Universitat Oberta de Catalunya
www.uoc.edu

Treball de Fi de Grau

TÍTOL:	*Ultimatum to Earth*: Videojoc per a Windows Phone 7
AUTOR:	Jesús Bosch Aiguadé
TITULACIÓ:	Grau en Multimèdia
CONSULTOR:	Vicent Moncho Mas
DATA:	Gener de 2011

Agraïments

El primer agraïment ha de ser forçosament per a la Universitat Oberta de Catalunya com a institució. Tot sovint ens adonem a la vida de que les universitats van per darrera de les necessitats del mercat laboral, oferint titulacions que han canviat molt poc al llarg dels últims deu anys. En canvi a la UOC, i especialment en els estudis de Grau en Multimèdia, sóc conscient de que els estudis que he realitzat em són útils al mercat laboral des del primer dia de la primera assignatura que vaig començar a cursar, ja fa un bon grapat d'anys. Aquest fet fa que m'enorgulleixi el fet d'haver format part de la UOC durant tot aquest temps.

Deu ser un fet molt habitual en els treballs de fi de carrera presentats a la UOC, però no puc deixar de mostrar el meu infinit agraïment a la meva familia per la paciència que han tingut durant tot aquest temps, havent de dedicar molt de temps a la universitat, un temps que en molts casos els he hagut de restar de les relacions familiars.

Tampoc puc deixar passar la oportunitat de mostrar el meu agraïment a Microsoft Ibérica i en especial al equip de "Programas Académicos", Alfonso Rodríguez i Elisa García. Farà cosa de dos anys vaig unir-me als DotNetClubs que promociona Microsoft, creant el UOC DotNetClub, del que he estat i sóc coordinador des d'aleshores. Això m'ha permès viure moltes experiències úniques que d'altra banda no haurien estat possibles, conèixer a gent fantàstica i aprendre moltíssimes coses. També gràcies al UOC DotNetClub vaig iniciar-me en la tecnologia XNA, en la que es basa la part tècnica d'aquest projecte.

Resum

Fa ja molts anys que es podria dir que la nostra vida està essent informatitzada, però amb la actual generació de dispositius mòbils l'abast d'aquest canvi està creixent enormement, participant aquests en qualsevol tasca, per trivial que pugui semblar. Podem afirmar què els telèfons mòbils estan cambiant la manera en què tots nosaltres vivim les nostres vides.

Els mòbils d'avui dia poden tenir centenars i fins i tot milers **d'aplicacions** que ens poden ajudar en gran quantitat de petites tasques que ens anem trobant en el nostre dia a dia, amb un hardware d'alta capacitat i velocitat de processament que permeten la execució d'aplicacions que requereixen un gran rendiment, com son les aplicacions multimèdia relacionades amb l'entreteniment.

M'estic referint a realitat augmentada, videojocs 3D, pel·lícules sota demanda, música, i diferents tipus de serveis en temps real, com geolocalització assistida (GPS, triangulació de cèl·lules de cobertura telefònica i accés a xarxes conegudes), etc.

Amb aquests mòbils ens enduem les aplicacions a qualsevol lloc, i també els videojocs. Els llargs viatges en metro o tren per anar a la feina poden ser menys aborrits. Interactuem amb ells de la forma més natural possible, tocant-los, sense botons físics. Fins i tot amb la veu.

Tots aquests nous serveis crearàn –i ja estàn creant- la eclosió de multitud d'empreses que els hauràn de satisfer. En aquest sentit, el perfil d'un Graduat en Multimèdia s'hi ajusta perfectament.

En aquest treball desenvoluparé una part d'un d'aquests videojocs, concretament un joc d'acció en 2D que es desenvolupa a l'espai intergalàctic. El jugador podrà evadir-se en una batalla a les galàxies més llunyanes a bord d'una nau espacial. La lluita esdevindrà decisiva per a la supervivència de la raça humana.

Com és ben sabut, la creació d'un videojoc comporta una gran complexitat. Els jocs actuals contenen una gran quantitat de recursos multimèdia, bàsicament models 3D, dibuixos, música i efectes especials gràfics i de so. Tot això sincronitzat amb una part molt important de coneixements de programació avançats , que requereixen alhora bases de física (especialment dinàmica), àlgebra, vectors, i trigonometria. Per tant les empreses que produeixen videojocs acostumen a armar-se amb equips importants en nombre de persones d'alta qualificació.

Projectes destinats al gran mercat de consum poden tenir equips de més de 500 persones, incloient personal de gestió i màrketing.

Donat que aquest és un treball de final de carrera que s'ha d'elhaborar de manera individual, i què la planificació és molt limitada, esdevè un exercici de responsabilitat ser conscient de les meves limitacions, i per tant la versió del videojoc que sentregarà serà una demo, i no un videojoc complert amb diferents nivells, i provablement gràfics millor acabats i majors detalls tècnics en la programació. No obstant aquestes limitacions, el treball esdevè una versió totalment jugable i espero que adictiva, com tot bon videojoc.

Índex de continguts

CAPÍTOL 1. Introducció

1.1 Objectius i justificació del Treball de Fi de Grau

Un videojoc és la màxima expressió de les aplicacions multimèdia. Depenent del tipus de joc, pot incloure gràfics 2D i 3D, música, vídeo... i cal que tot funcioni de manera fluida en temps real. Desenvolupar un videojoc, doncs és un repte molt interessant per un Graduat en Multimèdia.

