Universitat Autònoma de Barcelona

escola d'enginyeria

Enginyeria Informàtica

Memòria del Projecte de Fi de Carrera:

6332-1 Videojoc Tower Defense 3D

Realitzat per **Andreu González Llinàs**

Dirigit per **Enric Martí Gòdia**

Bellaterra, Gener 2016

Universitat Autònoma de Barcelona

escola d'enginyeria

El sotasignat, ENRIC MARTÍ GOMIA

professor/a de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en/na ANDREU GONZÁLEZ LLIÑÁS

I per a que consti firma la present.

Signat: .....

Bellaterra, 23 de GENER de 2016

Català

En aquest projecte es presenta el disseny, la planificació i el desenvolupament d'un videojoc del gènere *Tower Defense* per a plataformes de PC utilitzant el motor de joc Unity. El joc implementa un sistema d'ajustament dinàmic de la dificultat utilitzant algorismes d'intel·ligència computacional, així com l'ús del *LEAP Controller* com a dispositiu d'entrada. S'exposa tot el procés de creació del videojoc: el disseny, l'art desenvolupat i el *gameplay*.

Castellano

En este proyecto se presenta el diseño, la planificación y el desarrollo de un videojuego del género *Tower Defense* para plataformas PC utilizando el motor de juego Unity. El juego implementa un sistema de ajuste dinámico de la dificultad utilizando algorismos de inteligencia computacional y utiliza el *LEAP Controller* como dispositivo de entrada. Se expone todo el proceso de creación del videojuego: el diseño, el arte y el *gameplay*.

English

In this report we present the design, the planification and the development involved to realize a *Tower Defense* game for PC using the Unity game engine. The game implements a dynamic difficulty adjustment using Computational Intelligence algorithms and the use of the *LEAP Controller* as the input interface. We expose all the proces involved in the creation of the videogame: the design, the art and the gameplay.

Índex

1 Introducció

Els videojocs són aplicacions que ens entretenen explicant una història, ens fan interactuar amb ella, alguns ens intenten ensenyar i d'altres divertir-nos, però tots ens permeten viure experiències úniques, evadint-nos de la realitat i immergint-nos en mons completament diferents. Per aquesta gran capacitat que tenen, ja des dels seus inicis, han fascinat a la gent.

Malgrat totes les diferències entre videojocs, el que podem assegurar és que aquests són aplicacions molt complexes que combinen diferents àmbits, ja siguin de la informàtica, la literatura, el disseny gràfic, la música, etc. Aquest aspecte interdisciplinari sempre m'ha fascinat, sobretot pel repte de fer que tot funcioni de manera fluida i interactiva. La qualitat dels videojocs sempre ha vingut relacionada amb el *hardware* disponible i, tot i així, hi ha hagut prou enginy per aconseguir efectes amb molt baix cost computacional.

Al llarg de la carrera he realitzat diferents cursos sobre videojocs, aprenent els seus secrets, tècniques i proporcionat-me la visió que hi ha darrera els jugadors, la del creador d'un videojoc. En altres cursos com el de *Tecnologies per a Jocs* vaig veure la importància del *Story telling*, dels joc *seriosos*, de les diferents maneres d'interactuar i d'analitzar jocs. A més, un dels projectes més satisfactoris que he realitzat a la carrera va ser el projecte de *Gràfics per Computador*, on vaig tenir l'oportunitat de formar part d'un equip per fer un joc des de zero i va ser una experiència molt enriquidora, ja que cada un dels membres teníem antecedents molt diferents.

Quan va arribar l'hora de fer el projecte de final de carrera vaig adonar-me que volia fer un videojoc. Ara bé, la decisió del gènere i disseny del joc va costar-me molt ja que sempre m'ha agradat el gènere d'aventura gràfica, pel seu caràcter 2D i perquè un dels seus elements principals és una bona narrativa, però aquestes característiques no s'ajustaven al projecte. Després de donar-hi moltes voltes, vaig pensar en un joc 3D del gènere *Tower Defense* que contingui elements innovadors pel gènere i que a la vegada incorpori diferents àmbits de la informàtica i noves tecnologies.

El gènere *Tower Defense* consisteix, de manera breu, en eliminar els enemics que intenten creuar el mapa mitjançant la construcció de torres que els disparen automàticament quan els enemics passen dins del seu abast. Els enemics i les torres tenen diferents característiques, i aquestes defineixen els diferents atributs: la velocitat a la qual es desplacen, la vida, la potència dels trets, el rang de tir, etc. El jugador, a mesura que derrota enemics, va guanyant punts i recursos per poder construir més torres o millorar-les. En el capítol 2 detallarem les característiques d'aquest gènere.

Amb tot això, els **objectius del projecte** són dissenyar, implementar i validar un videojoc del gènere *Tower Defense* en 3D amb les següents característiques:

- El jugador interactua sobre un mapa format per cubs que podrà rotar per orientar les torres.
- Conté un ajustament dinàmic de la dificultat, utilitzant algorismes d'intel·ligència computacional, que afegeix interactivitat i longevitat al joc.
- La càmera serà en tercera persona.
- El videojoc es desenvoluparà per a entorns de PC (Windows i Mac).

1.1 Eines de desenvolupament

Abans de la realització del projecte s'ha realitzat una cerca de les eines utilitzades en el món dels videojocs, per poder decidir així les que s'utilitzaran en el projecte. Es divideix la cerca en tres blocs: els motors de joc per al desenvolupament del *gameplay*, les eines per la creació de l'art i els dispositius d'entrada.

1.1.1 Motors de joc

El motor de joc (*game engine*) són un conjunt d'eines o *framework* que implementa les funcionalitats i requisits per a desenvolupar els *gameplay* dels videojocs, interactuant amb el *hardware* de la manera més eficient possible per obtenir el màxim rendiment.

Per a desenvolupar qualsevol videojoc és necessari disposar d'un motor de joc, aquest pot ser creat específicament per al videojoc o adaptar un motor ja existent. El motor de joc permet la programació de les diferents funcionalitats que necessita el videojoc, destacant les funcionalitats més importants. Un exemple podria ser el joc *Sniper Elite*

[37], on el motor de joc està dissenyat per a calcular de la manera més realista la trajectòria de la bala. En canvi, altres jocs de tirs fan una aproximació de les trajectòries per donar importància a tàctiques d'equip o altres funcionalitats.

Encara que els diferents videojocs tinguin els seus propis requisits, la quantitat de funcionalitats que podem exigir a un motor de joc venen restringides pel *hardware*, ja que per a que un joc sigui *"jugable"* ha de complir-se la interactivitat entre el joc i l'usuari. Un videojoc ha de respondre en concordança a les interaccions de l'usuari, sense que aquest percebi una falta de fluïdesa que disminuirà la capacitat d'immersió en el joc.

Existeixen multitud de motors de joc, més o menys potents, especialitzats en diferents plataformes o dispositius, ordinadors, videoconsoles, *smartphones* o multi-plataforma, permetent generalitzar gran part del codi i especialitzar la HUI [1]. Seguidament veurem els components d'un motor de joc i alguns dels que hi ha en el mercat actualment.

a) Components d'un motor de joc

Els motors de joc són una combinació d'eines per crear un videojoc i pràcticament tots ofereixen aquests cinc components:

- **Motor gràfic.** El motor gràfic proporciona la capacitat de carregar i renderitzar gràfics, ja siguin 2D o 3D, així com aplicar textures, il·luminació i gestió dels models. Els motors 3D també ofereixen suport en 2D per a poder gestionar els gràfics de la interfície d'usuari o GUI. Alguns motors també suporten l'ús de *shaders*, programes creats pel desenvolupador de joc, que donen llibertat per a treballar el càlcul de la il·luminació, així com generar efectes personalitzats als processadors gràfics (GPU). La GPU ens permet treure el càlcul dels elements gràfics de la CPU i, actualment, les plaques gràfiques disposen de molts de *cores* que ens permeten realitzar el càlcul de manera paral·lela.

- **Motor de físiques.** El motor de físiques s'encarrega de simular principalment fenòmens físics, col·lisions entre objectes i sistemes de partícules com aigua, foc, vent, etc. Treballen amb una representació simplificada dels models, per facilitar els càlculs. Alguns motors de físiques també s'utilitzen per simular moviments realistes de roba, els cabells, etc.

[1] HUI: Human-Computer Interaction

- **Musica i efectes sonors.** El sistema de so s'encarrega de carregar i reproduir fitxers de so amb sintonia amb els requeriments del videojoc. Així mateix, alguns motors també ofereixen la capacitat d'esmorteir el so en funció de la distància o posicionar el so dins l'espai 3D respecte del jugador, creant així realisme en el joc (l'anomenat So 3D).

- **Xarxa.** Gràcies a la millora de les connexions d'Internet s'han popularitzat els jocs multi-jugadors. El sistema de xarxa ens permet una abstracció dels protocols, facilitant el desenvolupament de videojocs multi-jugador.

- **Motor de *Scripting*.** El motor de *scripting* ens permet codificar accions del joc, de manera que el joc les interpreta al moment d'executar-se. Aquest fet ens permet realitzar modificacions sense haver de tornar a compilar tot el programa.

b) Motors de joc al mercat

Actualment existeix una gran varietat de motors de joc, alguns comercials i d'altres gratuïts. Seguidament es presenten tres motors de joc gratuïts més rellevant al mercat: Unity3D, CryEngine i Unreal Engine, així com els seus punts forts.

- **Unity3D** [22] És un motor de joc desenvolupat per *Unity Technologies*, inicialment pensat només per OS X però que actualment suporta els principals sistemes operatius (Windows, OS X i Linux), així com dispositius mòbils i videoconsoles.

  La versió actual Unity 5 proporciona les següents funcionalitats:

  - Un editor amb moltes funcionalitats, en la figura 1.1 podem observar la interfície gràfica del motor.
  - Motor gràfic amb suport per a 3D i 2D, així com funcions pel tractament de la GUI.
  - Un motor de físiques amb detecció de col·lisions, sistemes de partícules i suport per animació esqueletal.
  - Sistema d'il·luminació i personalització de *shaders*.
  - Un sistema de so en 3D, així com mesclador d'àudio.
  - *Scripting* amb C#, Unity Script i Boo.

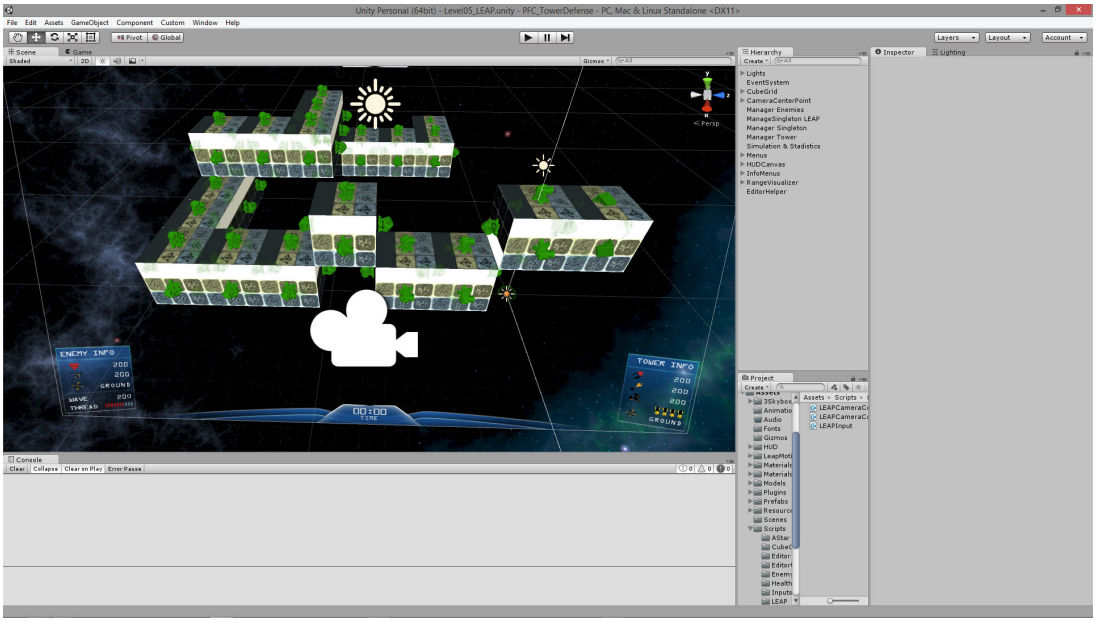


Figura 1.1: *Interfície d'usuari de l'editor Unity3D.*

– Un sistema de xarxa.

Unity treballa amb un sistema orientat a components, cada objecte esta format per diferents component que defineixen com és i les funcionalitats que tindrà l'objecte. Tots els objectes d'una escena tenen la component bàsica de transformació que defineix la seva posició, rotació i escalat dins l'escena. Podem afegir altres components que contenen un script, components de motor de físiques com *colliders*, etc.

Alguns dels jocs que s'han desenvolupat utilitzant Unity3D són *Superhot* [21], *Kerbal Space Program* [32] i *Besige* [19], entre molts d'altres.

- **CryEngine** [5] És un motor desenvolupat per la companyia alemanya *Crytek*, per tots els seus videojocs des de *Far Cry*. Actualment suporta Windows, Linux, OS X, PlayStation 3 i 4, Wii U, Xbox 360 i One, iOS i Android.

  Algunes de les funcionalitats d'aquest motor són:

  – Un editor enfocat al WYSIWYG [2] que proporciona *feedback* directe amb el producte final, el la figura 1.2 podem veure l'entorn de treball.

  – Un motor gràfic d'última generació per a 3D amb *shaders* enfocats al realisme i efecte cinematogràfic.

