"Going Home"

Disseny i desenvolupament d'un videojoc d'aventura i presa de decisions

Pau Sabatés Campos

Resum— 'Going Home' és un videojoc en 2D que presenta una història on el protagonista ha d'aconseguir sortir del planeta en el qual es troba per tal de tornar a casa seva. Durant el transcurs d'aquest, les accions que prengui el jugador podran influir tant en el transcurs com en el final del joc, donant lloc a varis finals possibles. El desenvolupament d'aquest projecte s'ha fet utilitzant el motor gràfic Unity i la programació s'ha realitzat en C#. La principal característica del joc és el sistema de diàlegs amb els diferents personatges i el sistema de decisions, però també consta de moments de més acció al haver d'esquivar atacs d'enemics per poder avançar. Tot això vist des d'una perspectiva *top-down* amb un disseny artístic en 2 dimensions, del qual la seva millora gràfica utilitzant tècniques informàtiques com els Shaders és un dels altres punts forts del projecte. Per a la realització d'aquest projecte s'ha seguit la metodologia àgil Kanban a partir d'una planificació setmanal i s'ha obtingut com a producte una primera versió completa i estable del joc per a PC.

Paraules clau— Videojoc, C#, Unity, Decisions, 2D, Shaders, Kanban, PC.

Abstract— 'Going Home' is a 2D videogame that shows an history where the main character has to be able to get out of the planet where he is trapped and return to his house. During the course of the game, the actions that the player chooses may have an impact on the game, resulting this in some different endins. The development of this project was made using Unity and C#. The main feature of the game is the dialog system with all the other characters and the decision system, nevertheless, the game also features a dynamic combat where the player will have to evade all the enemies and projectiles. All of this using a top-down perspective and using a 2 dimensional sprites for the design, whose quality improvement usign computer programming skills like Shaders is another big point in this project. In order to achieve all of that, the metodology used for this project was Kanban using a weekly planification, and the resultin product of this project has been a complete and stable version of the game for PC.

Index Terms— Videogame, C#, Unity, Decisions, 2D, Shaders, Kanban, PC.

1 Introducció

Aquest projecte consisteix en el desenvolupament d'un videojoc d'un sol jugador amb una història i diàlegs entre personatges en el que les accions o decisions del jugador tenen repercussió durant el transcurs del joc. Per tant, s'ha desenvolupat en C# i Unity un sistema de diàleg i decisions per tal d'aconseguir-ho, a més s'han desenvolupat diferents tipus d'enemics i també s'han dibuixat tots els sprites del joc, els quals posteriorment s'han millorat gràficament amb l'utilització de shaders.

En els següents apartats d'aquest document s'explicarà tot el procés de desenvolupament d'aquest treball, començant per la motivació i objectius, després es parlarà de la planificació i metodologia realitzada, s'explicarà com s'han aconseguit desenvolupar els principals objectius del projecte, i finalment es presentaran els diferents resultats i conclusions.

1.1 Motivació

El món dels videojocs sempre m'ha apassionat, per tant vaig tenir clar d'es del principi que aquest treball havia d'estar relacionat amb aquest món, no només per gust personal, sinó també com pas per obrir portes en el món laboral dels videojocs. És per això que en el treball es va decidir fer servir una eina utilitzada en el món professional com Unity i tocar tants temes com sigués possible, des del dibuix i disseny de les diferents parts fins a la programació i utilització de tècniques utilitzades professionalment.

1.2 Estat de l'art

Per a la realització d'aquest videojoc es volia agafar de referència algun joc que hagués triomfat en els darrers anys i tingués una temàtica que encaixés amb la meva idea meva d'integrar decisions amb repercussions. El joc 'Undertale' per tant, publicat en els darrers anys amb

- *E-mail de contacte: pau.sabatesc@e-campus.uab.cat*
- *Menció realitzada: Enginyeria del Software.*
- *Treball tutoritzat per: Yolanda Benítez Fernández (departament de Ciències de la Computació)*
- *Curs 2018/19*

milions de còpies venudes, combina un estil 'pixel art' antic però innovador, amb una bona història i música que ho acompanya i finalment decisions en les accions del jugador que fan que l'experiència de joc depengui totalment del que decideix. [1]

Al voler fer un joc on la part de diàleg és important ja que és el que farà prendre una decisió o una altre al jugador, es van tenir en compte els jocs de l'empresa 'Double Fine' ja que tracten la narrativa com una mecànica més en el joc [2] fent així els diàlegs la part central.

1.3 Objectius

Els objectius que es van establir a l'inici del projecte van ser els següents, ordenats per prioritat:

- Diàleg amb personatges: els diferents personatges expliquen la història del joc al jugador mitjançant diàlegs.
- Accions amb repercussió: el jugador ha de poder veure que algunes de les seves accions tenen repercussió en el joc.
- Varis finals del joc: el joc ha de tenir diferents finals, depenent d'accions preses durant el seu transcurs.
- Joc gràficament atractiu: els nivells han de ser gràficament atractius pel que respecte a la seva ambientació i disseny.
- Combat simple i entretingut: el combat ha de ser simple d'entendre i entretingut pel jugador.
- '*User friendly*': controls fàcils d'entendre i interfície amigable. Així com realitzar accions per a que l'experiència d'usuari sigui la més satisfactòria possible
- Dificultat progressiva: la dificultat del joc augmenta a mesura que el joc avança.

Acordats els objectius del projecte, es van desglossar i traduir a tasques per tal d'organitzar-los al llarg del temps. La planificació d'aquestes tasques s'explica a continuació.

2 PLANIFICACIÓ

La planificació de les tasques es va fer utilitzant Microsoft Excel, amb una plantilla que generava un diagrama de Gantt.

Les tasques es dividien en blocs delimitats per milestones o fites, es va començar amb el disseny del joc i la programació de les parts més crítiques i importants del projecte.

Un cop es va tenir una versió amb diàlegs i decisions es va començar el desenvolupament del combat dels enemics. L'aplicació de shaders per millorar el joc gràficament es va realitzar al final ja que era una tasca amb molta incertesa al no tenir cap coneixement del tema, per tant es va decidir d'integrar-ho un cop ja hi hagués una versió bastant completa del joc.

La durada de els tasques va ser bastant encertada i es va poder portar el projecte al dia, tot i així durant el seu transcurs algunes tasques van requerir més temps o es van poder començar abans, en el diagrama adjunt al apèndix es mostra amb un color vermell fosc.

A continuació es pot veure la plantilla amb les tasques:

	Start	End	Duration	Milestones	
Desenvolupar idea treball	7-Feb	17-Feb	11.0		Activity
Reunió Inicial	21-Feb	21-Feb	0.0	1.0	Milestone
Realitzar Informe Inicial	17-Feb	8-Mar	20.0		Activity
Entrega Informe Inicial	10-Mar	10-Mar	0.0	1.0	Milestone
Disseny del personatge principal	26-Feb	3-Mar	6.0		Activity
Moviment del personatge principal	26-Feb	3-Mar	6.0		Activity
Implementació 3 personatges	26-Feb	6-Mar	9.0		Activity
Mecànica de diàleg i decisions	6-Mar	17-Mar	12.0		Activity
Implemetació de l'escenari principal	17-Mar	24-Mar	8.0		Activity
Implementació del nivell 1,2 de la torre i ciutat	17-Mar	24-Mar	8.0		Activity
Implementació del nivell 3,4 de la torre i ciutat	24-Mar	3-Apr	11.0		Activity
Build de la v.Alpha1.0, testeig i bug fixing	29-Mar	13-Apr	16.0		Activity
Realitzar Informe de Progrés 1	30-Mar	14-Apr	16.0		Activity
Entrega Informe de progrés 1 + v.Alpha1.0	14-Apr	14-Apr	0.0	1.0	Milestone
Disseny diferents enemics	21-Apr	28-Apr	8.0		Activity
Implementació combat nivell 1,2 torre	28-Apr	5-May	8.0		Activity
Implementació combat nivell 3 torre	5-May	12-May	8.0		Activity
Aplicar shader gràfics	12-May	19-May	8.0		Activity
v.Alpha2.0	20-May	20-May	0.0	1.0	Milestone
Realitzar Informe de progrés 2	15-May	25-May	11.0		Activity
User testing	20-May	25-May	6.0		Activity
Informe de progrés 2 + v.Beta1.0	26-May	26-May	0.0	1.0	Milestone
Bug fixing, testing, user testing	27-May	3-Jun	8.0		Activity
Realitzar Informe Final	4-Jun	15-Jun	12.0		Activity
Realitzar documentació concreta del joc	15-Jun	28-Jun	14.0		Activity
Informe Final	16-Jun	16-Jun	0.0	1.0	Milestone
Realitzar presentació	17-Jun	28-Jun	12.0		Activity
Proposta de presentació + Lliurament Dossier	30-Jun	30-Jun	0.0	1.0	Milestone
Realitzar pòster	1-Jul	6-Jul	6.0		Activity
Practicar presentació	1-Jul	10-Jul	10.0		Activity
Lliurament del Pòster	7-Jul	7-Jul	0.0	1.0	Milestone
Prensetació al Tribunal TFG	11-Jul	11-Jul	0.0	1.0	Milestone