Així mateix, estem en un moment molt interessant dins del mercat dels dispositius mòbils, i especialment de les aplicacions, incloent-hi els videojocs, per aquests dispositius. Les principals companyies ofereixen als seus milions d'usuaris accés a una aplicació que s'anomena "Marketplace" dins del sector. Un Marketplace permet al propietari d'un telèfon mòbil descarregar demostracions d'aplicacions o jocs, i si li agrada pot descarregar-ne la versió completa, ja sigui aquesta gratuïta o de pagament.

Es tracta sens dubte d'un model de negoci emergent, i la realització d'aquest treball tindrà una aplicació directa en l'exploració d'aquest nou mercat.

De tots el sistemes operatius per a mòbil que existeixen al mercat actualment, he escollit Windows Phone 7. Aquest és un nou sistema desenvolupat per Microsoft. L'avantatge clar que tenen els productes d'aquesta empresa és que son relativament fàcils d'utilitzar, en comparació amb els seus competidors.

Puc afirmar doncs que desenvolupar un joc és més ràpid, i per tant més productiu, si es fa per a Windows Phone 7 amb la seva plataforma XNA, que si es fa per un altre sistema amb llenguatges de més baix nivell, com IPhone i el seu llenguatge de programació Objective C, molt similar a C++. Això és així per diversos motius, com que el llenguatge és de més alt nivell, conté un gran nombre de classes i utilitats que contenen molta funcionalitat, i sobretot, la gestió de la memòria que ofereix, que és molt més transparent per al programador del que ho seria amb Objective C, que utilitza punters. Per això la plataforma XNA i el seu llenguatge de programació C# son anomenats com un llenguatge de programació de memòria manejada.

No obstant la meva elecció, els altres sistemes no deixen de ser molt bons, i el que és també molt important, compten amb un recorregut relativament llarg, amb centenars de milers d'aplicacions publicades, mentre que Windows Phone 7 és un sistema totalment nou què acaba d'aparèixer al mercat en el moment d'escriure aquestes línies, amb un total de tot just 5.000 aplicacions publicades.

Espero que aquest treball em sigui útil per a consolidar els meus coneixements en programació gràfica, així com també la part de creació artística i disseny d'un interactiu com és aquest tipus d'aplicació. De passada espero poder arribar a explorar el Marketplace i la seva possible rendibilitat per a un desenvolupador de videojocs independent.

1.2 Enfocament i mètode seguit

Un videojoc és un tipus de projecte multidisciplinari, i no es pot dir que existeixi una metodologia concreta formal que en cobreixi tot el cicle de vida. No obstant a això, un videojoc no deixa de ser un producte de software, i com a tal existeixen diverses maneres d'afrontar-ne el cicle de vida del desenvolupament.

Seguiré un model iteratiu, considerant la entrega del TFG com a primera iteració, que serà una demostració a la que es pugui jugar del videojoc. Futures possibles iteracions em permetran millorar el contingut gràfic, d'audio, i també la lògica de programació de cara a la consideració de la possibilitat de publicar el videojoc al Marketplace de Windows Phone 7.

Aquest model iteratiu, considero que no seria útil aplicat directament sobre un videojoc perquè aquest és un tipus de projecte molt diferent del que ho podria ser una aplicació de gestió, es podria dir que és una barreja de producció audiovisual i producte de software, per tant caldrà aplicar una aproximació entre ambdós mons, però sempre basant-me en el rigor que aporta l'enginyeria del software.

Tot plegat s'acaba traduint en les següents fases:

- Planificació
- Disseny conceptual
- Disseny tècnic
- Desenvolupament
- Proves
- Entrega

Les diferències més evidents en la producció d'un software "clàssic" seran evidents en el disseny conceptual, on es parlarà de termes que tendiran a ser més artístics, d'usabilitat i jugabilitat, més que no pas tècnics.

També cal considerar que durant el disseny tècnic s'inclouran diagrames basats en UML

1.3 Planificació del projecte

El projecte ha estat previst de ser avançat per fases, entregades al tutor de manera progressiva. Aquestes tasques es divideixen de la manera següent:

- 31/10/2010: Disseny conceptual.
- 21/11/2010: Disseny tècnic.
- 10/01/2010: Versió funcional del videojoc, realitzades les primeres proves. Entrega memòria amb incorporació Proves d'usuari.
- 15/01/2010: Versió final del videojoc desenvolupada. Entrega de Conclusions.

1.4 Productes obtinguts

En finalitzar el desenvolupament d'aquest videojoc (anomenat "Ultimatum to Earth"), es generaran els següents documents:

- **Disseny conceptual**

  Esdevé una barreja d'anàlisi funcional que podríem trobar en un producte de software clàssic i el "briefing" d'un producte multimèdia. Defineix quina serà la funcionalitat del joc.

  Inclòs dins la secció "Disseny conceptual" d'aquest mateix document.

- **Disseny tècnic**

  Defineix en alt nivell les tecnologies utilitzades i la arquitectura del videojoc. Conté traces de tècniques d'enginyeria del software i també de disseny de produccions multimèdia.

  Inclòs dins la secció "Disseny tècnic" d'aquest mateix document.

- **Proves**

  Una aplicació necessita ser provada per l'equip de desenvolupament. Però especialment en els productes multimèdia, és molt interessant fer proves amb usuaris, i en això es centra aquesta secció.

  Inclòs dins la secció "Proves" d'aquest mateix document.

- **Conclusions**

  Inclou els següents fitxers:

  - Screencast que permet veure la meva presentació del treball, així com el joc en execució.

  - Presentació power point.

  Més informació dins la secció "Conclusions" d'aquest mateix document.

CAPÍTOL 2. Disseny conceptual

2.1 Introducció

“Ultimatum to Earth" té com a principal objectiu implementar un videojoc en 2D, per al dispositiu Windows Phone 7 aprofitant la tendència emergent dels dispositius smartphone.