[2]WYSIWYH: *What you see is what you get*: Tipus d'interfície d'usuari basat en el que veus és el que obtens.

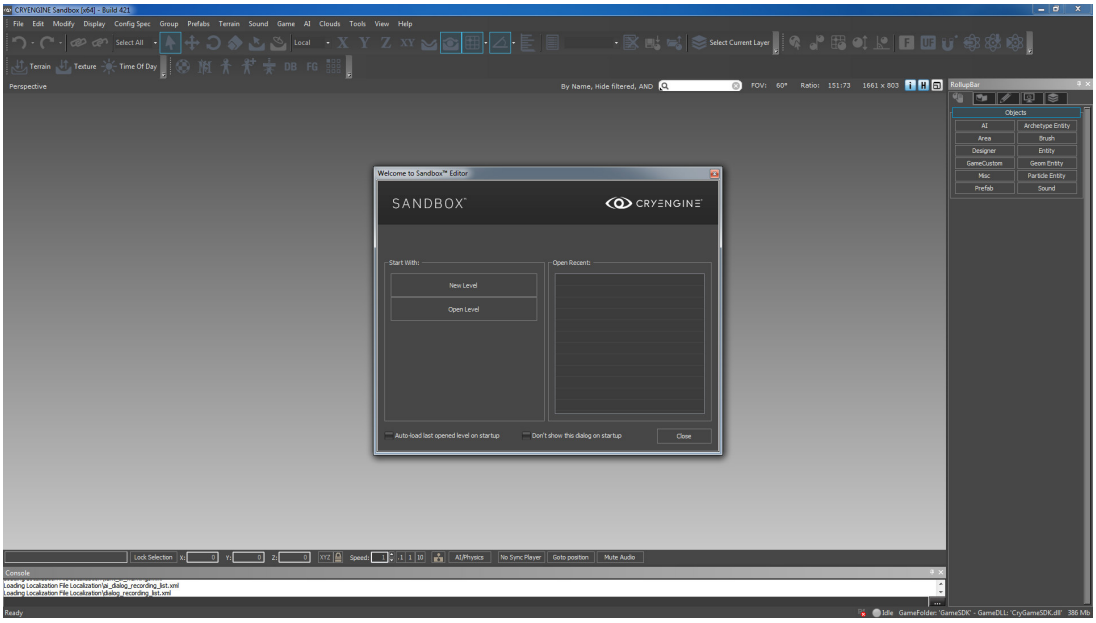


Figura 1.2: *Interfície d'usuari de l'editor CryEngine.*

- Un motor de físiques *multi-threading* integrat amb detecció de col·lisions, sistemes de partícules, animació esqueletal i objectes deformables.
- Sistema d'il·luminació amb funcionalitats de HDR [3], permetent més realisme a l'hora de calcular la il·luminació, incloent ombres dinàmiques, així com la personalització de *shaders*.
- Un sistema de so en 3D, així com mesclador d'àudio.
- *Scripting* amb Lua, llenguatge de programació imperatiu i estructurat.
- No incorpora un sistema de xarxa integrat.
- Funcionalitats per l'anàlisi del rendiment del videojoc de manera molt detallada, permetent veure estadístiques sobre l'ús de la memòria, CPU, GPU, temps en el renderitzat, sistemes de partícules, IA, etc.

El motor CryEngine és un motor d'alta qualitat i així ho demostren els títols que s'han produït amb ell, ja que veient les seves capacitats està pensat per aconseguir molt realisme. Alguns més coneguts són tots els títols *Far Cry* [31] i la saga *Crysis* [30].

[3] HDR: *High dynamic range.*

- **Unreal Engine** [7] És un motor desenvolupat per la companyia *Epic Games*, inicialment desenvolupat per a jocs de tirs en primera persona i que actualment es pot utilitzar per altres gèneres. Actualment suporta Windows, Linux, OS X, PlayStation 3 i 4, Wii U, Xbox 360 i One, iOS i Android entre d'altres dispositius.

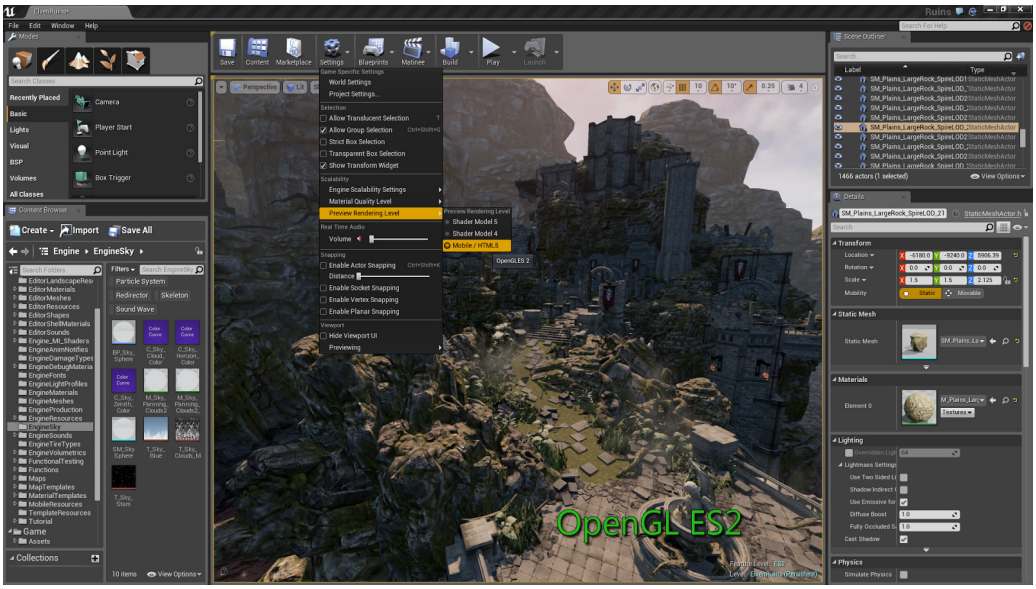


Figura 1.3: *Exemple d'interfície d'usuari de l'editor Unreal Engine.*

Algunes de les funcionalitats d'aquest motor són:

- Un editor que integra totes les funcionalitats des de creació de materials, modificació d'animacions, etc. En la figura 1.3 podem veure la interfície de l'editor.
- Un motor gràfic per a 3D amb funcionalitats per edició de la GUI.
- Un motor de físiques integrat amb detecció de col·lisions, sistemes de partícules, animació esqueletal i objectes deformables.
- Un sistema de so en 3D.
- *Scripting* mitjançant un llenguatge propi *UnrealScript* i, mitjançant *Blueprint Visual Scripting*, una representació gràfica de caixes representant els components i les arestes que els connecten. Aquest sistema també inclou un sistema visual per debugar els comportaments.
- Sistema de xarxa integrat.

Unreal Engine s'ha utilitzat en molts de jocs, principalment per a *first person shooters* com la serie *Unreal*, la franquícia *Bioshock* [28] i *Batman: Arkham City* [27] entre molts d'altres.

1.1.2 Art

A banda de la programació del joc, una gran part en el procés de creació d'un videojoc és l'elaboració d'elements gràfics, ja que cada vegada aquests són més demandats i, gràcies a l'avanç de les targetes gràfiques, cada cop tenim uns gràfics més realistes.

En aquest apartat ens centrarem en anomenar quins són els tipus de continguts que els artistes generen per a un videojoc en 3D i les eines que es solen utilitzar. Principalment, els artistes s'encarreguen de la creació dels models, la texturització, l'animació dels models, la il·luminació, la generació de cinemàtiques i vídeos que ajuden a explicar la història del joc o a entretenir l'usuari durant la càrrega del següent nivell.

Per la creació de models s'utilitzen programes de modelatge com el *Autodesk 3Ds Max* [1], *Blender* [4] o *Cinema 4D* [15]. Aquests programes permeten, a partir de primitives, esculpir o extruir les primitives per crear els models. Permeten treballar a diferents nivells de l'objecte, des de vèrtexs, arestes, cares, amb les normals, entre d'altres.

Una vegada tenim els models creats podem passar a generar un esquelet per aquest i animar-lo. El procés d'animar consisteix en generar moviments entre dos instants de temps, usualment creant punts claus (*keypoints*) on crearem un moviment respecte l'anterior i interpolarem entre aquests. La tècnica més utilitzada és l'animació esqueletal [36], que ens permet crear un sistema d'ossos que tenen una jerarquia ja que cada os l'associem a una part del nostre model mitjançant uns pesos. L'animació s'aconsegueix movent els ossos, que arrosseguen la malla en funció de la jerarquia i pesos. Això ens permet animar un sistema d'ossos i utilitzar-ho en diferents models, obtenint així animacions per a models similars.

Finalment podem passar a texturitzar el model per a donar-li color a la pell, definir vestimenta, etc. El procés de texturització consisteix en crear una imatge 2D (textura) i, per això, s'utilitzen eines de dibuix en 2D com poden ser el *Adobe Photoshop* o *GIMP*. La textura generada és mapejada al model mitjançant les coordenades de textura [39].

1.1.3 LEAP Motion Controller

Des de sempre utilitzem dispositius d'entrada per interactuar amb els dispositius, com el teclat, el ratolí o el comandament d'alguna videoconsola. En l'actualitat, s'estan reinventant aquests dispositius canviant la manera com interactuem amb els dispositius del nostre entorn afegint més naturalitat a l'hora d'interactuar amb els videojocs. Alguns exemples són les pantalles tàctils, el *Wii Remote* [42], el *Kinect* [33], *Steam Controller* [20] o el *LEAP Motion Controller* [12]. Destaquem el *LEAP* ja que és el que s'utilitza en el projecte.

El *LEAP Motion Controller* va aparèixer al voltant del 2013 i és un dispositiu *hardware* preparat per reconèixer els moviments de les mans i els dits, sense cap tipus de contacte. El controlador consta de dues càmeres infraroges i tres díodes LED infrarojos, que permeten crear imatges de l'espai que envolta el controlador i que són enviades a l'ordinador. Aquest compara les diferents imatges i extreu els moviments.

El dispositiu reconeix la posició i gest de les mans i està pensat per dues configuracions, com es veu a la figura 1.4: sobre la taula o sobre un casc de realitat virtual. Sobre la taula ens permet utilitzar-lo com un ratolí amb més funcionalitats que un de normal, i davant un casc de realitat virtual ens permet introduir de manera natural les mans en els nostres entorns virtuals.

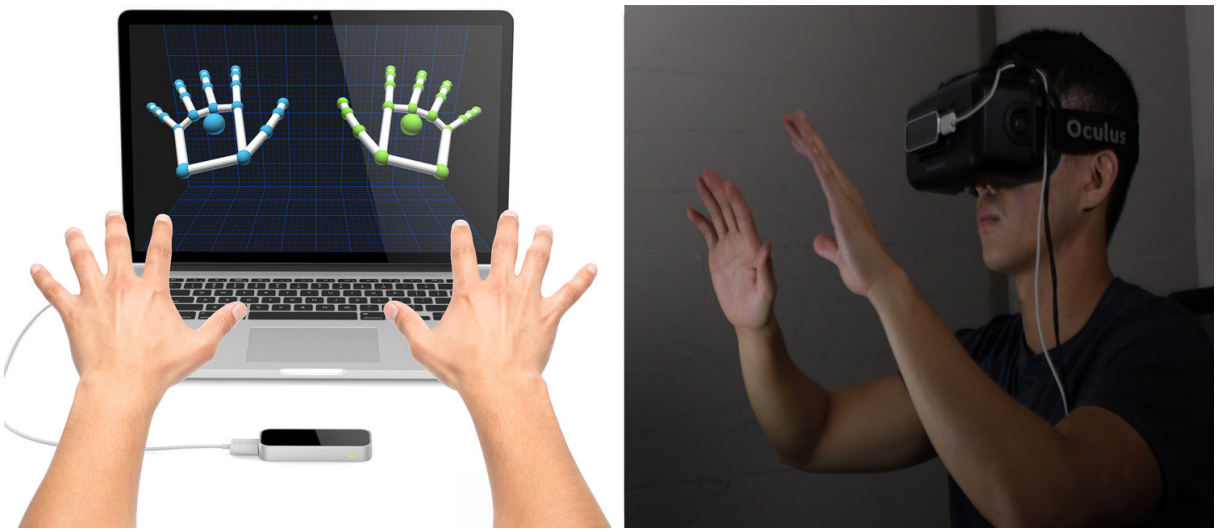


Figura 1.4: *LEAP Motion Controller diferents configuracions.*

LEAP ens ofereix diferents *SDK* depenent de la plataforma en la que es vulgui desenvolupar. Aquestes poden ser *Unreal Engine*, *C++*, *Unity3D*, *Javascript*, *Phyton* i *Objective C*, entre les més destacades. El *SDK* ens ofereix les eines bàsiques per comunicar-nos amb el dispositiu, així com un conjunt de classes que ens permeten reconèixer les mans, els dits, els gestos i, fins i tot, objectes com ara un llapis.

1.1.4 Eines per al projecte

Després d'analitzar diferents eines, per la realització del projecte s'ha elegit el motor de joc Unity3D. Unity ens ofereix funcions més bàsiques que es tradueixen en una utilització més senzilla, fent així la creació dels jocs més ràpida. Aquest fet ha afavorit la popularització d'aquest motor de joc entre la comunitat de desenvolupadors independents, que amb pocs recursos han pogut destacar amb els seus videojocs. Finalment, cal destacar que Unity ofereix una corba d'aprenentatge molt ràpida gràcies a la gran documentació i els tutorials amb exemples complets.