Fig. 1. Planificació de tasques

3 METODOLOGIA

Per a la realització i organització del projecte s'ha utilitzat la metodologia Kanban. La raó és que en general, el desenvolupament d'un videojoc no segueix un cicle de software similar al desenvolupament en cascada (*Waterfall*), sinó que es basa més en iteracions, i en cada iteració anar generant prototips, és a dir metodologies àgils de software prototyping.

Aquesta metodologia funciona especialment bé en els casos que els requisits inicialment no estan del tot clars, en aquest cas però, al tenir-los clars, es va decidir utilitzar una metodologia similar que es basa en el que més pot ajudar, tenir planificades les tasques i objectius que s'han de fer en cada iteració però amb la flexibilitat de poder-les moure i canviar d'ordre.

Kanban per tant permet això, mitjançant una taula amb tres o quatre columnes (backlog, to do, doing, done) permet visualitzar el flux de treball actual i per tant poder organitzar les tasques a fer en cada moment (programació, disseny, art) depenent del que es cregui més convenient i amb un límit de tasques a la columna 'doing', per tal de no voler fer moltes tasques a la vegada i que acabi sent poc òptim. Un cop es van realitzant les tasques de la taula, s'incorporen de noves des de la planificació general.

Per a la seva realització es va utilitzar l'eina Trello ja que visualment treballa amb targetes, com és ideal per Kanban, i és gratuïta.

4 ANÀLISI DEL SISTEMA

Per tal de documentar aquest projecte de software, es va realitzar un anàlisi de requisits, de casos d'ús i un diagrama de classes.

4.1 Requisits funcionals

- RF1: El jugador ha de poder prendre decisions mitjançant les seves accions.
- RF2: Entendre i implementar tècniques d'il·luminació dinàmica per millorar artísticament el joc a través de la programació de shaders.
- RF3: El joc permetrà el moviment del personatge en totes direccions, i permetrà que pugui atacar si és una zona de combat.
- RF4: Hi haurà tres NPC (Non playable character) principals, amb diferents diàlegs depenent del moment de la història.
- RF5: Els diferents NPC presentaran com a mínim una decisió al jugador durant el transcurs del joc que poden fer canviar el final del mateix.
- RF6: Hi haurà tres enemics principals amb diferents atributs per fer més desafiants els combats.
- RF7: Realitzar un combat engrescador i desafiant mitjançant enemics amb intel·ligència artificial.
- RF8: Aconseguir un rendiment acceptable de fotogrames per segon durant l'execució del joc.
- RF9: Implementar un sistema de guardat i càrrega de partides eficient i fiable.

4.2 Requisits no funcionals

- RNF1: Verificar correcte funcionament amb *user testing*.
- RNF2: El joc ha de ser fàcil d'entendre per l'usuari.
- RNF3: El temps de duració del joc no serà superior a les 4 hores
- RNF4: El joc podrà ser jugat en ordinadors amb mínim 4 gb de RAM i un processador de doble nucli.
- RNF5: El joc ha de ser executable en les últimes versions de Windows (windows 10).
- RNF6: Realitzar els diferents informes requerits per l'avaluació del TFG.

4.3 Diagrama de casos d'ús

Per representar els diferents actors i accions que permet el joc a nivell funcional es van realitzar dos diagrames de casos d'ús. El que es pot veure a la Fig.2 fa referència als casos d'ús dels personatges enemics, el cas d'ús del jugador es pot trobar a l'apartat A2 de l'appèndix.

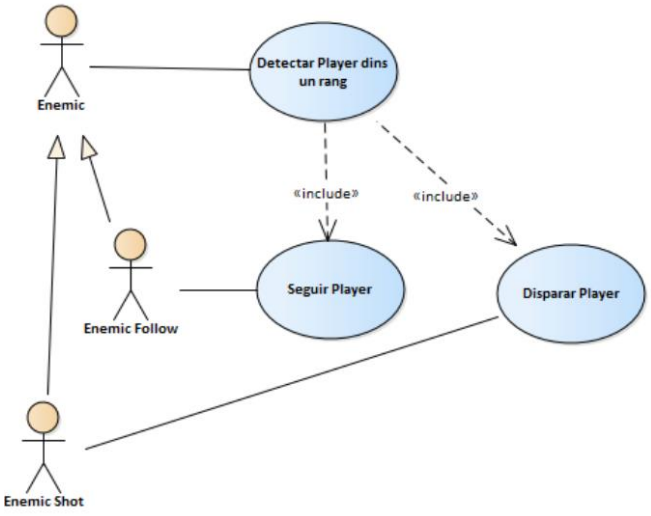


Fig. 2. Casos d'ús enemics

4.4 Diagrama de classes

El diagrama de classes del codi del joc es pot veure en l'apartat A3 de l'apèndix. En ell es pot veure com s'ha intentat modular el màxim del possible les diferents parts per intentar que siguin el més atòmic possible i així evitar modificar molt codi si es volen fer canvis. En els següents punts s'explica en més detall algunes classes com les de diàleg i decisions.

En el diagrama però per tal de fer-lo llegible i útil s'han obviat les classes de test i moltes de les dependències que tenen algunes classes amb classes del motor d'Unity. La classe d'Unity que si queda representada es la 'MonoBehaviour' ja que és la més important, d'ella s'hereten les funcions start() i update() que serveixen per executar codi a cada fotograma del joc i al ser instanciada.

5 DESENVOLUPAMENT

En les següents seccions s'explica com s'han implementat les principals característiques del joc per tal de satisfer els principals objectius.

El videojoc desenvolupat consta d'un sistema de diàleg on al jugador no se li proposen diverses opcions a respondre, sinó que la seva decisió en forma d'acció (per exemple si decideix ajudar o no) serà el que pot canviar el transcurs de la història en determinats moments. A més, el videojoc consta d'enemics els quals el jugador haurà d'esquivar els atacs, per tal de fer-lo més complet.

Per a realizar el projecte s'ha utilitzat el motor Unity3D ja que és considerat excel·lent per equips de desenvolupament petits i amb capacitat de crear contingut de molt alta qualitat [3]. El joc agafat de referencia, comentat a l'estat de l'art, es va fer utilitzant el motor GameMaker, però es va preferir Unity ja que a part de ser més potent i gran, és molt més utilitzat en el món laboral.

Unity també és molt útil per ajudar en la part artística, ja que al haver de centrar-me jo més en la programació,

m'ho facilitarà amb l'eina TilePalette [4] per crear escenaris ràpidament i també gràcies a la seva possibilitat de programar 'shaders' per crear il·luminació dinàmica [5], tècnica coneguda en el món professional per utilitzar la GPU per crear aquesta il·luminació volumétrica [6].