2.1 Gènere

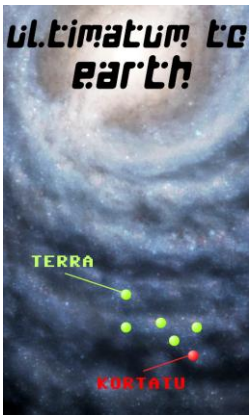
El joc forma part del gènere conegut popularment com “space shooters”, és a dir, que l’objectiu del jugador serà disparar i destruir als seus enemics, i sobreviure fins al final del nivell. Existeixen a més, traces de novel·la interactiva que el fan un joc més transversal. En aquest efecte el joc conté una història i un argument que donen sentit a la consecució dels objectius per part del jugador.

2.2 Descripció

Segle XXV, la raça humana explora l’espai en busca de nous mons per habitar, estenent colònies pacífiques en la constel·lació de Perseus.

Tot anava bé fins que l’any 2456, els exploradors espacials, amb el capità Blake al capdavant, són enviats al nou planeta Kortatu, on es toparan amb una sorpresa desagradable, el descobriment d’una raça intel·ligent amb tecnologia avançada i gens amistosa. La expedició és atacada amb molta violència. Els humans es veuen obligats a fugir del sistema, ja que la nau d’exploració no està preparada per defensar-se amb garanties. En els combats, la nau de Narbona és destruïda, encara que ella sobreviu al desastre. Blake farà el que calgui per recuperar a la noia...

Els humans fa segles que són una raça pacífica i no disposen de tecnologia militar avançada capaç d’aturar a les forces de Kortatu de manera clara, i la raça alienígena que habita el planeta: els Khorza. Ja no es tracta d’una guerra per obtenir recursos com les que havia viscut la història de la humanitat fins aquest moment, sinó per defensar la supervivència de la espècie.



Aquest argument introductori pot ser traduir-se en la creació de diferents nivells en un videojoc de guerra espacial. Donats els recursos limitats per la creació d'aquest projecte, en aquest treball s'entrega només un nivell a mode de demostració, però es deixarà preparada la estructura dels nivells següents a nivell conceptual, entrant en detall només en el primer nivell.

2.2.1 Personatges principals

1.2.2.1 Capità Blake

És el personatge principal o "heroi". Pilota la nau controlada pel jugador. Té un caràcter valent, decidit i amb una marcada irreverència cap als seus superiors.

2.2.2.1 Narbona

Narbona esdevé un dels objectius del joc. És segrestada pels Korzha, i Blake voldrà rescatar-la a tota costa.

No és una persona de perfil militar, però quan convingui no dubtarà a lluitar per tal de defensar la raça humana.

2.2.2 Nivell 1

En el primer nivell la nau d'en Blake topa amb les forces Kortatu i s'inicia una persecució. La nau d'exploració que és la que controla el jugador no té gaire armament i per tant l'objectiu d'aquest primer nivell és bàsicament la supervivència.

2.1 Objectius del jugador

El jugador controla una nau espacial amb una vista superior en dues dimensions. La nau pot ser controlada utilitzant l'acceleròmetre del dispositiu o bé la pantalla tàctil (a elecció del jugador).

La captura de pantalla mostra la pantalla d'opcions, on es pot seleccionar el volum de la música, dels efectes de so, així com el tipus de moviment, basat en l'acceleròmetre o la pantalla tàctil.

En els diferents nivell del joc hi ha diferents tipus d'elements:

- Naus enemigues
- Objectes
- Obstacles
- Foc amic
- Foc enemic

Els objectius del jugador són sobreviure fins al final de cada nivell, moment en el qual caldrà eliminar el enemic principal del nivell en el qual s'estigui jugant.

2.1 Interacció persona-ordinador

La pantalla del dispositiu Windows Phone 7 és multi tàctil. Té suport per a la detecció de fins a 5 contactes amb la pantalla (5 dits a la vegada). També existeixen tres botons físics al telèfon:

- Endarrere
- Inici
- Cercar

Tenint en compte aquestes característiques es defineix la manera d'interactuar amb la interfície.

2.1.1 Interacció amb menús

- El botó endarrere pausa la partida si estem jugant –donant la opció de tornar al menú principal-, en cas de que estiguem navegant pels menús del joc, com ara el d'opcions, torna a la opció de menú anterior menú anterior.
- Si ens trobem a la pantalla principal quan es prem el botó endarrere es fa sortir l'usuari de l'aplicació

- S'interactua amb els elements de menú i botons amb un toc del dit sobre la pantalla tàctil.

2.1.2 Interacció en el joc

Acceleròmetre:

La velocitat en la qual aquesta nau es mou dependrà de la inclinació que doni al mòbil en cada direcció. La sensibilitat de l'acceleròmetre haurà de ser elevada per evitar que el jugador hagi de moure tant el dispositiu que li sigui difícil seguir l'acció en pantalla.

Pantalla tàctil:

> La funcionalitat és la mateixa que amb l'acceleròmetre, només que per moure la nau caldrà arrossegar el dit cap a la direcció cap a la qual vulguem moure la nau. La zona sensible de la pantalla en serà la meitat esquerra. La meitat dreta es reservarà per a futures funcionalitats, com ara l'ús d'armes especials.

En la secció de disseny tècnic s'explicarà més detalladament la implementació d'aquests tipus de control.

2.2 Informació en pantalla

El jugador podrà veure informació en pantalla durant la acció del joc que li informarà del seu estat. Aquesta informació és es veurà a la part superior de la pantalla, i és la següent:

- Energia o escut
- Energia de l'arma especial
- Puntuació
- Vides

La captura de pantalla mostra la informació al jugador durant el joc.