Per altra banda, es pretén realitzar tots els aspectes gràfics i artístics del joc i, per això, s'ha triat el programa de modelatge i animació *Autodesk 3Ds Max 2015* i *Adobe Photoshop* per texturització i generació de la GUI.

Finalment, s'ha introduït el LEAP Controller com a alternativa a la utilització de la combinació teclat-ratolí.

1.2 Planificació del projecte

Al principi del projecte es va realitzar una planificació de les diferents fases: recerca i aprenentatge, disseny, implementació i resultats. La figura B.1, situada a l'annex, conté el diagrama de Gantt de les diferents tasques que tot seguit comentarem.

La primera fase és la de recerca i aprenentatge i consisteix en dues setmanes d'aprenentatge bàsic de les diferents eines de desenvolupament. Amb aquests coneixements, i obtenint una idea de les seves capacitats, es procedeix a dissenyar el joc.

Després comença la segona fase, el desenvolupament del joc, que ocuparà la major part del temps i que també s'ha fraccionat per treballar de manera incremental. Com a primer pas es prepararà el terreny de joc, tot el sistema de cubs i les funcionalitats que requereix. Després es treballarà en les funcionalitats bàsiques de les torres i els enemics, la seva col·locació i el moviment dels enemics. Un cop es tenen les funcionalitats bàsiques es treballarà amb la interfície de l'usuari i la interacció teclat-ratolí amb el jugador. Amb la interacció teclat-ratolí realitzada, afegim el suport per a la utilització del dispositiu *LEAP Controller*. Amb la primera versió del joc crearem els menús de joc. Finalment, millorarem les torres i els enemics afegint múltiples tipus.

Arribats a aquest punt ja tenim el joc creat i es passarà a la tercera fase: realitzar testeig. Amb els resultats es procedirà a realitzar modificacions i millores. Finalment, un cop acabada l'aplicació, en la ultima fase: dedicarem el temps a acabar la memòria del projecte.

En aquesta memòria s'exposa tot el procés que s'ha dut a terme per dissenyar i implementar el videojoc del gènere *Tower Defense*. En el capítol 2 analitzarem el gènere *Tower Defense*: veurem en detall en què consisteix el gènere, els seus components i les estratègies utilitzades. En el capítol 3, es mostrarà el *game concept*, document que conté les idees bàsiques del disseny del videojoc, el seu art, els elements que el formen i totes les funcionalitats que tindrà. En el capítol 4, veurem el procés de desenvolupament des del codi fins al *Gameplay*, i com s'han programat les diferents parts del videojoc. En el capítol 5, el dedicarem a mostrar els resultats, així com els problemes que han sorgit. Finalment en el capítol 6, s'exposen les conclusions de la realització del projecte i les millores. I en els annexes podem trobar els diagrames de casos d'us, els diagrames de classes i el diagrama de Gantt.

El gènere Tower Defense 2

El gènere *Tower Defense* és un subgènere del jocs d'estratègia focalitzat a l'administració de recursos per la creació i col·locació d'unitats. El gènere va aparèixer al 1990 amb el joc *Rampart* [34] creat per *Atari Games*. El joc consistia en sis nivells on el jugador defensava una fortalesa. Podem veure una captura del joc a la figura 2.1 amb la fortalesa a la dreta i els enemics (vaixells) a la esquerra.

Figura 2.1: *Captura de pantalla del joc Rampart.*

La partida es dividia en fases: inicialment el jugador creava torres en la fortalesa, després es passava a una ronda d'atac seguit per una fase de reparació i, si el jugador reparava tots els danys, podia crear torres per un temps limitat; finalment es tornava a la fase d'atac fins que es destruïen suficients vaixells o fins que el jugador fallés al reparar la seva fortalesa. La popularitat d'aquest joc l'ha portat a que es desenvolupin portabilitats a una gran varietat de dispositius més moderns com poden ser la *Game Boy*, la *PlayStation 2* i la *Xbox*, per anomenar-ne alguns, incorporant millores com més nivells i capacitats multi-jugador, entre d'altres.

Encara que *Rampart* és considerat el precursor del gènere i influencià mapes de jocs com el *StartCraft* [38], *Age of Empires II* [26] i *Warcraft III* [41], no és la definició del gènere tal com la coneixem actualment. Els desenvolupadors de jocs independents consolidaren definitivament el gènere desenvolupant jocs com el *Flash Element Tower Defense* [6] i *Desktop Tower Defense* [10].

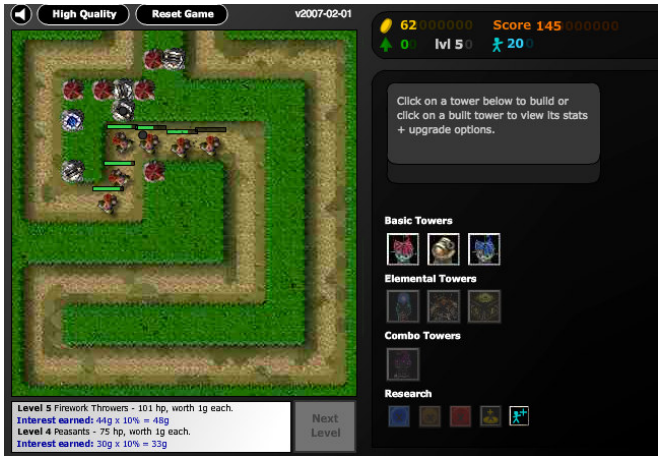


Figura 2.2: *Captura de pantalla del joc Flash Elements Tower Defense.*

El joc *Flash Element Tower Defense* va ser el primer videojoc dedicat al gènere i va aparèixer al gener de 2007 per a navegador web utilitzant tecnologia *Flash*. Tal com es mostra a la figura 2.2, els enemics (punts vermells), que sorgeixen en agrupacions, segueixen el camí i el jugador va situant torres al costat d'aquest. El jugador pot, en tot moment, construir, millorar torres i aconseguir millores per ser més efectiu contra els enemics.

Figura 2.3: *Exemple de partida del joc Desktop Tower Defense.*

Al març del mateix any també va sortir el videojoc *Desktop Tower Defense* [10], també per a navegador utilitzant tecnologia *Flash*. Aquest va adquirir una gran popularitat, guanyant així el premi del Festival de Jocs Independents (IGF, per les seves sigles en anglès), eclipsant la popularitat del *Flash Element Tower Defense*. Les partides, com podem observar a la figura 2.3, consistien en evitar que els enemics creuessin el mapa

representat per un escriptori i, per això, una de les estratègies era crear un camí el més llarg possible utilitzant les torres. D'aquesta manera, s'aconsegueix que els enemics passin al costat del màxim de torres possibles i ser derrotats.

Un altre exemple de videojoc del gènere que va aparèixer al 2009 i va tenir una gran acollida del públic és *Plants vs. Zombies*, un joc d'un jugador multi-plataforma en el que s'ha de defensar una casa de l'atac zombi.

Figura 2.4: *Captura de pantalla del joc Plants vs. Zombies.*

Com podem observar en la figura 2.4, els zombis apareixen de la dreta i el jugador defensa la part esquerra. EL jugador disposa de diferents tipus de plantes que ha d'anar col·locant per aconseguir recursos (llum solar), atacar i bloquejar el zombis. Si els zombis arriben a alguna de les plantes aquests comencen a menjar-se-les. També podem observar diferents tipus de zombis, que varien en les seves característiques velocitat, més resistència als cops, etc.

2.1 Els elements del Tower Defense

Ara que ja hem vist alguns dels jocs del gènere *Tower Defense*, analitzarem els seus elements. De l'apartat anterior podem extreure la següent definició [40]:

> El *Tower Defense* és un subgènere de l'estratègia a temps real, on el seu objectiu és defensar o evitar que els enemics arribin una posició especifica del mapa, mitjançant el posicionament de torres les qual disparen als enemics quan passen. Els enemics i les torres normalment tenen diferents habilitats, costos i millores.

Tot i que la definició anterior és prou encertada, el gènere ha estat reinventant-se constantment al llarg dels anys. Una altra manera de definir el gènere seria analitzant els diferents elements que típicament trobem dins els seus videojocs.

- **El terreny:** El mapa de joc defineix on el jugador pot situar les torres i per on poden anar els enemics, així com també l'origen dels enemics i la posició a defensar pel jugador. Els exemples de l'apartat anterior representen les topologies de terrenys més utilitzades: podem observar topologies lliures (figura 2.3), amb un camí definit per l'enemic (figura 2.2), o amb camins paral·lels (figura 2.4). Les diferents possibilitats de terreny generen diferents estratègies pel jugador, així com també diferents possibilitats de control de la dificultat del joc.

  Examinant el joc *Flash Element Tower Defense* i el seu estil de terreny (figura 2.2), podem identificar diferents punts on la col·locació de les torres farà que els enemics que passin puguin ser atacats per més recorregut de camí (punts calents), com col·locar una torre a la part interna d'una cantonada. En canvi, hi ha altres llocs del camí on la col·locació d'una torre només afectarà als enemics en un recorregut petit, com al costat d'una recta. Podem veure que el camí i la seva forma tenen un gran impacte en el control de la dificultat, essent els mapes lineals els més fàcils. Podem tenir mapes més difícils restringint llocs on poder construir, reduint "punts calents" o permetent que el camí tingui bifurcacions on l'enemic decideix quin camí elegir aleatòriament.

  Altres formes de controlar la dificultat són la limitació del nombre de torres que es poden construir i si aquestes poden ser destruïdes.

- **Les torres:** La construcció de torres requereix estratègia per part del jugador en la col·locació i una administració dels recursos dels que disposa. Normalment el jugador disposa d'una varietat de torres de diferents característiques i costos, com ara quin tipus d'enemic poden atacar, la capacitat de fer mal, la velocitat del tret, el rang del tret, etc. Aquests atributs poden millorar-se.

  Diferents jocs proporcionen diferents tipus de torres. Alguns inclouen funcionalitats com la capacitat d'atacar a l'últim enemic que es troba dins el seu rang de tir o al que es troba al mig, donant més opcions que al primer que ha entrat dins el seu rang. Altres afegeixen efectes com la capacitat d'enviar un enemic al principi, detectar enemics ocults, etc.

- **Els enemics:** L'elecció dels enemics i les seves capacitats ha d'anar directament relacionada amb les torres i, per tant, els enemics són del mateix tipus que les torres, tenint diferents atributs com la seva velocitat, punts de vida i armadura, entre els més comuns.

  Normalment els enemics apareixen cada cert temps, encara que el jugador pot forçar la sortida d'aquests. Els enemics apareixen en agrupacions o onades formades per enemics del mateix tipus o, en casos especials, un únic enemic més fort. Cada cop que arriba un enemic al final del camí, aquest resta vides al jugador.

- **El sistema de recompenses:** Existeixen dos sistemes de recompenses: un durant la partida i un de tot el videojoc. El sistema durant la partida consisteix en acumular punts per matar enemics, així com obtenir recursos per construir o millorar les torres. El segon sistema intenta incrementar la longevitat del joc introduint millores com a jugador, que després es podran utilitzar dins la partida, o creant reptes que el jugador ha de superar per obtenir una medalla o registrant el mèrit. Aquests sistemes incrementen l'interès del jugador per aconseguir-los i millorar-los, fent que jugui repetidament els nivells.

- **Un jugador o multi-jugador:** Principalment, els jocs del gènere són d'un jugador i l'ordinador de joc va enviant enemics cap al jugador. Tot i així, hi ha videojocs del gènere que tenen opcions de multi-jugador, com la de jugar cooperativament per eliminar els enemics i competir amb el nombre de punts obtingut cada un.

2.2 Les estratègies

Les estratègies en els jocs de *Tower Defense* varien segons els elements que el formen. Diferents enemics requereixen estratègies diferents i, per tant, el jugador ha de col·locar torres en funció del enemics que apareixen. És important, en un *Tower Defense*, l'existència d'un balanç que incrementi la dificultat de manera progressiva, requerint que el jugador administri els seus recursos per millorar torres, així com la seva col·locació.

Es per això que podem resumir les estratègies en dues problemàtiques: l'administració dels recursos i la col·locació de les torres. L'administració de recursos consisteix en decidir quan comprar una torre. En aquesta problemàtica les dues estratègies principals son comprar moltes torres barates i les seves millores o guardar per comprar torres més fortes i millores més cares. Pel que fa a la problemàtica de la col·locació de les torres depèn del mapa i la seva topologia, essent una bona estratègia reservar bones posicions per torres més fortes i situar al principi del camí una bona defensa.

El dissenyador del joc ha d'aconseguir un balanç, entre la quantitat de recorregut i el nombre de torres que es poden col·locar, restringint posicions per evitar acumulacions de torres, i entre els enemics i les torres. L'existència d'un balanç és prou important, ja que un mal balanç pot afectar l'experiència del jugador, fent que aquest ho trobi massa fàcil o massa difícil i causant que abandoni el joc.

Game Concept

3

El projecte consisteix en el disseny d'un videojoc del gènere *Tower Defense* en 3D. El joc serà en tercera persona, on el jugador interactua sobre un mapa format per cubs i col·loca torres de defensa per atacar els enemics, podent rotar els cubs per tal d'orientar les torres i protegir la seva base. A continuació es detalla el disseny del joc: primer veurem els diferents elements artístics que formen l'art, descriurem els components del joc, el sistema de joc i les seves mecàniques, la interacció amb l'usuari, l'ajustament dinàmic de la dificultat i, finalment, una cerca de jocs semblats al disseny.

3.1 L'art

El disseny de l'art consisteix en detectar els diferents factors que restringeixen les possibilitats de disseny o, millor dit, en obtenir les guies de com realitzar els diferents elements per obtenir una uniformitat en el joc, l'elecció de colors, les textures, l'estructuració dels menús, la informació a mostrar, etc.