Entre les pantalles que el joc presenta es distingeixen dos principals, la pantalla d'una ciutat que és on es troben les cases dels personatges, cada una amb la seva pantalla, i la pantalla de la torre on es troben els diferents nivells amb enemics. En total hi han 4 personatges a part del protagonista amb qui mantenir converses i 3 tipus diferents d'enemics.

El sistema de decisions, implementat al llarg de deu pantalles, permet que hi hagin 3 possibles finals diferents; un final possible és el que surt en el cas de que el jugador hagi sentit empatia per cert personatge i acabi confiant amb ell, un altre final sorgeix si s'actua de manera contrària a l'anterior, i finalment un final 'perfecte' que sorgeix d'una combinació del primer final possible i a més a més haver parlat amb tots els personatges en totes les pantalles. A part hi ha una altre decisió durant el joc on pots fer que un personatge acabi sent ric o no, però sense que influeixin el final.

Pel que respecte al combat amb els enemics es va decidir aplicar un combat d'estil 'bullethell', és a dir, avançar de pantalla esquivant atacs dels enemics. S'han dissenyat 3 pantalles de combat en el joc on la dificultat es va incrementant i apareixen combinacions dels diferents tipus d'enemics.

5.1 Sistema de diàleg

El sistema de diàleg consisteix en la comunicació del jugador amb els altres personatges. Aquests expliquen la història o proposen decisions depenent del moment en el que el jugador es troba en el joc.

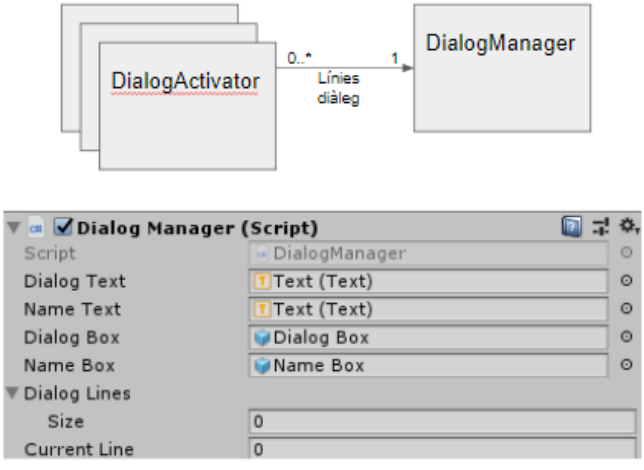


Fig. 3. Representació Sistema de diàleg

Cada personatge té un DialogActivator en cada situació diferent que conté les línies del diàleg a mostrar. També té una funció que està en tot moment comprovant si s'activa la tecla per parlar estant a certa proximitat del personatge.

La clase DialogManager s'encarrega d'agafar les línies i mostrar-les en la UI anomenada Dialog Box. Una única instància de la clase és suficient per tots els diàlegs ja que només hi haurà un diàleg actiu al moment.

5.2 Sistema de decisions

El sistema de decisions fa referència als canvis que es produeixen en el joc en funció de si el jugador pren una acció o una altre en determinats moments.

La clase 'DecisionActivator' indica que una decisió s'ha fet o no modificant la llista de 'DecisionManager'. Per altre banda, la clase 'DecisionManager' conté una llista amb les diferents decisions del joc, i finalment, la clase 'DecisionObjectActivator' comprova si la decisió ha estat completada o no. Depenent d'això es desencadenarà una acció o una altre. De forma gràfica es pot veure a la figura 4:

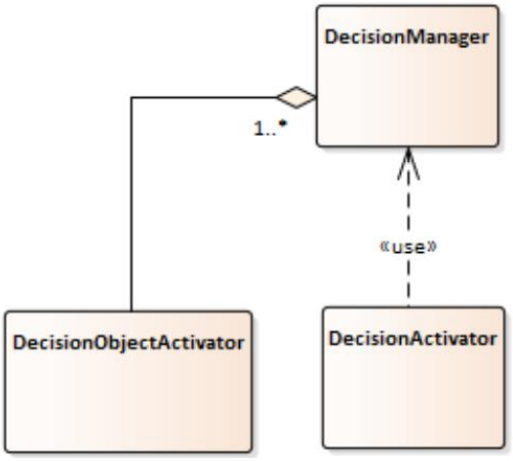


Fig. 4. Diagrama de classes del sistema de decisions

5.3 Sistema de guardat i càrrega

Per guardar la partida (pantalla, posició del jugador, decisions preses) s'utilitzen els PlayerPrefs de Unity. El que es fa és guardar valors en tipus enter o string en registres del sistema operatiu.

Es van contemplar altres alternatives com guardar i carregar els valors des de fitxers externs, però la solució dels PlayerPrefs és la més senzilla i ràpida d'aplicar i el seu funcionament era el desitjat, a més d'evitar d'aquesta forma problemes de directoris i permisos de fitxers.

5.4 Disseny d'escenaris

Per a construir les diferents pantalles amb els seus sprites i quins col·lisionen amb el jugador s'ha utilitzat un sistema de capes amb el motor de Unity.

Tal com es veu en les següents imatges, amb l'eina de Tile Palette s'apliquen els sprites, prèviament creats, a la escena de Unity, i es creen diferents capes per tal de crear varis nivells de renderitzat en 2 dimensions, és a dir, sprites que es mostren per sobre el jugador, per sota, i quins col·lisionen amb ell.

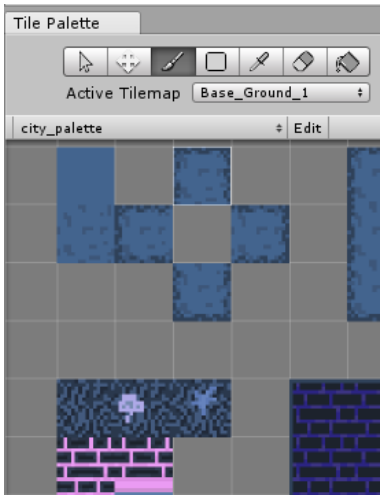


Fig. 5. Interfície eina Tile Palette

En les següents imatges es poden veure el nom de les diferentes capes dins un mateix grid, on per exemple al seleccionar la capa de col·lisions, es mostra al editor a quins sprites se'ls hi ha aplicat col·lisió:

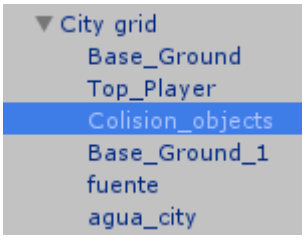


Fig. 6. Capes

Fig. 7. Representació dels sprites amb col·lisió

5.5 Intel·ligència dels enemics

En aquest apartat s'explicarà com s'han implementat les accions dels enemics; perseguir al jugador o disparar-lo.

La solució no va ser aplicar un algoritme d'intel·ligència artificial com el A* com es va pensar d'es d'un inici. Es va decidir la següent solució, més senzilla i que requereix menys càlcul computacional.

Tal com es pot veure a la figura 8 la posició del Jugador esta representada per vector 'P' i la del enemic pel vector 'E'. Per tant la direcció cap a la que s'ha de moure l'enemic és simplement el vector 'D' que va de E a P. Per calcular-ho es pot aplicar la resta d'ambdós vectors i s'obtindrà el nou vector direccional amb origen a E.
Finalment per tant s'aplicarà aquesta nova direcció a l'enemic i es desplaçarà cap aquella direcció i a cada frame del joc es va recalculant aquesta direcció per fer un seguiment del jugador.

