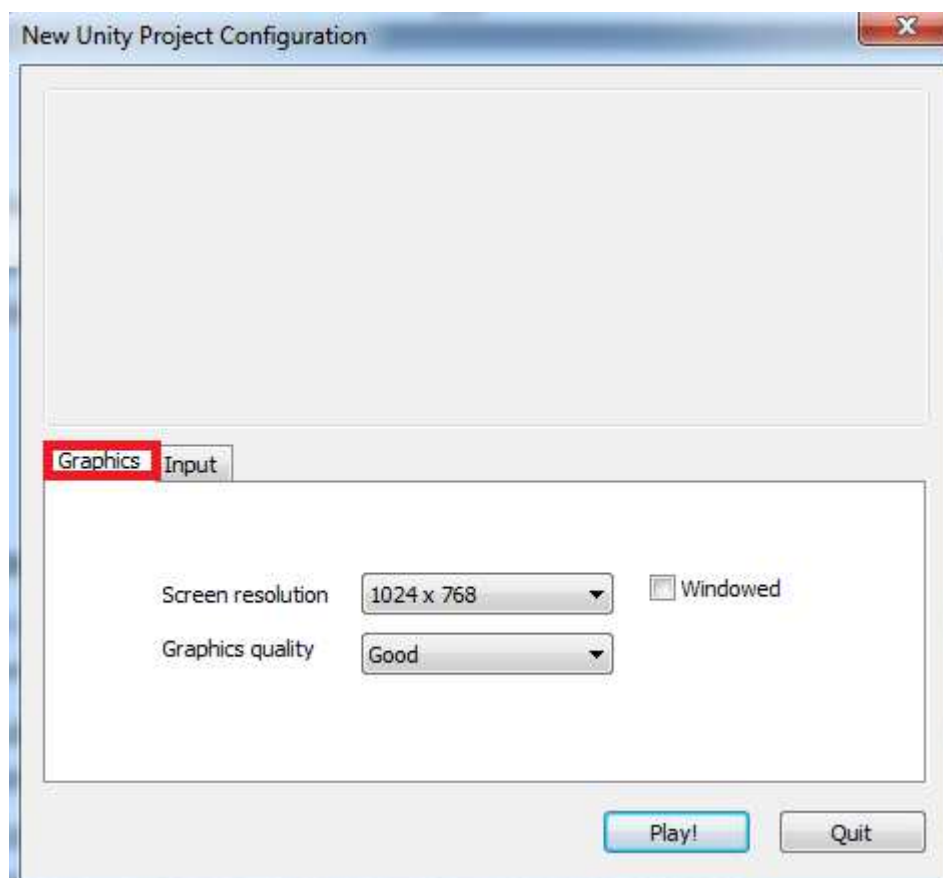
A continuació descripcions accions del usuari per tal de jugar al videojoc.

EXECUCIO DEL JOC

Un cop l'executable del joc sigui executat, apareixerà una finestra en la que es podrà modificar-les característiques de vídeo.

CONFIGURACIO DEL JOC

A la imatge 98 es pot veure la finestra un cop s'executa el joc. En el requadre vermell es podrà escollir la resolució en que es desitja executar el joc, es recomana que l'opció sigui de 1024x768. També es pot escollir la qualitat dels gràfics.



Imatge 98: Passos per la selecció d'una habilitat

CONTROLS

El control dels personatges en el joc es fa mitjançant la interacció amb el ratolí i el escenari de joc, així com amb el Hud. Les accions que pot fer l'usuari amb els personatges son:

Selecció de personatge: polsar el boto dret del ratolí sobre el personatge. També sobre el portrait del personatge.

Selecció múltiple per quadre de selecció: El primer pas es polsar sobre un punt del escenari. El segon pas es arrossegar el ratolí amb el boto polsat per crear el quadre.

Caminar: un cop seleccionat o seleccionats els personatges polsant al ratolí fins la part del escenari on es vol anar.

Selecció de habilitats: Per seleccionar les habilitats dels nostres personatges, en concret del arquer i del guerrer, s'han de seguir el següents passos.