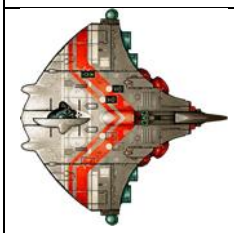
2.1 Naus enemigues

En el joc poden aparèixer diferents tipus de naus enemigues. Cadascuna pot tenir diferents tipus de moviments i armes. Cada cop que es destrueixi una nau s'incrementen els punts del jugador en el valor que tingui la seva energia.

2.1.1 Naus enemigues Nivell 1

Les naus enemigues poden ser destruïdes si el valor del mal en cas d'impacte de la nau del jugador és superior a la energia de que aquestes disposen.

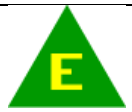
Biratu	
	Nau kamizake sense pilot, que té l'aspecte d'una serra que gira sobre el seu propi eix. Dimensions: Petita Desplaçament: Lineal, rotant sobre el seu propi eix Velocitat: Constant, 5 unitats Danys en cas d'impacte: 1 Energia: 1 Punts generats per destrucció: 3 Armament: no en té
Erasotzaileak	

	Nau de caça, de petites dimensions i poc resistents. Ataquen en petits grups i representen un perill lleu a les naus humanes, excepte per a les d'exploració, pel seu baix nivell d'energia. Tenen la capacitat de disparar pocs projectils en la direcció de la nau del jugador en el moment de disparar. Dimensions: Petita Desplaçament: Lineal Velocitat: Constant, 2 unitats Danys en cas d'impacte: 5 Energia: 10 Punts generats per destrucció: 10 Armament: 1 projectil. Genera danys de 5 unitats.
Erasotzaileak	
	Nau d'atac, de dimensions mitjanes. Tenen bastanta resistència. Acostumen a atacar de manera independent. Tenen la capacitat de disparar diversos projectils a la vegada en diferents direccions en el moment de disparar. Dimensions: mitjana Desplaçament: Lineal Velocitat: Constant, 2 unitats Danys en cas d'impacte: 5 Energia: 20 Punts generats per destrucció: 10 Armament: 5 projectils a la vegada. Generant cada un danys de 5 unitats.

2.2 Objectes

Cada cop que es destrueixi un objecte s'incrementen els punts del jugador en el valor que tingui la seva energia. En els diferents nivells existeixen alguns objectes en comú:

Objecte Energia (escut)

	L'objecte energia es va desplaçant rebotant per la pantalla un cop s'ha destruït el transport que la conté. Per a recollir-la el jugador ha de moure la seva nau damunt de la energia, i aquesta passa a incrementar en 20 unitats la energia del jugador (la energia del jugador mai pot superar el 100%, és a dir, les 100 unitats). Dimensions: Petit Desplaçament: Lineal rebotant per la pantalla, desapareix als 5 rebots. Velocitat: Constant, 2 unitats Danys en cas d'impacte: 0 Energia: 1 Punts generats per destrucció: 0
Objecte Potència de motors	
	La potència de motor té les mateixes característiques i comportament que l'objecte energia, la diferència està en que un cop recollida la potència de motors incrementa fins a un màxim de 10 la velocitat de la nau del jugador.
Objecte Potència de foc	
	La potència de foc té les mateixes característiques i comportament que l'objecte energia, la diferència està en que un cop recollida la potència de foc incrementa fins a un màxim de 10 la potència de foc de la nau del jugador.

2.3 Obstacles

2.3.1 Nivell 1

Meteorit	
	La potència de foc té les mateixes característiques i comportament que l'objecte energia, la diferència està en que un cop recollida la potència de foc incrementa fins a un màxim de 10 la potència de foc de la nau del jugador. Punts generats per destrucció: 10

2.1 Característiques del jugador

El jugador entra al nivell des de la part esquerra de la pantalla (en una petita animació automàtica), i és invulnerable durant tres segons (la nau parpalleja). Això passa al començament del nivell o després de que el jugador hagi perdut una vida.

El joc s'inicia amb un total de 3 vides per al jugador i el 100% d'energia o escut. Es perd un percentatge variable d'energia quan es rep l'impacte de foc enemic, o s'impacta amb objectes, obstacles o naus enemigues.

Quan s'arriba al 0% d'energia es perd una vida, i el jugador torna a disposar del 100% d'energia, i durant 3 segons és invulnerable al foc enemic i els impactes amb objectes, naus i obstacles (el jugador n'és conscient perquè el gràfic que representa la nau parpadeja).

Quan el nivell de l'escut baixa de manera perillosa, un indicador avisa al jugador:

Si s'esgoten totes les vides s'acaba el joc i el jugador és eliminat sense la victòria. Es mostra una pantalla de Game Over.

2.2 Naus

En una versió complerta del joc, s'aniria disposant de naus cada cop més potents a mesura que avancem en el joc, però com que en aquest treball s'entrega una versió de demostració que inclou només el primer nivell, el jugador només podrà utilitzar una nau.

2.2.1 Nivell 1

SpaceTrotter	
	Nau d'exploració. Té una bona velocitat, però poca potència de foc i una energia de resistència també baixa. Dimensions: Petita Desplaçament: Jugador Velocitat: 5 unitats Danys en cas d'impacte: 15 Energia: 20 Armament: fins a 10 nivells de potència, cada nivell incrementa la quantitat de projectils emesos per segon. Cada projectil resta 1 punt d'energia als enemics.