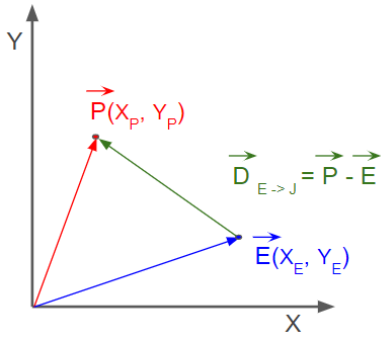


Fig. 8. Resta de vectors del jugador i enemic

En el joc però, també hi ha un altre tipus d'enemic diferent, el qual dispara projectils cap al jugador. Per implementar-ho per tant s'ha utilitzat el mateix raonament.

Simplement en aquest cas però, en comptes d'aplicar el vector de direcció resultant al enemic, s'aplica al projectil que es dispara i aquest llavors va cap a la posició del jugador.

Tot i això, la imatge del projectil s'ha d'enfocar cap a la direcció a la que es dirigeix també, ja que sinó el projectil pot anar en diagonal cap a dalt per exemple però la imatge del projectil estar apuntant per defecte cap a la dreta, és a dir, sense rotar cap a la direcció desitjada.

Per solventar-ho s'ha aplicat una coneguda funció matemàtica: arctan2 (En unity utilitzada com 'Mathf.Atan2') que calcula l'angle entre l'eix horitzontal (x) i el vector de direcció actual, així sabrem quants graus (angle θ) girar la imatge per enfocar-la a la mateixa direcció a la que es dirigeix. A la figura 9 es pot veure la gràfica que ho representa:

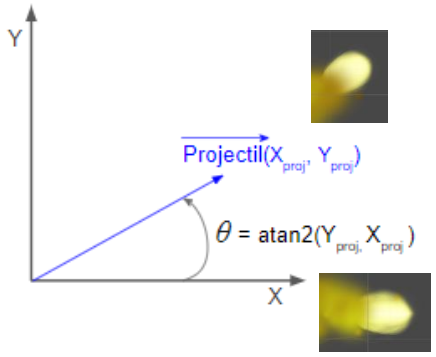


Fig. 9. Rotació del projectil

5.6 Millora de l'apartat gràfic utilitzant shaders

Fins ara l'apartat gràfic del joc consistia en renderitzar sprites en dos dimensions, per tant la qualitat de l'apartat artístic depenia completament de la qualitat del dibuix del sprite. Al voler millorar l'apartat gràfic però sense utilitzar tècniques artístiques, es va pensar pensar en millorar-lo a través de la programació, és a dir a través de shaders.

Un shader es un script que conté algoritmes i càlculs matemàtics per calcular el color de cada píxel processat en funció de la il·luminació que se li aplica i de la configuració del 'material', el qual defineix com ha de renderitzar-se una superfície.

Per tant, a continuació s'explicarà el shader programat per aquest projecte.

5.7 Estudi d'elecció del shader i explicació de l'implementació d'il·luminació dinàmica

La utilització de shaders és tot un món, hi han molts tipus, diversos llenguatges i diferents maneres d'aplicar-los depenent de l'objectiu que es vol aconseguir amb ells. Per tant decidir quin era l'adequat pel projecte era un pas important.

L'objectiu era aconseguir programar un shader que tractés les llums/il·luminació que hi hagin en pantalla d'una manera dinàmica, és a dir que els altres objectes es vegin afectats en temps real per la intensitat, rang, color, etc, d'aquests punts d'il·luminació.

Principalment els shaders es poden categoritzar en tres grans tipus dins el motor d'Unity: 'Surface shaders', 'Vertex and Fragment shaders' i 'Fixed function shaders':

- Surface shaders: És la opció escollida ja que són els més òptims a l'hora de tractar llums i ombres. Et permeten programarlos de manera que tractes amb una capa superior més abstracta que interactua amb el pipeline d'il·luminació d'Unity i facilita la seva programació. Com a inconvenient són molt poc òptims per efectes de post-processament i altres efectes de shaders més especials ja que realitzaria molts càlculs d'il·luminació sense sentit.

- Vertex and Fragment shaders: Són els més indicats per si no és necessari interactuar amb la il·luminació. S'utilitzen normalment per crear grans efectes visuals. És cert que un surface shader acaba convertint-se en un fragment shader internament i poden utilitzar les mateixes funcions, però això és algo més intern del motor gràfic d'Unity al compilar, de base un fragment shader no tindria sentit per el meu projecte ja que no aconseguiria l'efecte d'il·luminació dinàmica.

- Fixed function shaders: Estan pensats per hardware antic que no soporta shaders programables, és a dir, són un tipus de shader amb una funció molt simple ja predefinida i no modificable que al consumir tant pocs recursos asseguren que l'objecte s'acabarà renderitzant. Un shader d'aquest tipus s'utilitzarà en el projecte pel que es coneix com a funció 'fallback', que és la que s'activaria si el meu shader principal no el pogués renderitzar la GPU de l'ordinador del jugador.

5.8 Constitució del shader

El shader implementat aplica un surface shader en un normal map en un material d'Unity.

El normal map és una técnica que s'utilitza per donar relleu a una superfície de tal manera que es redueix el número de polígons de la figura original i per tant els càlculs d'il·luminació s'apliquen sobre aquest mapeig.

Per el meu projecte el meu normal map per el personatge es simplement una superfície llisa, ja que al aplicar-se sobre una imatge (que no tindrà cap vèrtex com tindria una figura en 3 dimensions) es pot aplicar un normal map d'aquest estil així millorar el rendiment a l'hora de calcular la il·luminació aplicada. L'únic inconvenient és que la imatge tindrà la mateixa il·luminació per cada un dels seus pixels d'una manera més homogènia i quedaria millor si per exemple la part de la cara tingués relleu per tal de que es generessin ombres en la mateixa cara i rebés diferents intensitats d'il·luminació, però la realització d'aquest modelat està fora de l'abast d'aquest projecte.

On si s'han aplicat diferents normal maps ha sigut en parets o en el terra dels diferents escenaris.
A continuació s'explicarà com funciona el shader, els normal maps utilitzats i els resultats obtinguts.

5.9 Llenguatge del Surface shader

Al escollir programar un surface shader implica que el llenguatge de programació necessari serà el 'Cg/HLSL' amb una capa per sobre que és el 'ShaderLab' que donarà l'estructura necessària per a que el motor Unity ho entengui.

Si el joc utilitzés un altre motor per renderitzar els gràfics com per exemple OpenGL o Direct3D o Metal framework

d'Apple, el llenguatge per programar aquest shader podria ser diferent.

En el cas de Unity però, s'utilitza Cg, que és un llenguatge desenvolupat per Nvidia basat en C però pensat per programar per a la GPU i no com a llenguatge de propòsit general. A més a més, Unity utilitza el llenguatge HLSL també per programar els shaders, per tant s'utilitzen els dos de manera complementària, ja que aquest segon permet utilitzar els vertex and fragment shaders.

La funció principal del shader és la següent:

```
void surf (Input IN, inout SurfaceOutput o)
{
    fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * IN.color;
    o.Albedo = c.rgb;
    o.Alpha = c.a;
    o.Normal = UnpackNormal(tex2D(_BumpMap, IN.uv_BumpMap));
}
ENDCG
```

```
struct Input
{
    float2 uv_MainTex;
    float2 uv_BumpMap;
    fixed4 color : COLOR;
};
```

Fig. 10. Funcions del shader

La funció 'surf' de la imatge s'encarrega de rebre un struct Input amb tota la informació del material, textura, dades en general, que al compilar generarà un Output en funció dels Inputs que es convertirà en un vertex shader que es tractarà en una altre funció.