- Pas 1: Seleccionar el personatge al que es vol aplicar la habilitat.
- Pas 2: Seleccionar la habilitat.

A la imatge 99 es poden veure els passos que es fan per la selecció d'una habilitat. El quadre de color vermell es el pas 1. El pas 2 es el quadre blau.



Imatge 99: Passos per la selecció d'una habilitat

Selecció de encantaments: Es pot escollir entre dos encantaments si el mag esta seleccionat:

- Armor: Per executar aquest encantaments s'han de seguir els següents passos:
 - Pas 1: Seleccionar el mag
 - Pas 2: Seleccionar l'encanteri corresponent al Hud.
 - Pas 3: Seleccionar un d'els personatges principals destinatari del encantament.

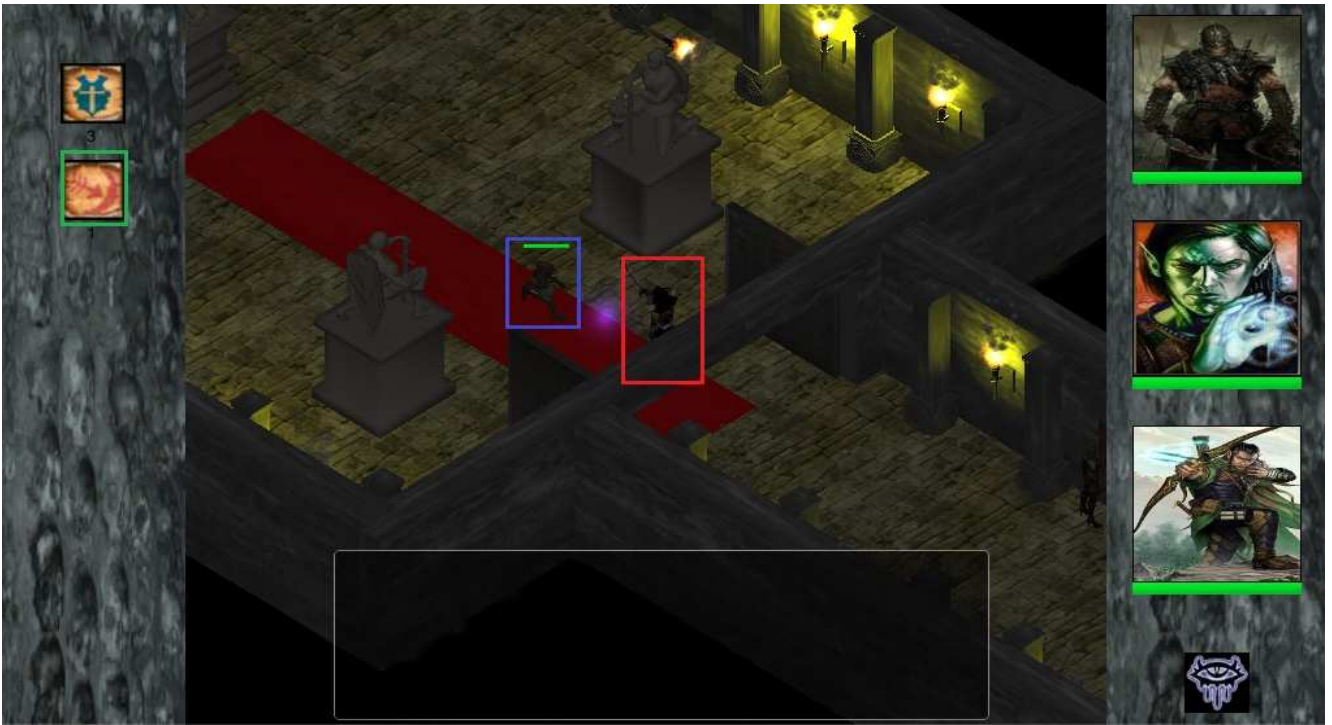
A la imatge 100 es poden veure els passos que es fan per executar l'encanteri Armor. El quadre de color vermell es el pas 1.El quadre de color verd es el pas 2. El pas 3 es el quadre de color blau.