2.2.2 Reacció de les naus als impactes

Quan es produeix una col·lisió entre una nau i un objecte, obstacle, nau o foc enemic, es resta la energia, i si aquesta baixa del 0% es resta una vida, i la energia es posa al 100%. Si no queda cap vida s'acaba el joc.

A més, en cas d'impacte, la nau parpellejarà durant tres segons, i durant aquest temps serà invulnerable a tot tipus d'impactes.

2.3 Nivells del joc

En el TFG s'entregarà una demostració de part d'un sol nivell, però el joc està plantejat i orientat per a tenir més d'un nivell. Es farà una descripció exhaustiva del primer nivell, que és el que es desenvoluparà, i quedarà la continuació oberta a possibles nivells següents.

2.3.1 Nivell 1

1.2.3.1 Sinopsis

Una sèrie d'il·lustracions introdueixen la història: els humans envien naus a l'espai, exploren planetes i creen colònies. En una de les exploracions més distants, una nau exploradora descobreix Kortatu i és atacada.

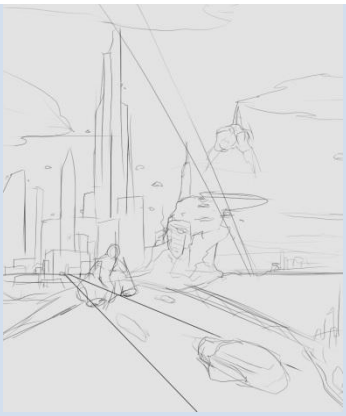
S'inclou un prototip el story board per facilitar la posterior il·lustració d'aquesta història:

Fotograma 1

Text en llegenda:

La raça humana ha sortit a explorar l'espai a la recerca de nous planetes habitables.

Fotograma 2



Text en llegenda:

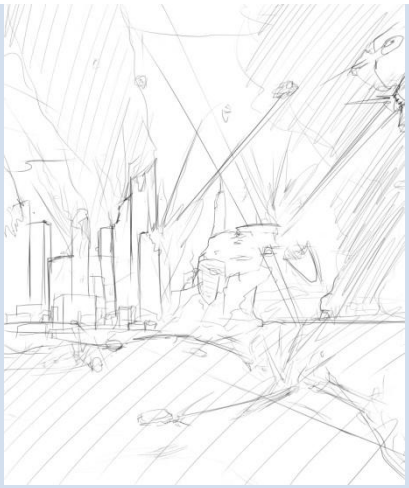
Els humans construeixen nombroses colònies a la Via Làctia, formant una aliança pacífica que s'estén per centenars de planetes en diferents sistemes.

Fotograma 3

Text en llegenda:

Tot semblava anar-los bé als humans, fins que una missió d'exploració liderada per Blake –un dels més condecorats capitans de l'aliança-, descobreix una raça alienígena al sistema Kortatu... la raça Khorza.

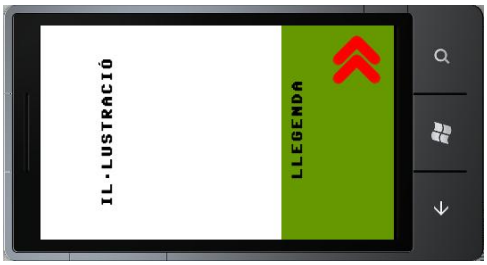
Fotograma 4



Text en llegenda:

...una raça alienígena agressiva, que intentaria arrasar, devorar i aniquilar completament la raça humana. Davant de tal amenaça, els humans necessitaran el valor dels seus millors homes i dones per evitar la extinció de la seva espècie.

El jugador pot passar d'una pantalla a l'altra tocant en un botó de següent, que es representa de forma esquemàtica la distribució de la pantalla en el següent esquema gràfic:

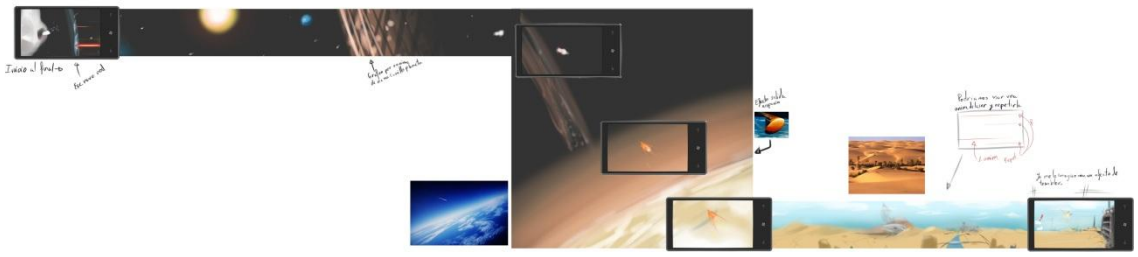


Posteriorment a aquesta seqüència, comença el joc, amb una seqüència programada. El jugador pot controlar la nau del capità Blake, que manté una conversa per ràdio amb la nau de Narbona. Mentre aquesta conversa es porta a lloc, reben un atac. La nau de Narbona és destruïda, i Blake comença la batalla en solitari, sota el control del jugador.

1.2.3.1 Inici del joc

Després de les il·lustracions i textos descrits en el punt anterior s'inicia la part interactiva del videojoc, és a dir que apareix la nau del jugador en pantalla i s'inicia l'acció de fugir i disparar.

S'inclou un esquema del nivell 1:



2.3.1 Nivell 2

Els Korzha han perseguit a Blake i ataquen la nau madrassa on es refugia (Mother). El propi Blake i els seus homes defensen la nau canons en mà.

En aquest nivell es controla un canó en vista subjectiva i diferents onades d'enemics s'aproparan a la càmera. El jugador ha d'eliminar tots els enemics que pugui.