L'altre part que es podria considerar més important del shader és que està dividit en dos parts. Primer es renderitza la part del sprite que el jugador veu i se li aplica el shader, a continuació s'aplica el mateix codi però per la part no visible del sprite, ja que així ambdues tenen el shader aplicat. Això s'indica amb les instruccions 'Cull Back' i 'Cull Front', i a més invertint els eixos y i z on les funcions apliquen el shader.

```
ENDCG

// Render de la part de darrera
Cull Back
Lighting On
ZWrite Off
Fog { Mode Off }

CGPROGRAM
#pragma surface surf Lambert alpha vertex:ve
#pragma multi_compile DUMMY PIXELSNAP_ON
```

Fig. 11. Codi per renderitzar part frontal i posterior

5.10 Resultats del shader

En el següent diagrama es veu de manera visual els diferents materials associats amb el corresponent sprite i el resultat al aplicar el shader:

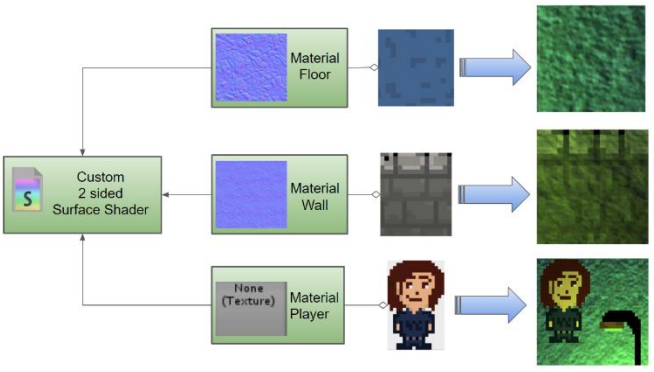


Fig. 12. Diagrama de l'aplicació del shader

El material del Player no té cap textura ja que fer un mapeig del personatge i transformar-ho a un normal map no és un objectiu del treball, però igualment es pot apreciar com el shader segueix aplicant-se ja que es veu afectat per l'il·luminació.

6 TESTS

Durant la realització d'aquest projecte s'han realitzat certes proves per verificar un cert nivell d'estabilitat i qualitat del joc.

6.1 Tests unitaris

Per tal de testejar el codi del projecte es va fer unit testing utilitzant l'eina Test Runner d'Unity. Aquesta eina et permet definir els tests en codi utilitzant el framework nUnit per C# permetent així testejar tant classes vinculades amb objectes d'Unity com classes a part. Un exemple de test es pot veure a continuació:

```
[Test]
public void onTriggerEnterTest()
{
    //ha de retornar true si el player entra
    var collider = new GameObject().AddComponent<BoxCollider2D>();
    collider.gameObject.tag = "Player";
    DialogActivator dial_act = new DialogActivator();
    dial_act.OnTriggerEnter2D(collider);

    Assert.IsTrue(dial_act.getCanActivate());

    //ha de retornar false si entra un altre objecte que no es el player
    collider.gameObject.tag = "enemy";
    DialogActivator dial_act1 = new DialogActivator();
    dial_act1.OnTriggerEnter2D(collider);

    Assert.IsFalse(dial_act1.getCanActivate());

}
```

Fig. 13. Exemple d'una funció de test

S'han fet per tant tests a les funcions més importants del joc, com per exemple els sistema de diàlegs i decisions. La interfície del Test Runner amb un resultat d'execució es pot veure a continuació:

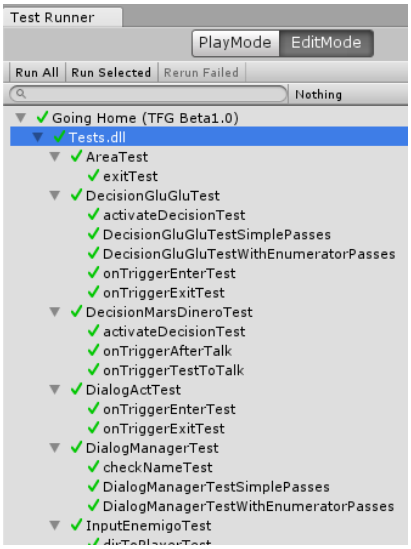


Fig. 14. Interfície del Test Runner amb resultat de l'execució

També per poder provar les diferents pantalles en qualsevol moment i no haver de jugar el joc fins arribar a certa pantalla per provar-la, es va fer que les diferentes dependències que podria tenir una pantalla(que existís un jugador, laUI del diàleg inicialitzada, etc) es guardèssin com un 'prefab' d'Unity. Un prefab actua com una classe pare de l'objecte la qual es pot instanciar com a classes filles en les diferents pantalles heretant de la classe pare i modificant els atributs que es vulguin per aquell test en concret.

6.2 *User testing*

Per complir amb el requisits de que el joc fos amigable, divertit i entenedor, es van realitzar proves amb usuaris i es van recopilar les seves respostes d'un formulari que se'ls hi entregava. El total de persones que van provar el joc van ser 8 i es poden veure alguns dels resultats extrets en forma de gràfica al punt A4 de l'apèndix. El formulari consta de més preguntes de les que es mostren en aquest informe, però aquestes seleccionades són les més representatives de l'opinió general.

Els resultats obtinguts van ser positius ja que el que més valoren del joc són els dos objectius principals del projecte i també va anar bé per veure que un 25% dels usuaris van tenir problemes al executar al joc i vaig poder identificar i solucionar el problema.

7 RESULTATS

En aquest apartat es presentaran els principals resultats obtinguts un cop acabat el projecte i també s'analitzaran comparant-los amb els objectius que es van proposar inicialment.

7.1 Resultats del projecte

El resultat o producte final obtingut d'aquest projecte consisteix en un videojoc per PC on el jugador ha de controlar un personatge per les diferents pantalles parlant amb altres personatges, prenent decisions i enfrontant-se a enemics.

Un dels principals objectius que tenia el projecte era la creació d'un sistema de diàleg i decisions. La solució aplicada és considerada una bona solució ja que és simple i fàcil d'anar-la implementant al desenvolupar cada decisió del joc. A més a més és bastant escalable ja que al estar modulada en diferents classes es podria adaptar a un sistema de missions dins el joc per exemple, o amb el temps també tenir diferents classes per a diferents accions que poden passar amb una mateixa decisió (actualment s'activen i desactiven objectes o personatges) i tot sense haver de modificar la classe activadora de la decisió.

Com a millora d'aquesta part del sistema de diàleg i decisions, pensant ja més en una continuïtat del projecte i crescuda del mateix en un futur, seria interessant aplicar una localització del text per tal de que pogués ser adaptat a diferents idiomes.

En les següents imatges es pot veure el resultat d'haver escollit una acció que fa que un personatge sigui ric o no:

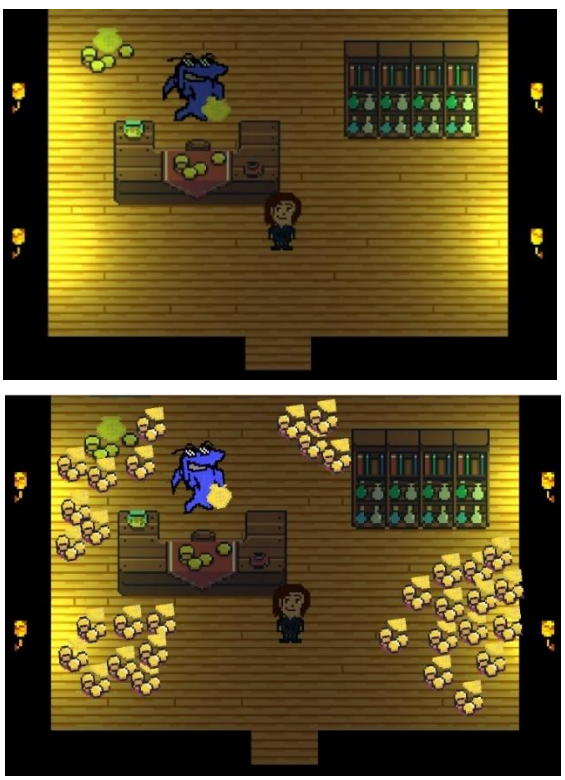


Fig. 15. Comparativa imatges al prendre una decisió o no

Un altre gran punt i objectiu que es volia aconseguir amb el projecte era la millora de l'apartat gràfic utilitzant tècniques informàtiques i no artístiques.