En el nostre cas, el disseny de l'art té una estètica de l'espai i, per aquest motiu, la resta dels elements s'inspiren en formes i colors relacionats amb la temàtica. El disseny de l'art s'ha separat en dues parts: la interfície d'usuari i el modelatge dels objectes 3D.

La interfície d'usuari ha estat dissenyada pensant en la interacció del *LEAP Controller*, intentant crear menús de forma circular i fent que apareguin al voltant de la posició del cursor, reduint així la distància de moviment que ha de realitzar el jugador. A la figura 3.1 podem veure la interfície d'usuari, així com els dos menús: el de construcció i el de millora o venta de la torre.

El disseny dels models 3D segueixen la temàtica espacial, com podem observar a la figura 3.2: els enemics *(a)* sorgeixen d'un ou alienígena *(c)*, les torres *(b)* tenen un caràcter industrial i el model del jugador *(d)* sembla una estació de comunicacions.

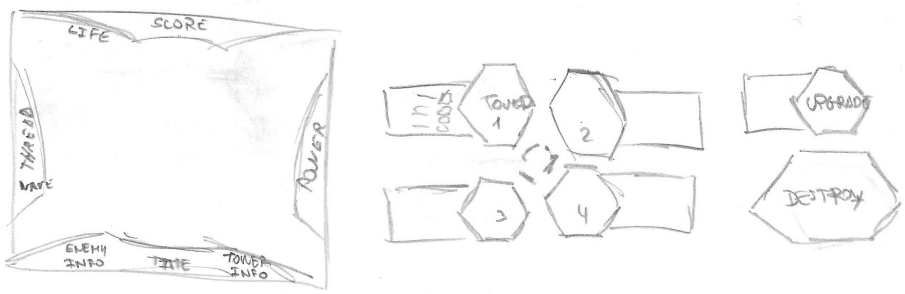


Figura 3.1: *Esborrany de la GUI, el menú de construcció i el menú de millora o venta.*

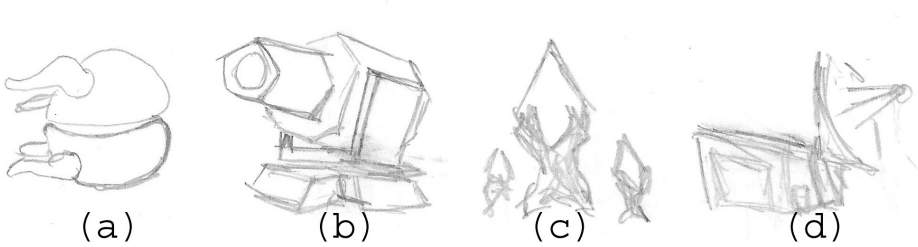


Figura 3.2: *Esborrany dels diferents models que formaran el joc.*

3.2 Els components del joc

1. **El terreny:** Un dels grans punts que diferencien el videojoc del projecte de la resta és la topologia del mapa. Aquesta estarà formada per cubs (figura 3.3), que formen una estructura de tres dimensions, connectant tots els elements del joc, i el camí el formaran les cares visibles d'aquests cubs. Per simplificar al màxim possible el terreny, els cubs només podran situar-se als eixos i, per tant, es connectaran en la totalitat de la seva cara sense deixar espai entre ells. Per tal de diferenciar l'origen dels enemics i la posició de defensa es crearan models que marquin aquests elements.

2. **Els enemics:** El camí definit pel terreny comporta bifurcacions i, per tant, cada agrupació d'enemics pot elegir per quines cares formem el seu camí, sempre i quan siguin amb connectivitat a quatre[1], pel que no poden canviar a cares que es troben en la diagonal.

   Per simplificar el videojoc es realitzaran dos tipus d'enemics: un d'aire i un terrestre. Els seus atributs defineixen un enemic, definint la quantitat de vida i la velocitat que es mouran. Aquests seran generats per l'ordinador amb algorismes evolutius que detallarem més endavant.

[1] La connectivitat a quatre només permet moviments en un únic eix a la vegada.

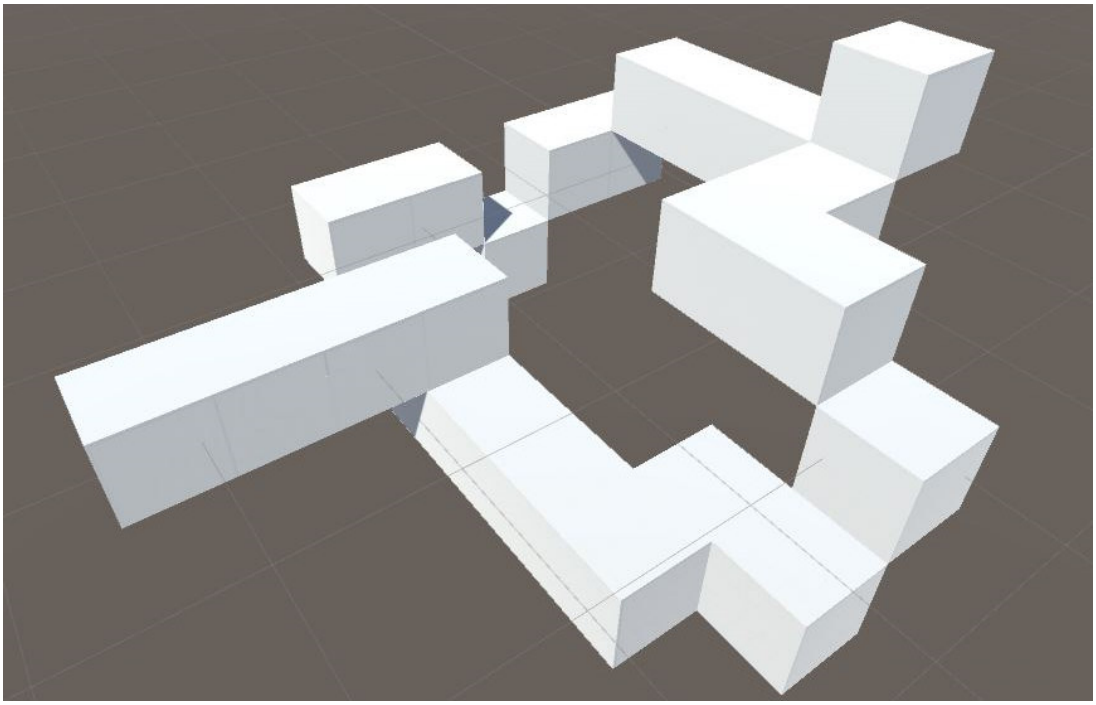


Figura 3.3: *Representació del terreny de joc.*

3. **Les torres:** Les torres es situaran sobre les cares visibles del cubs sense impedir el pas de les unitats enemigues. Les torres atacaran automàticament però només podran atacar en el pla de la cara en la que estan situades. És per això que els cubs que continguin torres podran ser rotats si existeix un eix de rotació. L'existència de l'eix de rotació d'un cub ve determinat pels seus cubs veïns[2] i és la recta resultant d'unir el centre del cub amb el del seus veïns si aquesta no es creua. Per tant, per a que un cub pugui rotar ha de tenir un o dos veïns i, si els té, aquests han d'estar alineats i, llavors, l'eix de rotació és l'eix que els uneix.

   Es crearan dos tipus de torres per poder atacar als dos tipus d'enemics. Els atributs que, com abans, defineixen les propietats de la torre també seran creats amb algorismes evolutius que explicarem més endavant. Una torre tindrà els atributs de rang de tir, quantitat de dany que realitza i velocitat de recàrrega per poder seguir disparant, així com el nivell de la torre. Les torres podran tenir 5 nivells, que milloraran els seus atributs. Les millores consistiran en incrementar els atributs per una factor constant.

[2]Es consideren veïns aquells que estan en contacte amb les cares i no els cubs que estan a les diagonals.

4. **El sistema de recompenses:** A nivell de partida, el jugador necessita energia per construir torres. Aquesta energia s'aconsegueix amb el temps i eliminant enemics. Eliminant enemics també aconseguim punts que s'acumularan fins a acabar la partida que, un cop acabada, es guardaran en un fitxer a l'ordinador. Això permetrà tenir un historial de puntuacions de les partides realitzades.

5. **Un jugador o multi-jugador:** El joc serà d'un únic jugador i no comptarà amb funcionalitats en xarxa.

3.3 Sistema de joc

Les partides del videojoc consistiran en col·locar torres i orientar-les rotant els cubs, eliminant el màxim d'enemics possibles. L'eliminació d'enemics permetrà al jugador obtenir punts, així com recursos per construir més torres. Ara bé, la partida acabarà si arriben enemics al final o bé el jugador es queda sense vida, o bé quan el temps de partida hagi finalitzat.

3.4 Interfície d'usuari GUI

En el videojoc tindrem diferents menús: el menú d'inici, el de sortida i els menús dins el joc. EL menú d'inici permetrà seleccionar el temps de joc i sortir, el de sortida sortir del joc. Els menús dins el joc (HUD [3]) mostraran: la vida del jugador, l'energia que disposa, la puntuació, el temps de partida restant, la informació sobre el tipus i els atributs dels enemics, així com una barra que mostra si existeixen o no enemics sobre el mapa. També mostrarem la informació de la torre quan situem el cursor per sobre d'ella. Altres menús apareixeran quan el jugador vulgui construir una torre, per poder seleccionar quina vol construir, o el menú per millorar o vendre la torre seleccionada. En l'annex hi ha les figures A.1 i A.2 on podem veure els casos d'ús que representen la interacció del jugador amb els menús explicats.

[3]HUD: Heat-up-display: Informació que en tot moment es mostra per pantalla.

Els controls que tindrà l'usuari per interactuar amb el joc, estan pensats principalment amb les possibilitats que pot oferir el *LEAP Controller*, intentant que siguin el més intuïtius possibles. Tot i així, també podrem utilitzar la combinació teclat-ratolí, utilitzant el ratolí principalment i detectant *clicks* com si els botons es mantenen pressionats i es mou el ratolí. El botó principal rotarà els cubs i el secundari obrirà el menú de construcció o millora-venta. Les opcions d'aquest menú es seleccionaran amb el botó principal. Per al moviment de rotació mantindrem el botó secundari pressionat i mourem el ratolí. El simple fet de moure el ratolí mourà la càmera per obtenir un millor control. Per tal d'aconseguir la mateixa funcionalitat amb el *LEAP*, quan aquest reconeix una única mà actuarà com el ratolí, reconeixent els gestos dels dits, i quan reconegui dues mans realitzarà el moviment de la càmera.

3.5 Intel·ligència evolutiva per ajustament dinàmic

En tots els videojocs, les experiències que es transmeten a cada jugador són diferents. El jugador pot trobar fàcil un nivell i avorrir-se ràpidament o frustrar-se si és massa complicat. Encara que molts jocs ofereixen l'opció de seleccionar el nivell de dificultat, l'experiència estàtica que aquests ofereixen pot no ser suficient. Existeix tot un camp de recerca en aplicar intel·ligència computacional (IC) per ajustar dinàmicament la dificultat al jugador. Prenent com a base l'article de *Avery et al. (2011)* [3] definim per al nostre joc una IC per a que gestioni els atributs dels enemics i les torres.

La IC utilitzarà algorismes evolutius. Aquests funcionen de la següent manera: es crea una població inicial d'individus diferenciats pels seus cromosomes, i que defineixen els individus i les seves característiques. Aquests individus es reprodueixen per generar una nova població que maximitza la funció *fitness*, funció que puntua el comportament dels individus. La selecció de quins individus es reprodueixen pot realitzar-se de diferents maneres. Tot i així, utilitzarem l'elitisme, que consisteix en reproduir els millors individus i descartar la resta.

Utilitzarem un paràmetre de puntuació que ajustarà de manera global el grau de força que poden aconseguir els enemics, permetent poder controlar la dificultat. El paràmetre s'incrementarà o es reduirà en un 5%, en funció del resultats del *fitness* que ha obtingut

l'agrupació d'enemics. A mesura que augmenta, permet que els enemics puguin ser més forts.

Abans d'enviar una agrupació o onada d'enemics cap al jugador, els enemics seran evolucionats per a que siguin màximament efectius amb l'estat del mapa actual. La seva funció *fitness* expressarà el nombre d'enemics que han arribat al final del camí o la distància que han aconseguit viatjar. Durant la simulació es realitzaran una sèrie d'iteracions agafant la mitjana dels valors, ja que una és bastant indeterminista.

Per a les torres també utilitzarem un algorisme evolutiu, que es basarà en situar els atributs entre un mínim i un màxim. La reproducció dels individus es realitzarà quan el jugador col·loqui una torre i, aquesta es reproduirà amb una altre de la població de torres. L'evolució es realitza per les eleccions del jugador, sense haver així de definir una funció *fitness*.

3.6 Dissenys semblants

Per a trobar dissenys de joc semblants s'ha cercat pels tres aspectes de disseny destacats: que sigui del gènere *Tower Defense*, l'ús d'un algorisme d'intel·ligència evolutiva i utilitzant el dispositiu *LEAP Motion Controller*.

Aquest disseny difereix d'altres jocs del gènere i encara que compleix amb les característiques de disseny del gènere, no s'ha trobat cap referent. Essent la principal diferència l'ús d'un terreny en tres dimensions, on els enemics poden anar per totes les cares. Més endavant veurem quines conseqüències ha tingut aquest disseny.

Referents que continguin intel·ligència evolutiva només s'ha trobat el joc descrit a l'article de *Avery et al. (2011)*[3], d'on s'ha basat la IC del projecte.