2.3.1 Nivell 3

En aquest nivell l'objectiu és rescatar a Narbona, de manera que Blake armarà la seva nau i contraatacarà a Kortatu en una missió un tant suïcida per la diferència en nombre.

El nivell tindrà un boss final, o nau madrassa que caldrà eliminar. Atacarà massivament al jugador i aquest haurà de tenir la habilitat suficient per evitar-ne els atacs.

2.3.1 Nivell 4

Blake s'introdueix a la nau madrassa, on rebrà tot tipus d'atacs de làsers provinents de les parets i formacions de naus enemigues.

2.3.1 Nivell 5

Còmic interactiu en que Blake arriba al hangar de la nau madrassa, baia de la seva nau, pistola en mà per rescatar a Arbona. En el còmic interactiu el jugador haurà d'efectuar diversos moviments amb el dit sobre la pantalla tàctil per a prosseguir amb la acció. Blake acaba rescatant a Arbona i una gran descendència els espera...

CAPÍTOL 3. Disseny tècnic

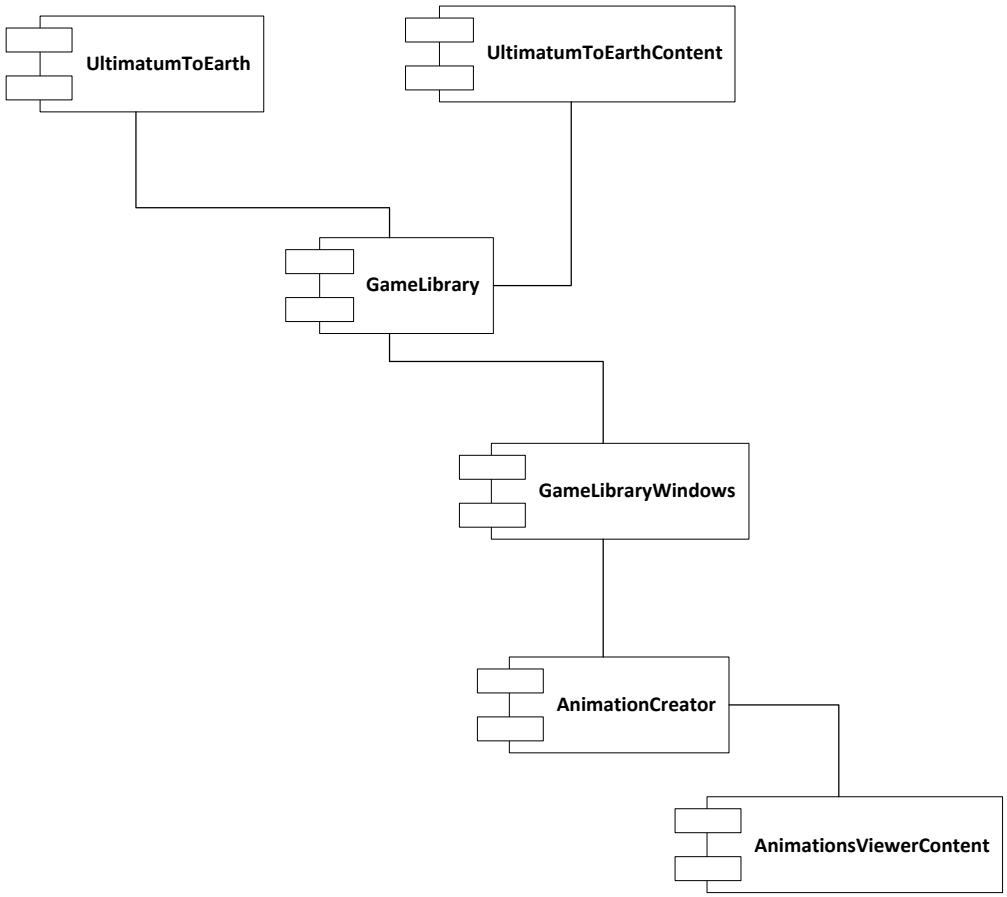
3.1 Introducció: Particularitats del entorn

A l'hora de dissenyar la solució tècnica que permetrà implementar el videojoc, s'han tingut en compte les particularitats de l'entorn, molt importants, que són:

- Hardware del dispositiu. Es tracta d'un mòbil, és per tant un dispositiu amb limitacions, i s'ha de tenir cura a l'hora d'escollir algoritmes. La memòria no es pot malgastar.
- Gestió de la memòria, que pot ser problemàtica. Per a desenvolupar jocs per a Windows Phone 7 s'utilitza C#, sobre una versió adaptada de .Net Compact Framework. C# és un llenguatge de memòria manejada. Un llenguatge de memòria manejada es diferencia dels de memòria no manejada, en que el programador no és qui s'encarrega de la gestió de memòria. Per exemple, en C++ el programador té el control "absolut" de la memòria. Cada cop que es destrueix un objecte aquesta memòria és alliberada. En C# és el .Net Framework qui s'encarrega de gestionar la memòria. El programador en pot ser partícip, però amb limitacions.
  En .Net la gestió de la memòria és en certa manera transparent. El programador va creant instàncies d'objectes. Aquestes instàncies en algun moment o bé son substituïdes, o bé es deixen d'utilitzar. No obstant, en C# la memòria segueix sent ocupada per aquestes instàncies. Aquí entra en joc el que s'anomena recollidor de brossa o "Garbage Collector". El recollidor de brossa s'executa, per defecte, de manera automàtica, quan la brossa arriba a un límit determinat. En el cas de Windows Phone 7, això passa cada cop que arribem a generar 1MB de brossa. I en un videojoc, si no és te cura, és fàcil generar aquesta i molta més brossa.
  Per afrontar aquesta situació s'ha seguit la estratègia següent:
  - Cridar manualment al recollidor de brossa després de que s'hagi carregat el nivell (quan encara apareix la pantalla de "carregant").
  - En la inicialització del nivell, inicialitzar els arrays que aquest contingui amb un límit d'elements. Es a dir, si per exemple tenim un array de naus enemigues, aquest serà d'una mida fixa, i no es modificarà dinàmicament, per evitar posicionaments dels objectes en memòria, amb la conseqüent generació de brossa.