Aquest objectiu ha sigut el que més temps ha portat completar-lo ja que al ser el meu primer contacte amb la tecnologia dels shaders vaig haver de documentar-me molt més del que tenia pensat. Tot i així es va acabar complint en el temps planificat la seva implementació, i tot i que pot ser millorada, l'actual implementació compleix l'objectiu que es pretenia aconseguir sens dubte.

El rendiment del joc no es veu pràcticament afectat amb la utilització de shaders i també es va tenir en compte en que es desactivessin per si l'ordinador de l'usuari no els suportés. El resultat per tant d'aquest objectiu és més que satisfactori ja que és el primer que crida l'atenció del joc als usuaris que l'han provat. Els objectius restants prevists per complir fan referència als enemics i al *user testing*.

Respecte a la programació dels enemics es va optar per programar la seva intel·ligència a través d'operacions entre vectors de posició en comptes d'aplicar algun algoritme d'intel·ligència artificial de pathfinding. Com a resultats, s'ha obtingut un codi més senzill i que requereix menys càlcul computacional, a més a més de poder utilitzar el mateix càlcul per l'enemic que persegueix el jugador com per el projectil que va cap al jugador.
A efectes pràctics, la meva solució per aquest projecte no es distingeix de l'alternativa d'aplicar pathfinding, tot i si que és cert que si el mapa tingués moltes parets on es pot quedar encallat l'enemic, l'algoritme de pathfinding seria una millor opció.

En les següents imatges es pot veure una comparativa de la millora gràfica que ha implicat aplicar shaders i un enemic disparant un projectil que persegueix el jugador:

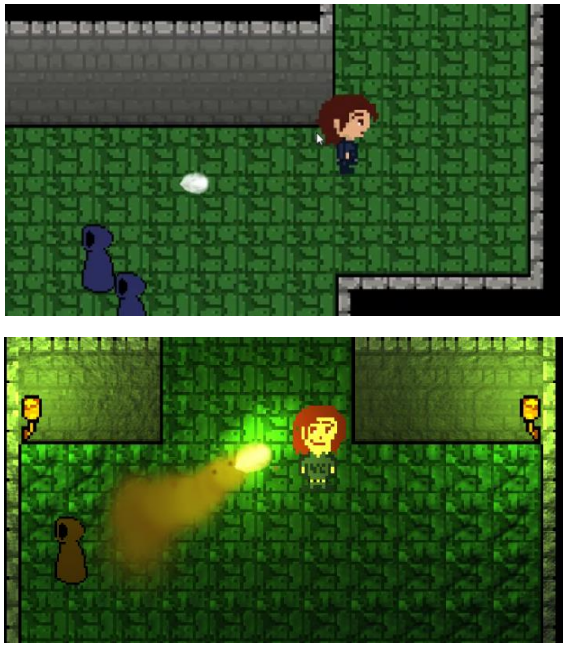


Fig. 16. Comparativa d'imatges al aplicar el shader

Pel que respecte a l'objectiu de *user testing* es va realitzar cap al final del projecte, hagués sigut ideal poder anar-lo realitzant durant el desenvolupament també, per exemple a cada fita, però per per gestió del temps i de persones voluntàries a realitzar el test es va decidir fer-ho al final del projecte. Els resultats obtinguts com ja s'ha comentat van ser positius i van servir per arreglar errors i assegurar que funcionaven els principals objectius del projecte i els valoraven positivament.

8 CONCLUSIONS I LÍNIES OBERTES

A continuació s'exposaran les conclusions d'haver realitzat aquest projecte i com pot evolucionar en un futur.

8.1 Línies obertes

Per tal de millorar el projecte es podrien afegir noves pantalles per allargar la durada del joc. Gràcies a com està fet el diagrama de classes i els objectes d'Unity, la addició de noves pantalles amb noves decisions no implicaria quasi tocar codi i per tant només caldria temps per poder dissenyar aquestes noves pantalles.

Un altre punt interessant seria fer diferents tipus de shaders per cada un dels diferents objectes del joc, i no utilitzar un per vàries coses com es fa actualment, així encara es milloraria més la part gràfica del joc.

També es podria millorar el diagrama de classes, es bastant modular i serà fàcil la seva escalabilitat, però si que hi han algunes classes amb moltes dependències i això podria dificultar la realització de tests unitaris.

8.2 Conclusions

Aquest ha sigut un projecte que des d'un inici s'ha portat al dia la planificació, les tasques van estar ben planificades i ben prioritzades al aplicar la metodologia per treballar, això ha permès que s'hagin pogut complir tots els objectius que es van decidir i que els canvis que hi han hagut durant el projecte a la planificació no acabèssin afectant el resultat final per possibles retards.

L'experiència realitzant aquest projecte ha sigut molt enriquidora ja que he desenvolupat un videojoc aprenent a utilitzar un motor gràfic profesional com Unity, he fet i integrat els diferents sprites de cada personatge i tot el codi per fer el joc que tenia pensat, i finalment he descobert el món dels shaders, on he après com programar-los i veure la gran utilitat que poden tenir.

AGRAÏMENTS

M'agradaria agrair primer de tot a la meva tutora Yolanda Benítez, per guiar-me i aconsellar-me durant el projecte. També agrair a un company, Àlex Casals, per fer-me una cançó pel joc de manera gratuïta, i finalment agrair a tots els que m'han recolzat durant el projecte i als qui s'han ofert de voluntaris per fer els tests d'usuari i han dedicat part del seu temps per donar-me feedback.

BIBLIOGRAFIA

[1] Melody Madhavan, "How Indie Game Undertale Became A Top Selling Game On Steam In 2015" [en línia]. [Últim accés: 17 de Febrer de 2019]. Disponible a: https://www.referralcandy.com/blog/undertale-marketing-strategy/

[2] Game Development Conference, "Dialogue systems in Double Fine Games" [en línia]. [Últim accés: 17 de Febrer de 2019]. Disponible a: https://www.youtube.com/watch?v=0hMiPBe_VRc

[3] Game designing, "Most popular video game engines" [en línia]. [Últim accés: 24 de Febrer de 2019]. Disponible a: https://www.gamedesigning.org/career/video-game-engines/

[4] Unity3d, "TileMap Palette Manual" [en línia]. [Últim accés: 24 de Febrer de 2019]. Disponible a: https://docs.unity3d.com/Manual/Tilemap-Palette.html

[5] Unity3D Documentation, "Materials, Shaders & Textures" [en línia]. [Últim accés: 26 de Febrer de 2019]. Disponible a: https://docs.unity3d.com/Manual/Shaders.html

[6] ItHare, "Game Graphics 101: Rendering 2D on GPU. Shaders" [en línia]. [Últim accés: 26 de Febrer de 2019]. Disponible a: http://ithare.com/game-graphics-101-rendering-2d-on-gpu-shaders/

[7] Carmichael, Andy. J. Anderson, David (2016). 'Essential Kanban condensed'. Seattle, Washington. Lean Kanban University Press. Disponible a: http://leankanban.com/wp-content/uploads/2016/06/Essential-Kanban-Condensed.pdf

[8] Docs Unity3d, "PlayerPrefs" [en línia]. [Últim accés: 3 d'abril de 2019]. Disponible a: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html

[9] Docs Unity3d, "Scene management" [en línia]. [Últim accés: 3 d'abril de 2019]. Disponible a: https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.LoadScene.html

[10] Slynyrd, "Light and shadow pixelart" [en línia]. [Últim accés: 3 d'abril de 2019]. Disponible a: http://www.slynyrd.com/blog/2018/6/15/pixelblog-6-light-and-shadow