Finalment, tampoc s'ha trobat cap referent utilitzant el *LEAP Motion Controller*, ja que al ser un dispositiu recent la llibreria d'aplicacions és més reduïda. Tot i així, s'ha vist que la tendència d'ús d'aquest dispositiu és la combinació amb unes ulleres de realitat virtual.

Desenvolupament

El projecte s'ha realitzat de manera incremental ja que Unity ens permet obtenir resultats ràpidament. El procés de desenvolupament s'ha separat en dos apartats: la creació de l'art i el *gameplay*.

4.1 Art

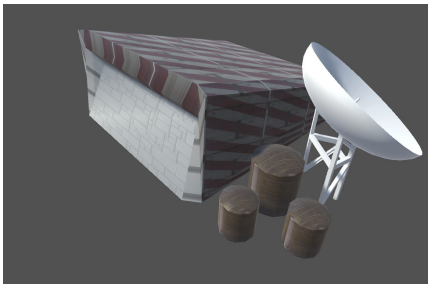
S'ha realitzat tot l'aspecte visual del joc, és a dir, els models, les textures, les animacions, el disseny de la GUI i els menús. A continuació detallarem de manera resumida el procés i el que es destaca de cada element.

Els objectes d'inici i de final del camí (figures 4.1a i 4.1b) són models que no contenen ossos ni animacions, a excepció de l'animació de l'ou de la figura 4.1a ja que aquesta crea la il·lusió de que l'enemic sorgeix de l'ou. L'animació d'explosió és una animació creada amb el motor de físiques de *3ds Max*, aplicant diferents forces i renderitzada a una animació de trajectòria.

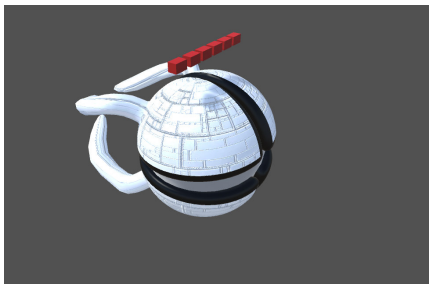
El model de l'enemic (figura 4.1c) és modelat a partir d'una esfera i és animat utilitzant animació esqueletal. La bala (figura 4.1d) és realitzada amb el sistema de partícules de *3ds Max* i conté unes 5000 partícules que s'han renderitzat en una serie de fotogrames. Dins el videojoc, crearem la bala utilitzant un petit sistema de partícules de Unity, amb unes 5 partícules que durant la seva vida reproduiran la sèrie d'imatges renderitzades, aconseguint així un efecte d'un sistema de partícules molt gran amb un cost molt baix.

La torre (figura 4.1e) és un model creat per tres peces i té una animació esqueletal que deforma el canó per afegir l'efecte del canó disparant. Les diferents peces s'utilitzen per a les diferents rotacions en el procés d'apuntar cap a l'enemic. A la figura 4.1e també podem observar una esfera de color verd que ens mostra el rang de tir que disposa la torre. Per aconseguir aquest efecte s'ha creat un *shader* amb el *Fresnel Effect* [11].

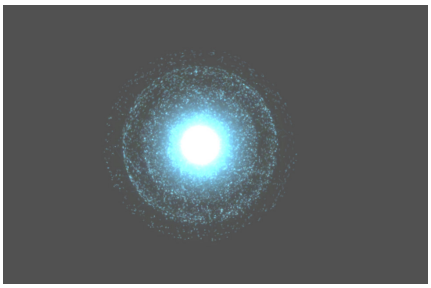
(a) *Inici del camí.*



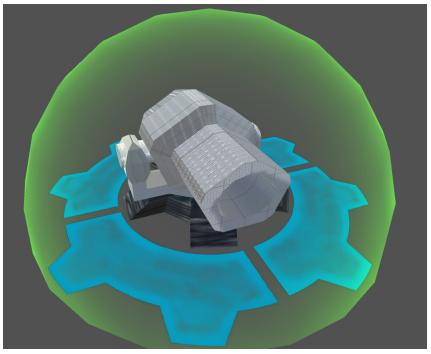
(b) *Final del camí.*



(c) *Model Enemic.*



(d) *Model d'una bala.*



(e) *Torre amb radi d'atac.*

Figura 4.1: *Els elements del joc*

Els cubs del terreny contenen una textura que és un atles de diferents textures. Les coordenades de quina textura s'utilitzarà són elegides aleatòriament al començament de la partida. Els cubs utilitzen mapes de normals (*Normal maps*) per aconseguir distorsionar la llum i crear efectes 3D.

La realització de la GUI i menús s'ha realitzat utilitzant *Photoshop*, creant diferents capes i efectes per cada estat de cada element. Un exemple serien els menús formats per botons, que s'han creat diferents efectes pels diferents estats del botó estàtic (*idle*), tenint el cursor a sobre o pressionat. Podem veure imatges de la GUI i els menús en l'apartat de resultats (figures 5.1, 5.2a, 5.2b i 5.2c).

Els únics continguts gràfics que no s'han creat han estat les imatges de la *skybox*, les icones dels atributs de torres i enemics, i la tipografia, que han estat obtinguts d'Internet.

4.2 Gameplay

El *gameplay* conforma tot el joc i, tal com podem veure en el diagrama de mòduls de la figura 4.2, conté tots els elements del joc i com interactuen entre ells. A continuació veurem en detall cada mòdul, els quals utilitzen el motor Unity internament.

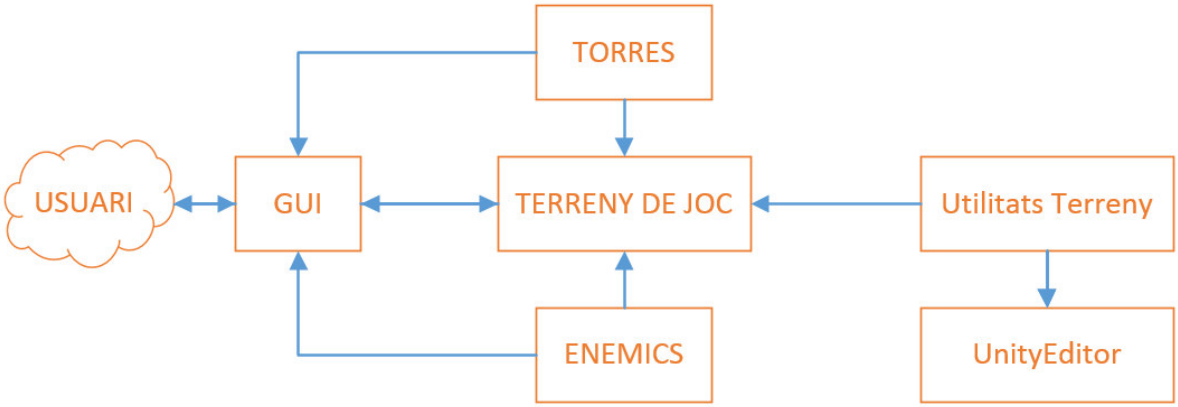


Figura 4.2: *Diagrama de mòduls.*

El joc consisteix en que l'usuari va col·locant torres per eliminar els enemics. Per a fer-ho, el jugador ha d'invertir energia en la compra de torres i aquesta energia ha de ser administrada de manera estratègica.

La gestió i el canvi d'informació entre els diferents mòduls ve produïda principalment per cinc esdeveniments: quan un enemic arriba al final del camí, quan una torre derrota un enemic i quan es crea, millora o destrueix una torre. Quan un enemic arriba al final del camí, el jugador perd vida i es comprova si la vida restant li permet continuar jugant o no. Ara bé, si l'enemic és derrotat, s'incrementa la puntuació i l'energia del jugador. Als esdeveniments de creació o millora es comprova que el jugador tingui energia suficient per realitzar l'acció. Finalment, la venta d'una torre permet al jugador recuperar energia.

El mòdul de la GUI s'encarrega de mostrar els diferents elements i la informació dels mòduls de torres, el terreny de joc i els enemics. L'usuari interactua amb la GUI per a crear els esdeveniments de creació, millora o destrucció de les torres. Aquests esdeveniments utilitzen el terreny de joc per trobar la posició i crear, millorar o destruir la torre seleccionada. El terreny de joc utilitza la meta-informació sobre el terreny creada per les funcions del mòdul d'utilitats del terreny i aquest mòdul també interactua amb

l'editor de Unity per millorar les funcionalitats de l'editor. Finalment, els mòduls de torres i enemics s'encarreguen de la gestió i les necessitats pròpies de cada element, com ara el moviment del model, atacar, la vida, etc.

4.2.1 El terreny de joc

El primer mòdul a implementar del videojoc és el terreny de joc, ja que contindrà tots els altres elements. Aquest consistirà en formar una estructura connectada a partir de cubs.

Començarem per preparar l'element bàsic que formarà el terreny: el cub. El cub ens ha de permetre tenir un control sobre les seves cares per a col·locar torres o models que representen l'inici i el final del camí. A més a més, hem de poder rotar el cub així com tots els objectes de les seves cares. Finalment, també hem de poder trobar un camí entre dues cares del mapa.

Com es mostra al diagrama de classes de l'annex A.4, es pot observar que cada cub conté sis objectes de la classe *CubeFace*. Els sis són fills[1] del cub i cada un representa una cara del cub, pel que cada un ens permet col·locar un model a una de les cares. Com que els models es creen com a fills dels objectes *CubeFace* i aquests són fills del cub, si rotem el cub també rotaran tots els objectes que conté.

La creació d'un mapa consisteix en la col·locació dels cubs un al costat de l'altre fins aconseguir l'estructura que es desitja. A l'inici de la partida cada cub s'identifica a les funcions d'utilitat, que generen les estructures necessàries per l'obtenció dels veïns de cada cub podent cercar camins, així com si el cub pot rotar i amb quin eix de rotació.

Un cop tenim el veïnatge de les cares, per poder trobar camins entre les cares s'ha utilitzat l'algorisme $A*$ [25]. S'ha programat aquest per què ens permet d'una manera ràpida trobar camins de cost mínim sense repetir cares. Hem definit la funció heurística com la *Distància de Manhattan*, ja que no permetem moviments diagonals, i el cost de cada cara és d'1, encara que pot augmentar si hi ha una torre.

[1] L'organització de Unity permet jerarquitzar objectes, fent les transformacions dels fills relatives a les dels pares.

4.2.2 Utilitats del Terreny

Aquest mòdul del joc permet la creació de meta-informació sobre el terreny, mitjançant la creació de diferents scripts que automatitzen comportaments i modifiquen i personalitzen l'editor de Unity per facilitar el treball.

Una de les funcionalitats d'aquests scripts és la creació d'una estructura que permet trobar el veïnatge de cada cara. Per aconseguir-ho, fem una transformació al centre de cada cara transformant-ho a un punt d'un espai de tres dimensions, on les coordenades són nombres enters. Aquest espai ens permet cercar els veïns d'una cara incrementant o disminuint una unitat en la direcció a cercar.

El script parteix de que tots els cubs tenen la mateixa mida i que cada cara està a una distancia ± 1 del seu centre. Per tant, el script aplica la funció 4.1 a cada cara de cada cub, obtenint així la representació de les cares. Apart de poder trobar fàcilment veïns, també ens permet trobar i desactivar les cares d'unió amb altres cubs, ja que coincideixen els punts de l'equació. En la figura 4.4 podem veure una representació gràfica de dos cubs amb les cares representades per punts (esferes), i la col·locació que realitza el script, mostrant dues cares que son veïnes.

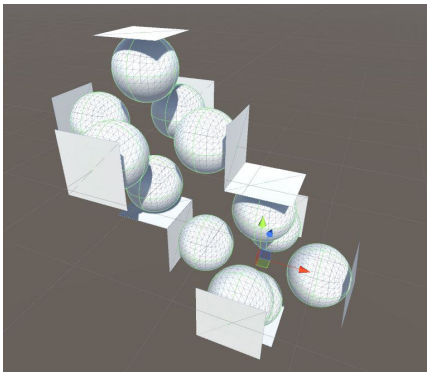


Figura 4.3: *Representació de dos cub, els seus PathPoints i les cares veïnes.*

$$\overrightarrow{Point} = 2\,\frac{\overrightarrow{Center}}{\overrightarrow{Scale}} + \overrightarrow{Normal} \tag{4.1}$$

$$\text{on } \overrightarrow{Point} = (x, y, z) \in \mathbb{Z},\ \ \overrightarrow{Center} = (X_c, Y_c, Z_c),$$
$$\overrightarrow{Scale} = (S_x, S_y, S_z) \ \text{ i } \ \overrightarrow{Normal} = (N_x, N_y, N_z).$$

Un altre script realitzat és l'automatització que troba els cubs que poden rotar i en quin eix. Per a fer-ho, es realitza un procés similar al de trobar veïns de cada cara però pels cubs, utilitzant els centres dels cubs per extreure l'organització d'aquests i decidir, per cada cub, si pot o no rotar, així com en l'eix que ho ha de fer.