Imatge 100: Passos per la selecció del encanteri Armor

- Magic Bullet: Per executar aquest encantaments s'han de seguir els següents passos:
 - Pas 1: Seleccionar el mag
 - Pas 2: Seleccionar l'encanteri corresponent al Hud.
 - Pas 3: Seleccionar el personatge enemic.

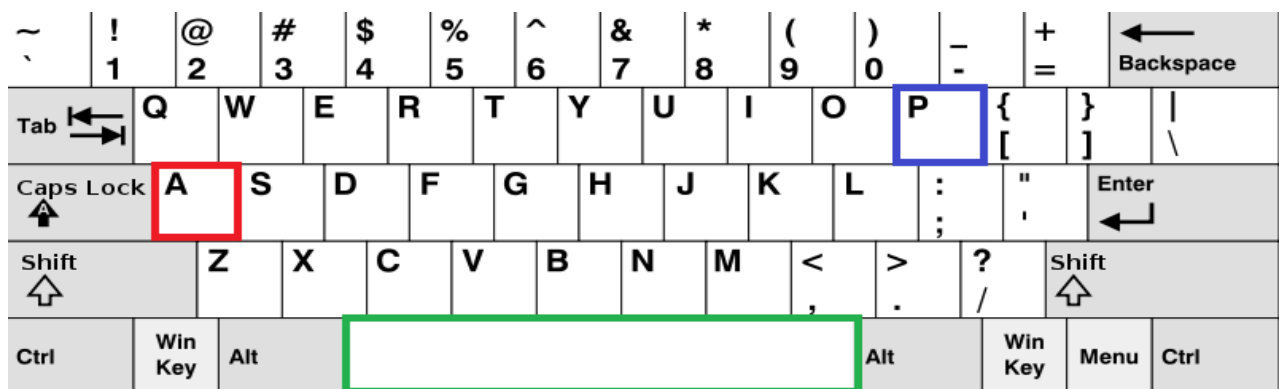
A la imatge 101 es poden veure els passos que es fan per executar l'encanteri Magic Bullet. El quadre de color vermell es el pas 1.El quadre de color verd es el pas 2. El pas 3 es el quadre de color blau.



Imatge 101: Estat final dels personatges principals la joc

Atac a un enemic: Per atacar a un enemic tan sols has de posicionar el ratolí sobre el enemic, el cursor del ratolí canviarà, i després polsar el boto. Automàticament el personatge atacarà.

A la imatge 102 es poden veure els controls dels personatges que s'utilitzen per teclat.



Imatge 102: Controls que s'utilitzen per teclat

El requadre vermell es la selecció múltiple dels personatges.

El requadre verd es parar el joc.

El requadre blau es obrir el menú de pausa del joc.

MENU

El menú principal disposarà de les següents opcions:

- Jugar: Iniciar una nova partida
- Opciones: Escollir la dificultat del joc
- Salir: Sortir del joc

MENU DE PAUSA

Un cop a la partida, per accedir a un menú dintre del joc s'hauria de polsat la tecla "P".

Es disposarà de les següents opcions:

- Continuar: Continuar la partida
- Salir: Sortir del joc.

10. Glossari

- PG: Vida que te un personatge o vida que es treu a un personatge.
- CA: Defensa de un personatge.
- Portrait: Representació de un personatge en un dibuix.
- Prefab: Tipus de GameObject reutilitzable que es pot inserir en qualsevol escena.
- XDY: tirada de daus, la Y significa el numero de cops que es llença un dau i la Y son les cares de un dau.
- Decal: Imatge que es utilitzada com a calcomania.
- Background: Fons de pantalla
- Fbx: Tipus de format utilitzat per
- Keyframe: Conjunt de posicions de les parts del esquelet en un determinat instant de temps que defineix la posicio d'un objecte.
- Asset: Objecte que es pot afegir a Unity3D
- Engine: motor grafic encarregat de processar videojocs.
- Tag: Identificador d'un objecte o conjunt d'ells.