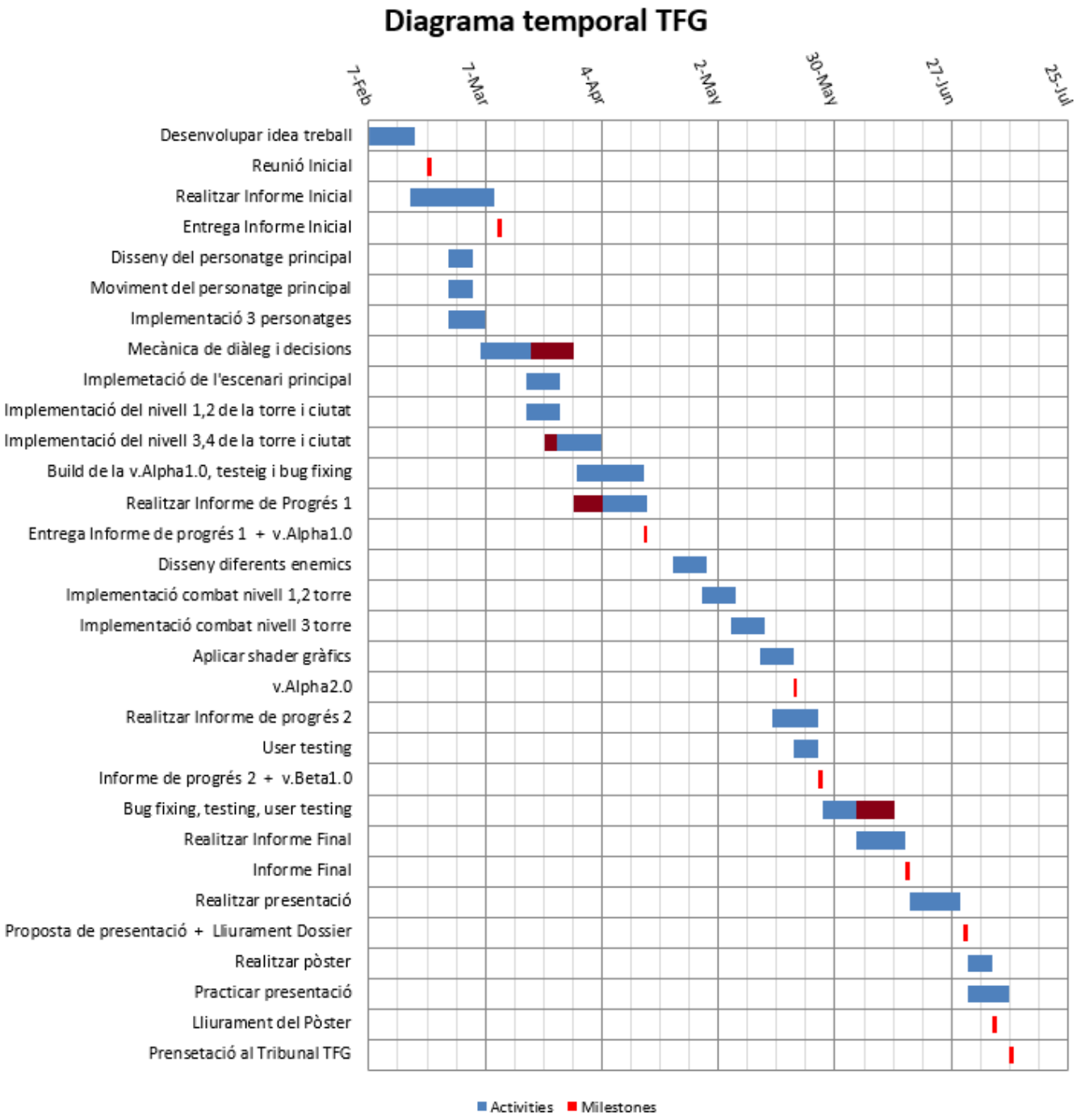
[11] Pyxel Edit [en línia]. [Últim accés: 4 de març de 2019]. Disponible a: https://pyxeledit.com/

[12] Docs Unity3d, "Writing shaders" [en línia]. [Últim accés: 19 de maig de 2019]. Disponible a: https://docs.unity3d.com/Manual/ShadersOverview.html

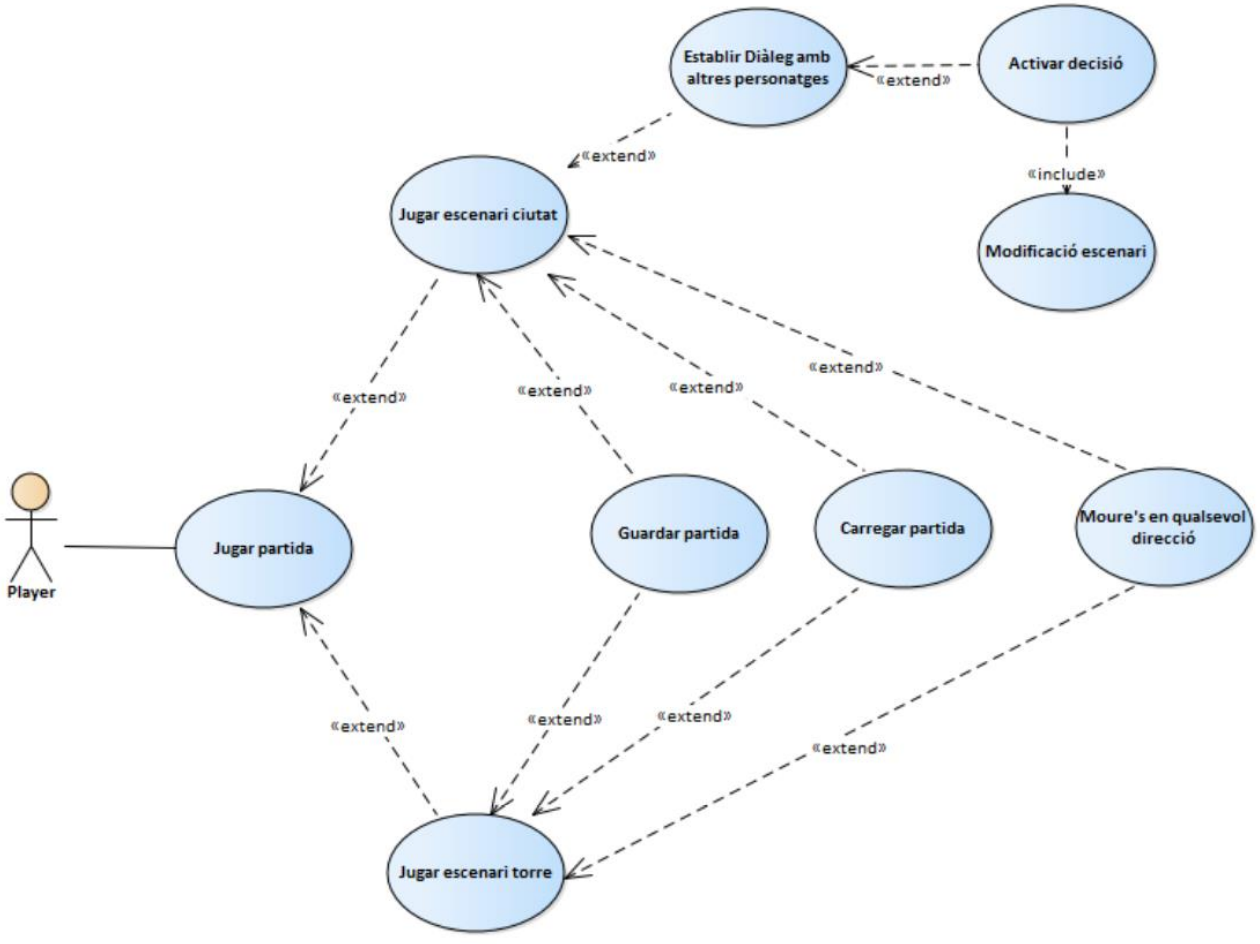
[13] Alan Zucconi, "Surface shaders in Unity" [en línia]. [Últim accés: 3 d'abril de 2019]. Disponible a: http://www.alanzucconi.com/2015/06/17/surface-shaders-in-unity3d/