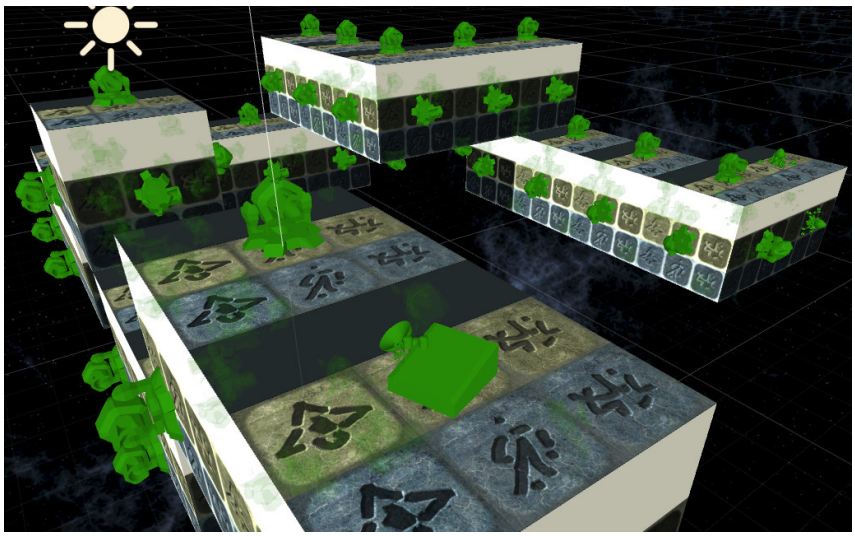


Figura 4.4: *Personalitzacions de l'editor Unity.*

Finalment, també s'han realitzat modificacions a l'editor de Unity, afegint eines per facilitar el procés de construcció, podent visualitzar petits models en verd (figura 4.4) que representen el que hi ha o pot haver-hi en cada cara, permetent modificar-ho mitjançant la GUI de Unity. També es pot decidir quina cara o cares són l'inici del camí (on surten els enemics), on es troba el final d'aquest (el que s'ha de defensar) i decidir quins cubs poden rotar i amb quin eix de rotació. Aquestes funcionalitats de l'editor, que no estan disponibles a l'usuari, s'han creat per facilitar la tasca al desenvolupador així com també per aprendre a realitzar aquestes personalitzacions.

4.2.3 Enemics

El mòdul de creació dels enemics ha consistit en implementar diferents components: el moviment, l'administració de múltiples enemics i els atributs evolutius.

Al principi, al no tenir un model realitzat, s'ha utilitzat un model 3D tipus càpsula de Unity per programar el moviment. El moviment consisteix en seguir un camí, desplaçant-se per les diferents cares dels cub, rotant quan el model es trobi al final de la cara.

L'estructura de múltiples enemics s'ha realitzat utilitzant un *pool d'objectes*. Aquests tenen precarregats un conjunt de models d'enemics, preparats per ser activats i inicialitzats amb els seus atributs. Tal com podrem observar al diagrama de classes A.6, la classe *EnemyController* s'encarrega del moviment del model.

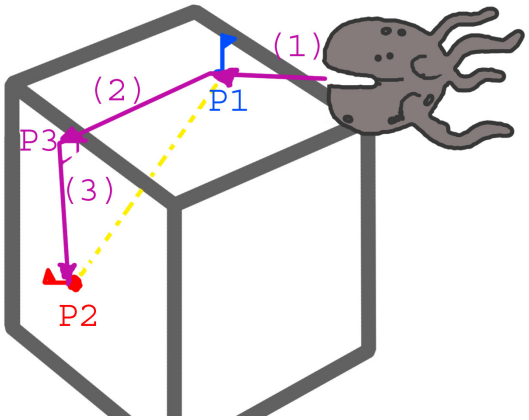


Figura 4.5: *Representació moviment.*

El procés de moviment (figura 4.5) consisteix en demanar a cada cara del camí, una posició aleatòria de dins la cara. L'enemic es desplaçarà fins a la primera posició *(P1)* i un cop arribat demanarà la següent posició *(P2)* i es desplaçarà cap a ella mantenint-se al pla de la cara actual fins a detectar el final de la cara. Quan arriba a *(P3)* el final de la cara és detectat, i l'enemic rotarà i s'encararà cap al punt *(P2)*. El procés es repeteix fins arribar a la seva destinació.

Els atributs són creats per una agrupació d'enemics que, com s'ha explicat abans, es realitzarà mitjançant un algorisme evolutiu. Aquest definirà un cromosoma amb 4 gens amb valors entre 0 i 1, uns paràmetres constants per ajustar el pes de cada propietat i el paràmetre de puntuació. Amb ells podem definir una agrupació d'enemics com:

- Nombre d'enemics en l'agrupació: $gen0 * 20 + 1$
- Punts per enemics: $Punts\_totals/(\#\_d'enemics + 2)$
- Factor de pes: $gen1 + gen2 + gen3$
- Velocitat: $(gen1/Factor\_de\_pes) * Punts\_per\_enemic * costant\_velocitat$
- Vida: $(gen2/Factor\_de\_pes) * Punts\_per\_enemic * costant\_vida$

On $gen0$, $gen1$, $gen2$, $gen3$ són els gens que formen el cromosoma.

Cada agrupació d'enemics segueix el cicle que es mostra a continuació, que es repeteix fins que s'acaba el temps o el jugador es queda sense vida i perd el joc:

1. Es genera la població inicial de cromosomes i seleccionem el primer com el millor individu.
2. S'envia l'onada utilitzant el millor individu i esperem fins que tots els enemics que formen l'onada han arribat al final o han estat derrotats abans de passar al punt 3.
3. Es realitza una simulació de la resta de la població, seguint els camins que ha utilitzat l'onada del punt anterior. Una única iteració és considerada poc determinista i, per tant, es realitzen varies iteracions del procés de simulació per obtenir resultats més fiables.
4. Amb la mitjana dels valors dels *fitness* obtinguts es reprodueixen els millors individus i es selecciona el millor.
5. Adaptem els paràmetre puntuació en funció del resultats de l'onada enviada al jugador.
6. Tornem al punt 2.

4.2.4 Torres

El mòdul de torres també ha consistit en implementar diferents components: detecció d'enemics, rotació la torre, i tret i els atributs evolutius, tal com podem veure en el diagrama A.7.

Pel que fa al terreny, per poder col·locar una torre primer hem de seleccionar una cara. La classe *TowerManager* crea la torre i l'envia a la cara seleccionada. Un cop podem col·locar torres al mapa, passem a la detecció d'enemics dins el rang. Per aconseguir-ho, utilitzem el motor de físiques de Unity i creem un cub que ens notifica si un enemic ha entrat o sortir d'aquest, mantenint la llista *FIFO*[2] d'enemics que es troben dins el cub.

Un cop hem tingut el model 3D de la torre, s'ha realitzat l'automatització d'apuntar i disparar. Apuntar consisteix en rotar les diferents parts del model per apuntar el canó cap al primer enemic de la llista i, si tenim l'enemic al punt de mira, disparar cap a l'enemic. Quan el canó dispara, es que crea un objecte bala en la direcció i amb la velocitat de tret de la torre. Aquest objecte bala té associada una esfera en el món de físiques de Unity que ens permet detectar quan la bala intercepta un enemic.

[2]FIFO: First In, First Out (primer en entrar, primer en sortir).

Com comentàvem al capítol anterior, els atributs defineixen les característiques d'una torre, el seu dany, el rang de tir, el temps de recàrrega, etc. Per trobar els valors de manera dinàmica utilitzem un algorisme evolutiu que situa els atributs entre un valor mínim i màxim. Cada atribut te associat un gen i l'agrupació d'aquests forma el cromosoma de la torre. El càlcul del valor de l'atribut de la torre es realitza mitjançant la funció 4.2, on el gen associat a l'atribut pot prendre valor entre 0 i 1. Inicialment, els gens prenen valors de 0 a 0.2 i no de 0 a 1, per a que el jugador no tingui torres massa bones. A mesura que el jugador va construint torres aquestes es reprodueixen per obtenir millors torres.

$$valor\_atribut = gen * (maximValor\_atribut) + (minimValor\_atribut) \tag{4.2}$$

4.2.5 GUI

La GUI s'encarrega de mostrar a l'usuari la informació de la partida, així com d'interactuar amb ell. La informació que mostra és la vida, la puntuació, l'energia restant, la informació sobre els enemics, el temps que resta de joc i la informació sobre la torre.

La interacció de l'usuari permet moviments de càmera, rotar cubs i la navegació de menús que permeten construir torres, millorar-les, destruir-les o sortir del joc. Aquestes interaccions poden ser capturades per dos dispositius d'entrada: la combinació teclat-ratolí o amb el *LEAP Controller*.

a) Teclat-ratolí

Per a simplificar la interacció i l'ús de les tecles utilitzarem la situació del cursor, el botó primari, el botó secundari i el botó «ESC».

El moviment del ratolí ens permet rotar la càmera des del seu origen. Aquest moviment ens permet més control sobre el mapa però ens fa fixar el cursor del ratolí al centre de la pantalla i, al tenir-ho fixat, ens permet que l'objecte d'interès del jugador estigui sota el cursor. Utilitzem les funcionalitats de *raycasting* de Unity per detectar què hi ha. El *raycast* ens permet llançar un raig imaginari des d'un origen que segueixi una direcció, detectant tots els objectes que tinguin presència en el món de físiques de Unity. D'aquesta manera podem detectar les torres i cubs.

Si es detecta un cub i en funció d'alguns dels botons que pressioni s'interactua amb aquest: si pressionem el botó primari intentarem rotar el cub i si pressionem el botó secundari mostrarem el menú de construcció, que consistirà en mostrar les diferents torres que el jugador pot seleccionar i construir. Per a poder col·locar una torre hem de trobar primer a quina cara s'ha de col·locar. Per trobar-la, utilitzant el vector de la normal del punt de col·lisió entre el raig i el cub, ja que amb la normal podem cercar quina de les cares té associada. Cada cop que rotem el cub canviem la normal de la cara, per tant cada cara guarda l'associació inicial i l'actual per assegurar una única col·locació d'una torre per cara.

En canvi, si en el cub detectem una torre podem extreure els seus atributs i mostrar-los per pantalla, així com pressionar el botó secundari per a que ens aparegui el menú de millora o venta de la torre. Finalment, tenim l'acció de mantenir el botó secundari pressionat i moure el ratolí, fent que la càmera roti respecte el centre del terreny, canviant la visió de l'usuari.

b) LEAP Controller

El controlador ofereix principalment tres maneres de treballar, oferint diferents abstraccions: per punts, que seria la de més baix nivell; treballar detectant les mans i els dits; i per gestos, que seria la de més alt nivell. La integració al projecte s'ha realitzat utilitzant una combinació d'aquests i, com que ja teníem la integració amb el teclat+ratolí, el que s'ha realitzat és convertir el controlador en un ratolí virtual.

Per extreure les característiques d'un ratolí, es detecta la mà dreta i s'extreuen els moviments entre les diferents iteracions, utilitzant el punt del palmell de la mà per detectar el moviment. Per la detecció dels botons primaris i secundaris s'ha utilitzat la detecció de gestos, detectant el gest *tap* de la llibreria en els dits índex (botó primari) i cor (botó secundari). Finalment, s'ha creat un ratolí virtual mitjançant crides a llibreries externes, que ens permeten crear un ratolí per *software*. La creació d'aquest ratolí virtual és dependent del sistema operatiu: per al sistema Windows s'ha utilitzat la llibreria externa *User32.dll* i per a Mac OS la llibreria externa *Quartz*.

Per a la rotació de la càmera respecte el centre del terreny utilitzarem les dues mans per crear el moviment.

Resultats 5

En aquest apartat es veuran els resultats del videojoc: el que s'ha aconseguit fer, el seu funcionament, les validacions que s'han realitzat i els errors que s'han detectat, així com la valoració dels usuaris que l'han provat.

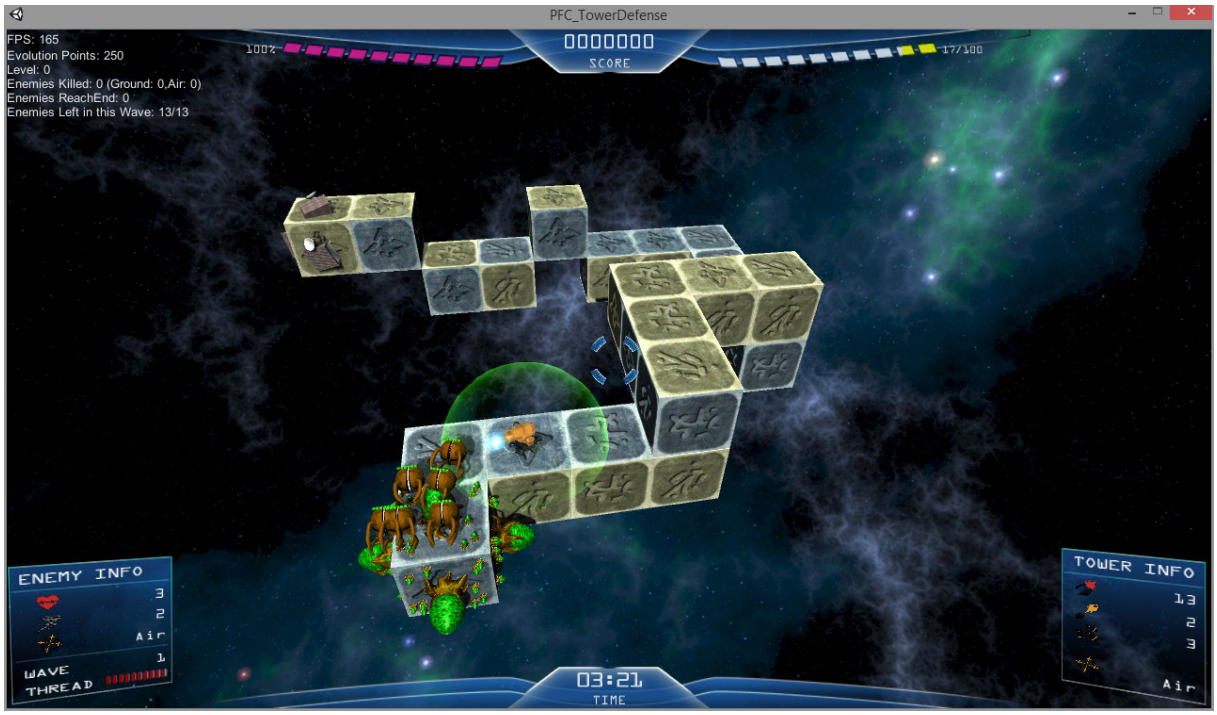
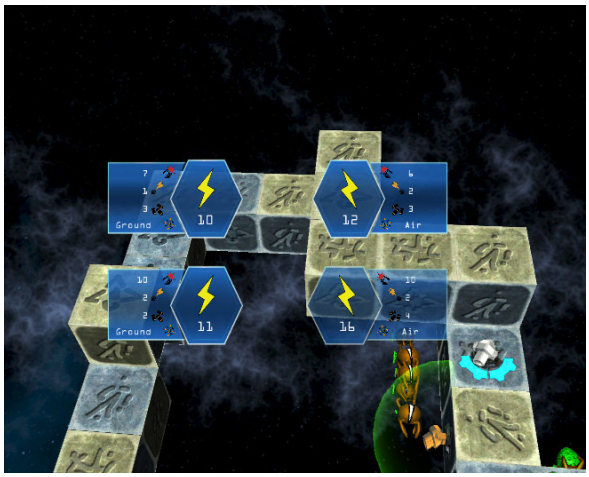