3.2 Diagrama de components

El següent diagrama descriu els principals components que formen el joc. El diagrama dona una visió a alt nivell de l'arquitectura de la aplicació. L'objectiu ha estat obtenir una arquitectura desacoblada però dins els límits pràctics. S'observarà que tota la notació utilitzada és en llengua anglesa.



A continuació es descriuen més detingudament aquests components:

- **GameLibrary:** És el cor de l'aplicació. Conté classes genèriques (que es descriuran en els punts següents del disseny). Aquestes classes queden compilades com un sol ensamblat (dll), i són utilitzades, heretades o implementades per altres aplicacions. L'avantatge d'utilitzar aquest ensamblat genèric és que aquest pot ser reutilitzat de manera senzilla per a crear nous videojocs, o bé extensions d'aquest.

- **UltimatumToEarth:** Conté el joc en sí mateix, les classes que representen els nivells, les naus, els enemics, les bales... Fa referencia al ensamblat GameLibrary, del que fa un ús extensiu.

- **UltimatumToEarthContent:** Conté el contingut que no és programari del videojoc: gràfics, vídeos, fonts, animacions, sons i cançons. Té l'estructura següent:
    - Animations
    - Audio
        - Music
        - Sound
    - Fonts
    - Textures
        - Epilogues
            - Epilogue1
        - Levels
            - Level1
        - Loading
        - Screens
    - Videos

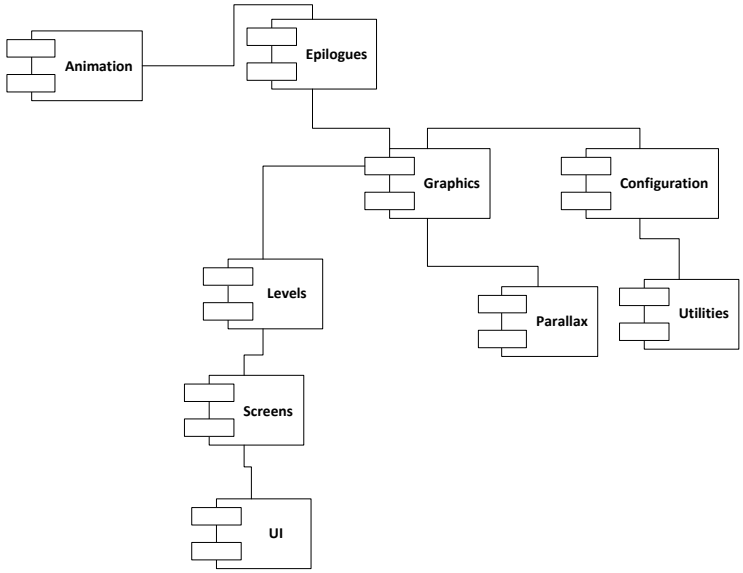
- **GameLibraryWindows:** En realitat conté el mateix codi que GameLibrary. El Microsoft Visual Studio (que és la eina de desenvolupament utilitzada per realitzar el videojoc) ens obliga a crear una copia mirall del ensamblament que serà específica per a eaplicacions tipus Windows (l'anterior era específica d'aplicacions de telèfon).

- **AnimationCreator:** El joc conté animacions, que poden estar composades de imatges i texts, amb animacions basades en interpolacions de posició, escalat, fade in, fade out, vibracions, etc. Aquesta és una aplicació de Windows que permet generar aquestes animacions i mostrar-les en temps real en un emulador del Runtime de XNA per a mòbils però que funciona en entorn Windows. Per això és necessària la existència de l'ensamblament "GameLibraryWindows". Finalment aquesta aplicació permet emmagatzemar les animacions en format XML, aquests fitxers poden ser interpretats per un seguit de classes, que veurem més endavant, dins de Game Library que facilitaran el procés de creació d'aquestes animacions.

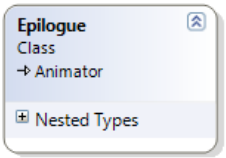
- **AnimationsViewerContent:** Contenidor de recursos para generar la animació, té la mateixa estructura que UltimatumToEarthContent.

3.1 GameLibrary

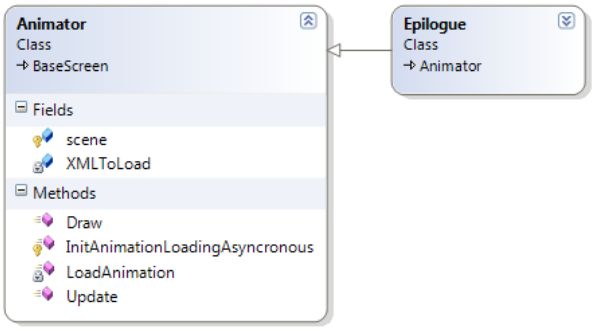
Ja s'ha comentat que aquesta llibreria és el centre de l'aplicació, així doncs entraré en especial detall definint-ne els seus components. No obstant, donat que aquests tenen una complexitat relativament alta, es mostraran primer a alt nivell, i s'aniran detallant progressivament al llarg d'aquest document.