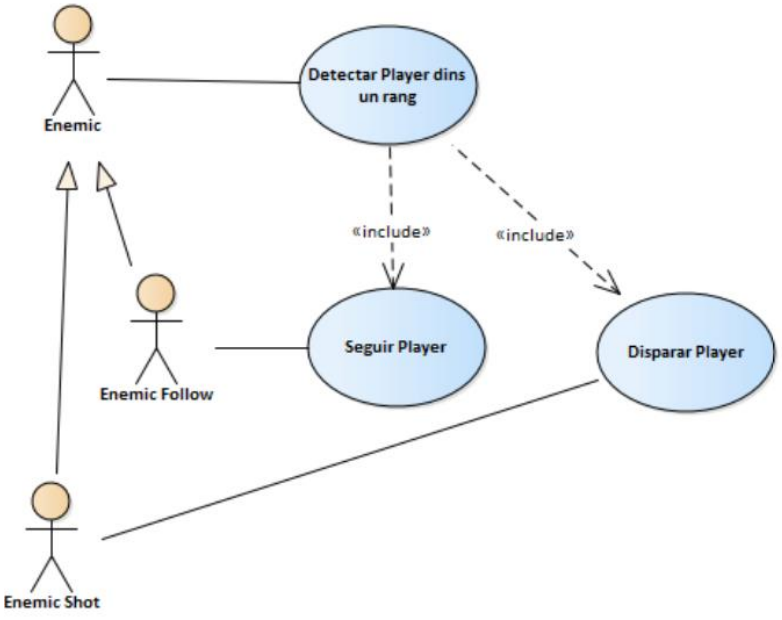
APÈNDIX

A1. PLANIFICACIÓ

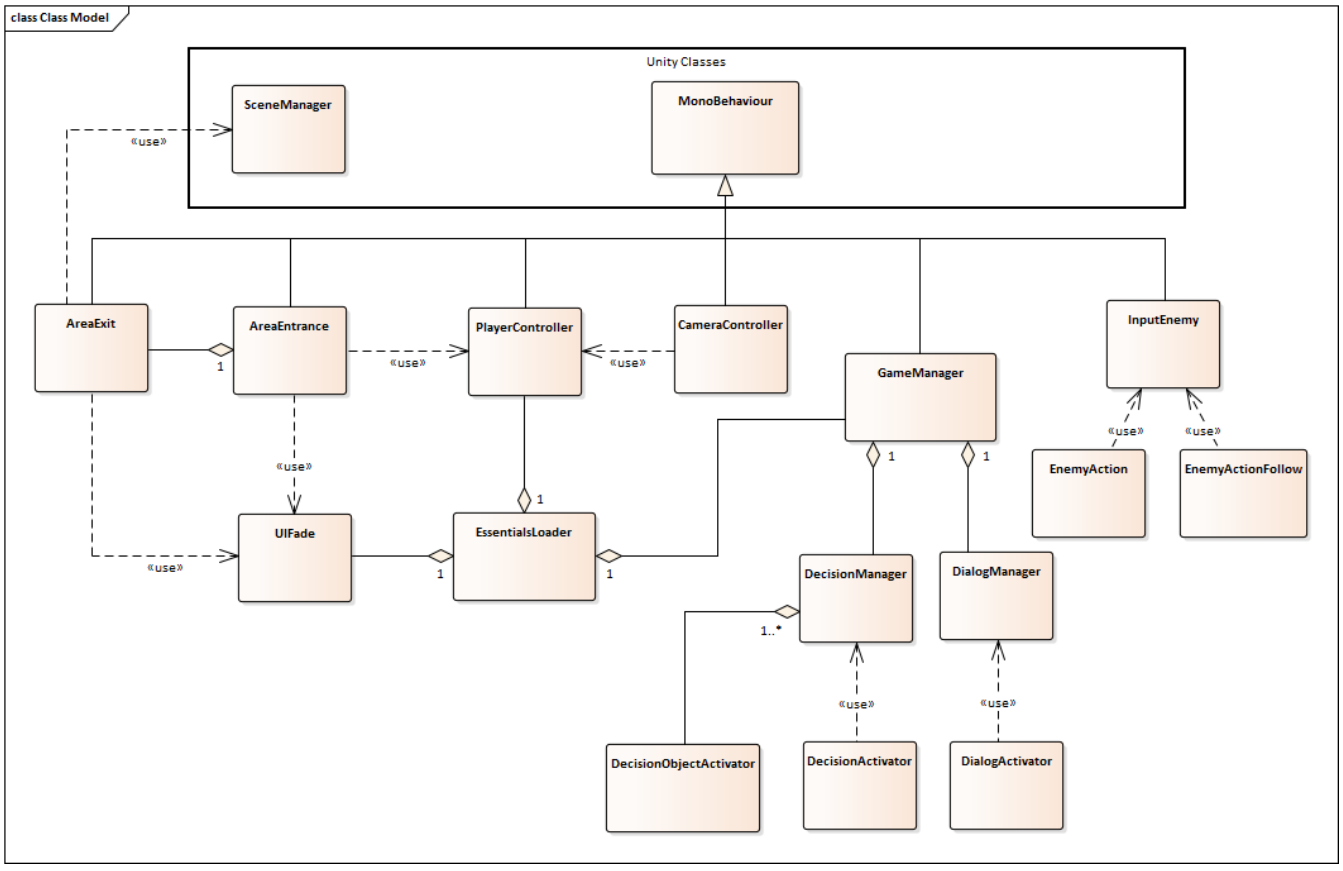


A2. DIAGRAMES DE CASOS D’ÚS





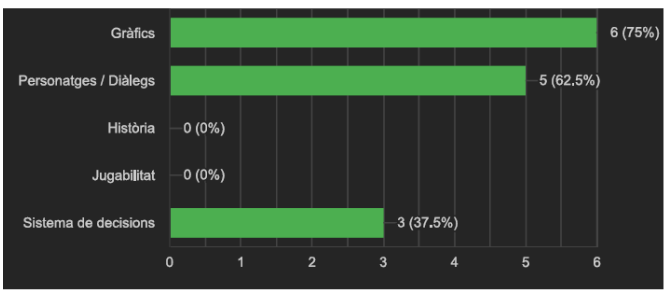
A3. DIAGRAMA DE CLASSES



A4. RESULTATS *USER TESTING*

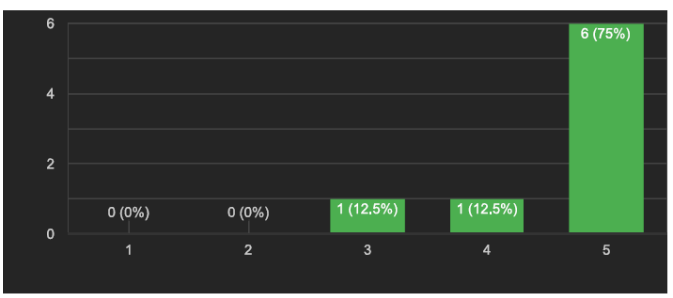
Que destacaries del joc?

8 responses



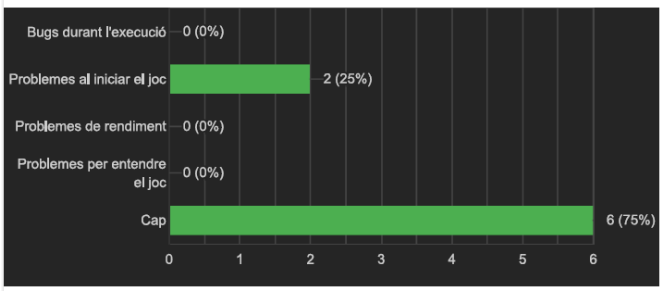
Com valores la millora gràfica amb shaders?

8 responses



Quins problemes has tingut?

8 responses



Com valores el sistema de decisions?

8 responses