Figura 5.1: *Videojoc captura de pantalla.*

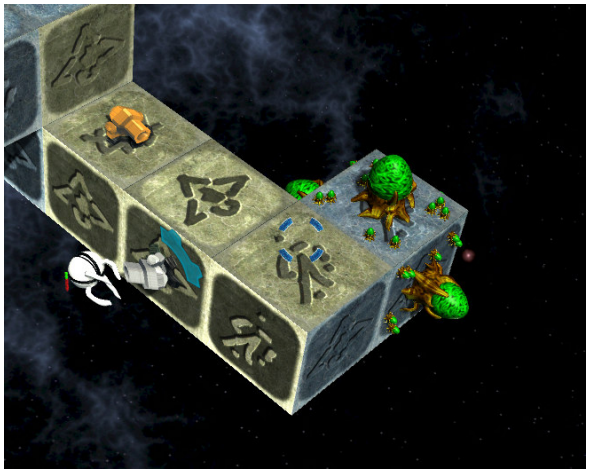
A la figura 5.1 es pot observar tots els elements que formen la GUI. A la part superior podem trobar la informació relacionada amb el jugador, a la part esquerra l'indicador de la vida, a la part central la puntuació i a la part esquerra l'energia. A la part inferior podem trobar la informació relacionada amb l'onada dels enemics actual, a la part central el temps que resta de joc i a la part esquerra la informació de la torre a la qual estem apuntant. La informació dels enemics mostra els atributs de vida, la velocitat i el tipus de l'onada actual. També podem veure quantes onades hem superat i un indicador que reflexa el grau d'amenaça, que ens permet veure si encara hi ha enemics al mapa. La informació de la torre ens mostra els diferents atributs de la torre que tenim seleccionada: el dany, el rang de tir, la velocitat del tret, el nivell i el tipus, respectivament.

(a) *Menú de joc.*

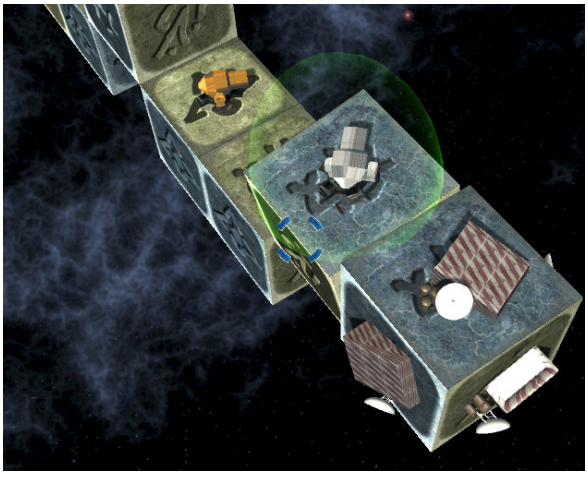


(b) *Menú construcció.*

(c) *Menú millora o venta.*



(d) *Inici del camí.*



(e) *Final del camí.*

Figura 5.2: *Captures del videojoc*

Al conjunt de figures 5.2 podem observar diferent elements del joc, com els models dels enemics, de color blanc si són de tipus *ground* i de color marró si són de tipus *air*. Hi podem veure les diferents torres de color gris o daurat, segons si són contra tipus *ground* o *air*, respectivament. També podem observar que les torres tenen un element gràfic de color blau: aquest mostra el nivell de la torre. A la figura 5.2a podem veure el menú principal de joc, el qual permet decidir el temps de durada de la partida. Podem veure el menú de construcció a la figura 5.2b, on es pot apreciar com es mostren els diferents atributs i el cost que té cada torre, així com què es permet seleccionar entre dues torres de tipus *ground* i dues de tipus *air*.

A la següent figura (figura 5.2c) podem veure el menú de millora o venta, i el cost i els atributs que tindrà la torre si aquesta és millorada, i també permet destruir (vendre) la torre. Finalment les figures 5.2d i 5.2e mostren el punts d'inici i final del camí, respectivament. Els camins es poden formar des de qualsevol dels punts d'inici fins a qualsevol dels punts finals. Aquesta funcionalitat va ser inclosa per poder fer que el jugador defensés diferents punts i que cada un tingués una vida associada, afegint al joc més estratègia. Tot i així, aquesta funcionalitat no s'ha completat.

El procés de validació de cada element del joc ha consistit en el seu testeig al moment de cada funcionalitat. Al realitzar una implementació incremental afegint millores, el testeig s'ha realitzat de manera incremental. En alguns casos s'han creat algoritmes en programes apart utilitzant *C#*, podent testejar amb exemples petits i controlats, i exportant la funcionalitat creada a Unity.

Com a incidències destacades, el desenvolupament del joc ha portat a constants modificacions i havent de redissenyar el codi vàries vegades. Les modificacions han vingut principalment per problemes de rendiment del joc. Es treballava amb dues màquines amb capacitats *hardware* molt diferents: un portàtil (Mac Os) i un ordinador Windows d'última generació, i uns dels objectius era que havia de funcionar al dos. Això ha portat a realitzar constantment *profiling* del joc, amb les eines que ofereix Unity, per detectar colls d'ampolla i millorar el rendiment. Això ha portat a profunditzar en el *software* de Unity, així com a millorar la meva programació.

Algunes d'aquestes modificacions han estat passar a un model de *pool d'objectes*, evitant la creació i destrucció d'objectes, i actualitzar manualment els valors de les variables,

evitant les funcions d'actualització pròpies de Unity. Tot i que el rendiment final no ha estat l'esperat, podem observar a la figura 5.3 que és degut a les funcions de renderitzat (color verd), i es considera que la causa està en els *shaders* de Unity, que cada cop són més potents però que per aplicacions simples afegeixen un *overhead* massa gran. A la figura també podem observar els diferents temps d'execució de cada *frame* que requereixen els diferents elements del joc: els scripts (en blau), el procés de renderitzat (en verd), el motor de físiques (en taronja), etc.

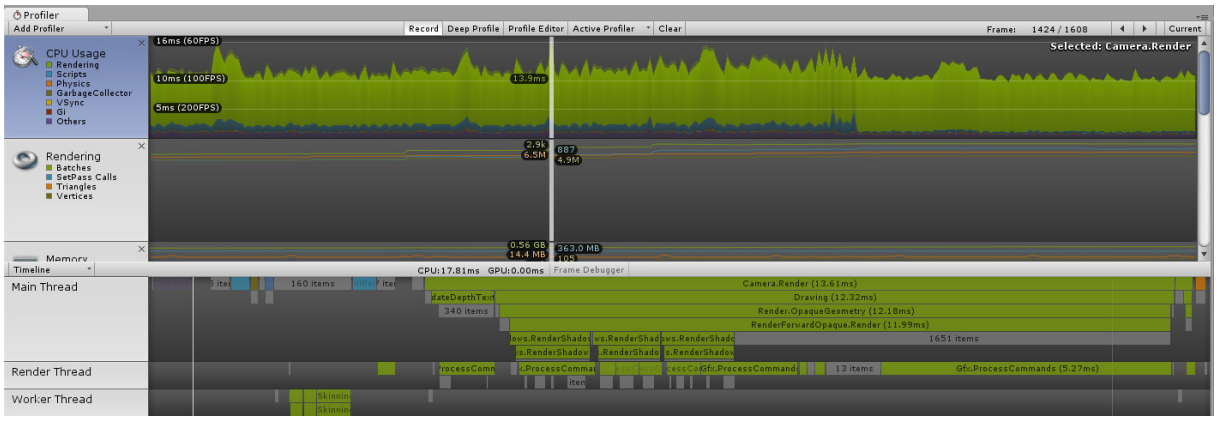


Figura 5.3: *Unity Profiling.*

El disseny del mapa va portar a que l'algorisme d'adaptació de la dificultat al jugador tingués una dificultat afegida. La dificultat radica en la simulació dels enemics en un mapa de *Tower Defense* típic en dues dimensions amb un únic camí on tots passaran per les mateixes torres, ja que seguiran el mateix camí. Ara bé, en el mapa del videojoc els enemics poden prendre diferents camins i, per tant, entre simulació i simulació no podem assegurar que els enemics estiguin afectats per les mateixes torres.

Tot i així, per minimitzar-ho s'ha fet que l'agrupació d'enemics enviada al jugador crei els diferents camins i que les simulacions només poden utilitzar els camins ja creats, ja que al realitzar diferents iteracions de la mateixa simulació reduïm la probabilitat de dispersió dels camins. El resultat és bastant satisfactori encara que caldria dedicar més refinament en els paràmetres de l'algorisme, ajustant millor els increments de dificultat i entre els atributs de torres i enemics.

A nivell dels usuaris, des del principi, i preguntant a amics i amigues, el joc es considera innovador. Tot i això, el mapa que el feia més innovador ha estat el que ha causat més problemes. Al voltant d'uns 20 usuaris han provat el joc, en un rang d'edat d'entre 20 i

30 anys i, després d'ensenyar el resultat final i demanar la seva opinió, tots coincidien en tres aspectes: el consideren innovador i els hi agrada el concepte de joc, però el mapa i l'ús del *LEAP Controller* se'ls hi feia molt complicat, ja que el *LEAP* és molt nou i el provaven per primera vegada. També jugaren amb el teclat i ratolí, obtenint d'aquesta manera una millor experiència de joc però, tot i així, la trobaven complicada. Cal dir que la complicació que tenien també radica en la novetat del joc: s'esperava que el mapa donés una nova estètica al gènere però ha resultat una complicació excessiva.

Conclusions i millores

6

- S'ha realitzat un videojoc del gènere *Tower Defense*, amb una càmera en tercera persona, per entorns PC Windows i Mac OS, excloent el sistema operatiu Linux.

- S'ha realitzat un mapa de cubs, innovador en el gènere, que permet orientar les torres. Aquest videojoc prova que és possible realitzar un disseny així. Malgrat això, sembla que aquest mapa no es del tot viable com un producte final ja que s'aparta de la senzillesa del gènere.

- També s'ha realitzat un sistema que ajusta dinàmicament la dificultat del jugador. A causa de la dificultat del mapa i els problemes que comportava la topologia per realitzar simulacions en igualtats de condicions, no es podia realitzar un anàlisi exhaustiu de la seva efectivitat. Tot i així, crec que els resultats han estat bons i que el concepte d'ajustament dinàmic de la dificultat no és utilitzat en gaires jocs, per la dificultat que comporta el tunejament de les variables.

- S 'ha realitzat el control del joc utilitzant el *LEAP Controller*, un dispositiu nou i que té una corba d'aprenentatge bastant gran. El *LEAP Controller* té molta sensibilitat en alguns moments i en d'altres no reconeix els gestos més simples, cosa que ha fet que sigui més fàcil utilitzar la combinació teclat-ratolí. Seria bo tenir un tutorial que ensenyi els controls i com s'ha de utilitzar el *LEAP Controller* en el joc, així els usuaris es sentirien més animats a utilitzar-ho.

- El desenvolupament d'aquest projecte ha estat una experiència enriquidora que m'ha aportat molts coneixements que en la carrera no s'havien vist, com els diversos motors de jocs, la creació d'art i altres tecnologies per la creació de videojocs, així com nous mètodes per l'usuari d'interactuar amb el *software*. S'ha vist la importància d'una bona planificació, que també ha de comportar una gran previsió dels problemes que poden sorgir, així com que no sempre és possible realitzar totes les idees que es tenien al principi. Tot i tenir alguns problemes de temps, el resultat ha estat satisfactori.

La principal incidència del projecte ha estat el concepte de mapa, que ha portat a la majoria de les dificultats. Una de les possibles millores seria tornar a fer la part del mapa, transformant-lo en un pla i simplificant així el joc.

Altres millores podrien ser:

- Afegir més tipus d'enemics i torres, així com més atributs per donar més funcionalitats, ja que el jugador tindria una varietat més gran per triar i podria definir més estratègies. Per aconseguir-ho, caldria realitzar més models en 3D i afegir les noves funcionalitats.
- La incorporació d'efectes sonors en el joc seria també una millora que ajuda a que l'usuari entri en l'atmosfera del joc.
- Millorar les habilitats de l'usuari com poder fer llançament directes als enemics i tenir més varietat de mapes, així com un editor de mapes que ajudaria a la longevitat del videojoc, proporcionant als jugadors l'opció de personalitzar diferents continguts del joc.

Bibliografia

[1] Autodesk. *3D Max*. [Online; Gener-2015] http://www.autodesk.com/products/3ds-max/overview

[2] Autodesk. *Autodesk 3ds Max Reference.* [Online; Octubre-2015] http://help.autodesk.com/view/3DSMAX/2016/ENU/

[3] Avery, P.; Togelius, J.; Alistar, E.; van Leeuwen, R.P., (2011): *Computational intelligence and tower defence games*. In Evolutionary Computation (CEC), 2011 IEEE Congress on, vol., no., pp.1084-1091. [doi: 10.1109/CEC.2011.5949738]

[4] Blender Foundation. *Blender*. [Online; Gener-2015] https://www.blender.org/

[5] Crytek. *CryEngine Features*. [Online; Setembre-2015] http://cryengine.com/features/

[6] David Scott. *Flash Element TD (videojoc)*. [Online; videojoc funcional; Setembre-2015] http://www.freewebarcade.com/game/flash-element-td/

[7] Epic Games. *Unreal Engine*. [Online; Setembre-2015] https://www.unrealengine.com/unreal-engine-4

[8] Glen Stevens. *Unity3D Best Practices.* [Online; Setembre-2015] http://www.glenstevens.ca/unity3d-best-practices/#Unity3D_Best_Practices

[9] Hunicke, R.; Chapman, V., (2004): *AI for Dynamic Difficulty Adjustment in Games*. In Proceedings of the 2004 Challenges in Game Artificial Intelligence AAAI Workshop. Northwestern University, San Jose, California, USA, 91-96.

[10] Kongregate. *Desktop Tower Defense*. [Online; videojoc funcional; Setembre-2015] http://www.kongregate.com/games/preecep/desktop-tower-defense

[11] Kyle Halladay. *The Basics of Fresnel Shading.* [Online; Desembre-2015] http://kylehalladay.com/blog/tutorial/2014/02/18/Fresnel-Shaders-From-The-Ground-Up.html

[12] LEAP Motion, Inc. *LEAP Controller - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Leap_Motion

[13] LEAP Motion, Inc. *LEAP Controller C# SDK.* [Online; Octubre-2015] https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Coordinate_Mapping.html

[14] LEAP Motion, Inc. *LEAP Motion Controller.* [Online; Setembre-2015] https://www.leapmotion.com/

[15] Maxon. *Cinema 4D Studio.* [Online; Gener-2015] http://www.maxon.net/products/cinema-4d-studio/who-should-use-it.html

[16] Microsoft. *Visual C# Documentation.* [Online; Octubre-2015] https://msdn.microsoft.com/en-us/library/618ayhy6.aspx

[17] Noobtuts. *Unity Tower Defense Tutorial.* [Online; Octubre-2015] http://noobtuts.com/unity/tower-defense-game

[18] PopCap. *Plants vs. Zombies.* [Online; Setembre-2015] http://www.popcap.com/plants-vs-zombies-1/

[19] Spiderling Studios. *Besiege (videojoc).* [Online; Gener-2015] http://www.besiege.spiderlinggames.co.uk/

[20] Steam. *Steam Controller.* [Online; Gener-2015] http://store.steampowered.com/app/353370/

[21] SuperHot. *SuperHot (videojoc).* [Online; Gener-2015] https://superhotgame.com/

[22] Unity Technologies. *Unity3D.* [Online; Setembre-2015] http://unity3d.com

[23] Unity Technologies. *Unity Documentation.* [Online; Octubre-2015] http://docs.unity3d.com/Manual/

[24] Unity Technologies. *Unity Scripting Reference.* [Online; Octubre-2015] http://docs.unity3d.com/ScriptReference/

[25] Wikipedia. *A* Search Algorithm.* [Online; Setembre-2015] http://en.wikipedia.org/wiki/A*_search_algorithm

[26] Wikipedia. *Age of Empires II (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Age_of_Empires_II

[27] Wikipedia. *Batman: Arkham Knight (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Batman:_Arkham_Knight

[28] Wikipedia. *Bioshock (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/BioShock

[29] Wikipedia. *Box–Muller transform.* [Online; Desembre-2015] https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform

[30] Wikipedia. *Crysis (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Crysis

[31] Wikipedia. *FarCry (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Far_Cry

[32] Wikipedia. *Kerbal Space Program (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Kerbal_Space_Program

[33] Wikipedia. *Kinect - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Kinect

[34] Wikipedia. *Rampart (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Setembre-2015] https://en.wikipedia.org/wiki/Rampart_(video_game)

[35] Wikipedia. *Shaders - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Shader

[36] Wikipedia. *Skeletal animation - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Skeletal_animation

[37] Wikipedia. *Sniper Elite (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Sniper_Elite

[38] Wikipedia. *StarCraft (videojoc) - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/StarCraft

[39] Wikipedia. *Texture Mapping - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Texture_mapping

[40] Wikipedia. *Tower Defense - Wikipedia, The Free Encyclopedia.* [Online; Gener-2015] https://en.wikipedia.org/wiki/Tower_defense

[41] Wikipedia. *Warcraft III: Reign of Chaos (videojoc) - Wikipedia, The Free Encyclopedia*. [Online; Gener-2015] `https://en.wikipedia.org/wiki/Warcraft_III:_Reign_of_Chaos`

[42] Wikipedia. *Wii Remote - Wikipedia, The Free Encyclopedia*. [Online; Gener-2015] `https://en.wikipedia.org/wiki/Wii_Remote`

Diagrames UML d'Anàlisi del Disseny

A.1 Diagrama de casos d'ús

A.1.1 Menú de joc

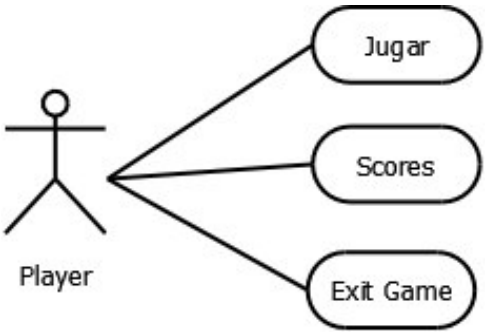


Figura A.1: *Cas d'ús dins el menú principal del joc.*

A.1.2 Durant la partida

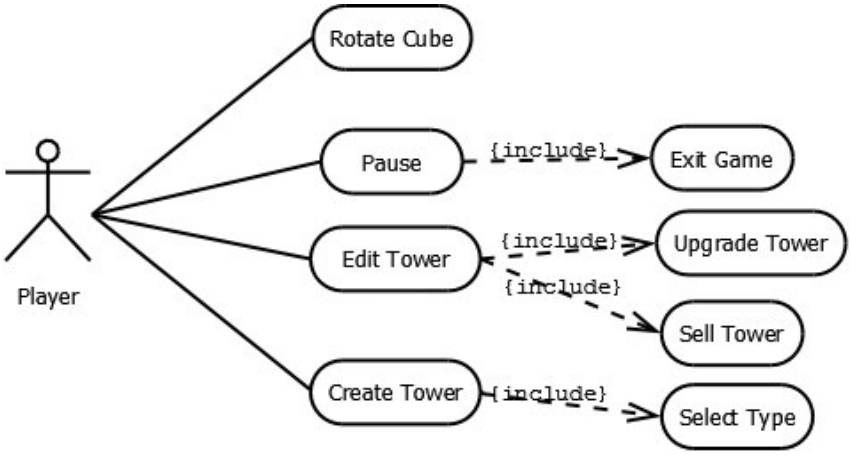


Figura A.2: *Cas d'ús del jugador en el temps de joc.*

A.2 Diagrama de classes

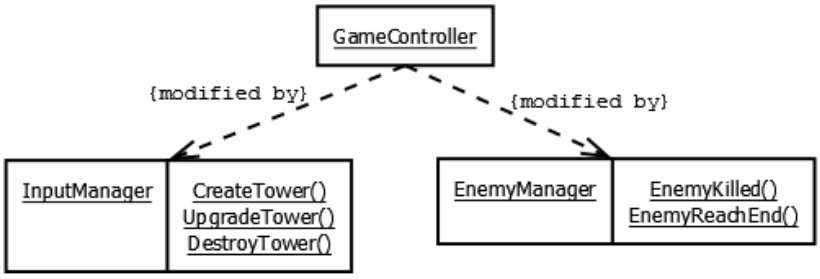


Figura A.3: *Diagrama de classes del joc.*

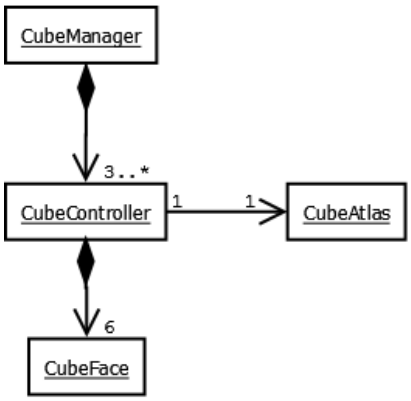


Figura A.4: *Diagrama de classes del terreny.*

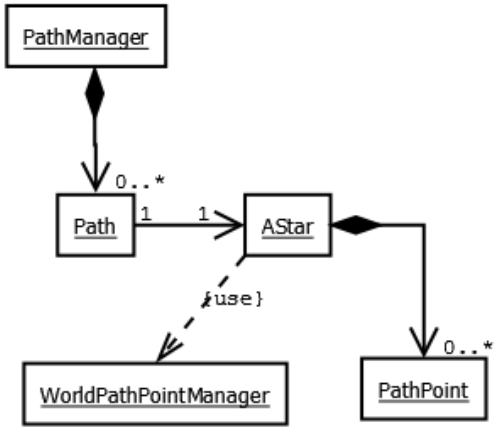


Figura A.5: *Diagrama de classes de la cerca de camins.*

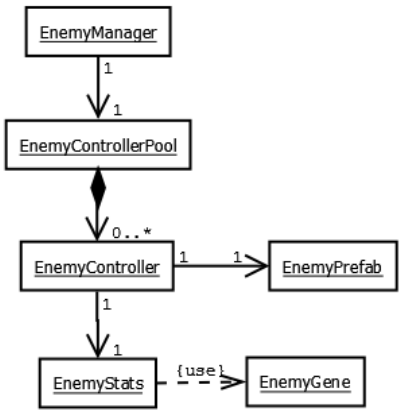


Figura A.6: *Diagrama de classes de les unitats enemigues.*

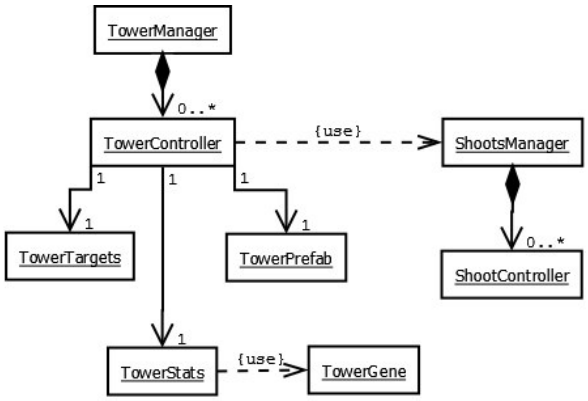


Figura A.7: *Diagrama de classes de les torres.*

Diagrama de Gantt

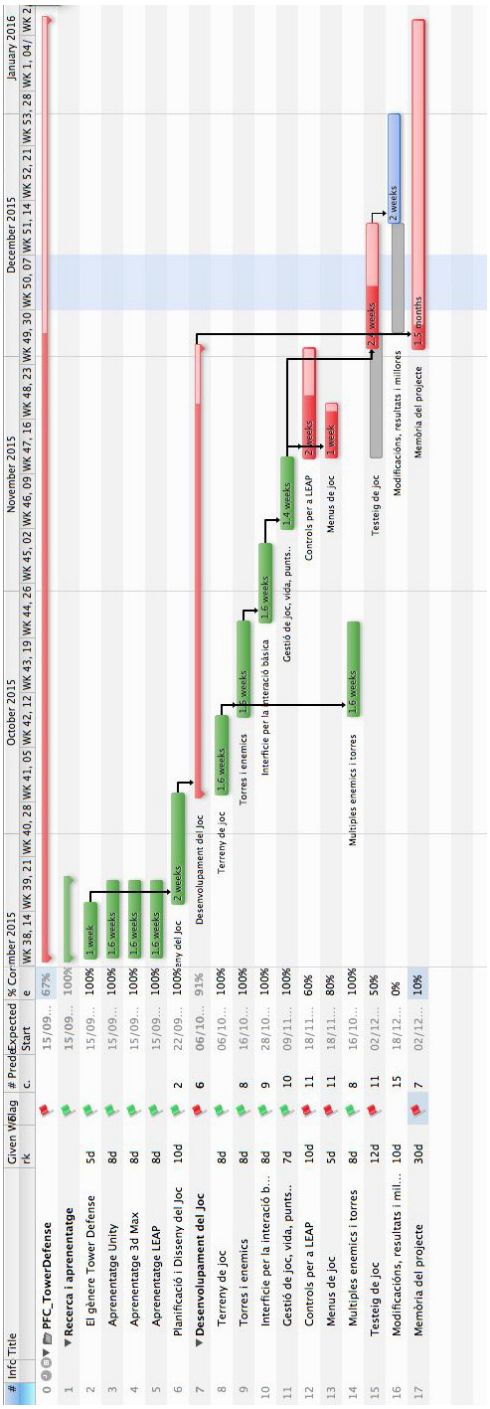


Figura B.1: *Diagrama de Gantt*