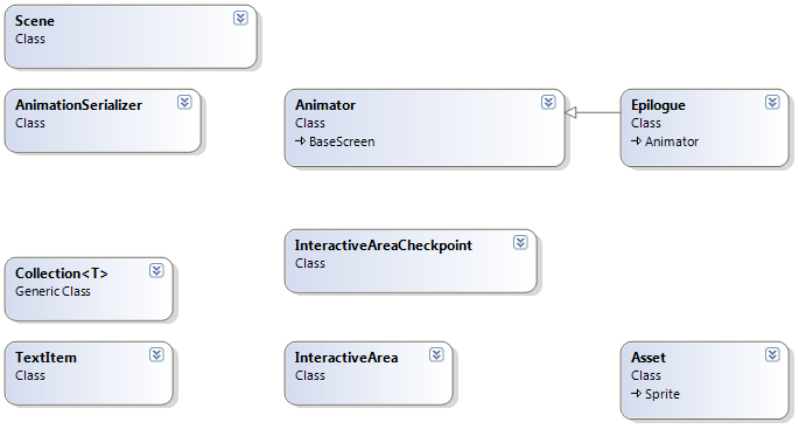
- **Epilogues:** Conté el codi necessari per a reproduir una animació creada amb l'aplicació descrita anteriorment: AnimationCreator.



- **Animation:** És la animació que serà reproduïda per les classes que heretin de Epilogues.



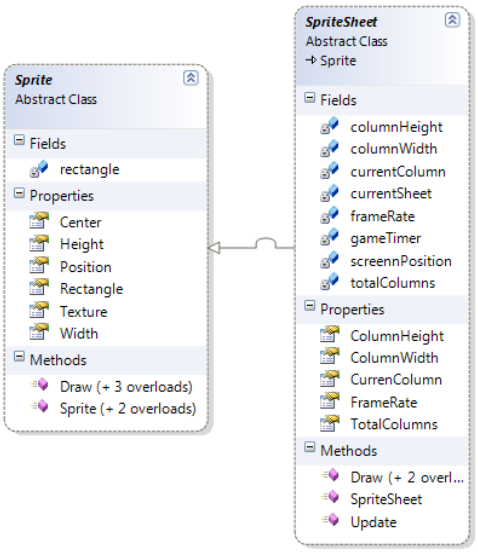
El diagrama mostra que el objecte Animator conté els mètodes necessaris per a carregar una animació a partir d'un XML, així com per dibuixar-la. La propietat scene representa tot el contingut de la animació.



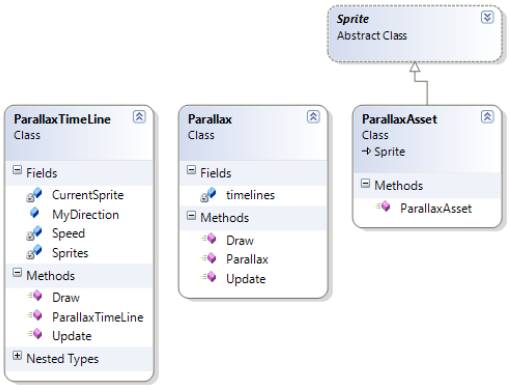
El diagrama mostra que la escena consta d'una animació, que fa ús d'un objecte de serialització i deserialització anomenat AnimationSerializer. Aquest procés és imprescindible per a carregar les animacions generades amb la aplicació AnimationCreator.

El objecte Collection és també interessant perquè representa una col·lecció d'objectes sense especificar-ne el tipus. Aquestes col·leccions podran ser d'elements de text (TextItem), o textures (Asset). Pel que fa a les àrees interactives, només n'hi ha una per escena.

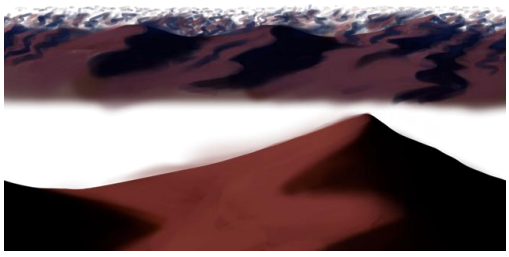
- **Graphics:** Conté funcionalitats gràfiques per renderitzar textures per pantalla, així com un objecte anomenat GraphicsDevice que representa les característiques i configuració del dispositiu gràfic a nivell de hardware del mòbil.



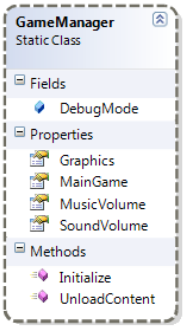
- **Parallax:** S'encarrega de pintar fons. Aquests fons poden ser fons de nivells, de menús, etc. Pot ser útil per a renderitzar el fons d'un nivell, el fons consisteix en decorat gràfic.



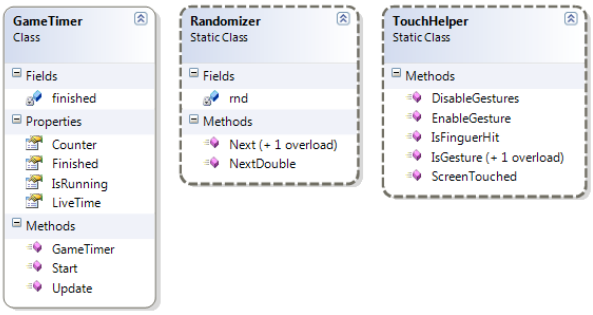
En la il·lustració es mostra, de manera esquematitzada, el concepte. El adequat posicionament per capes de diverses imatges, pot acabar creant composicions complexes.



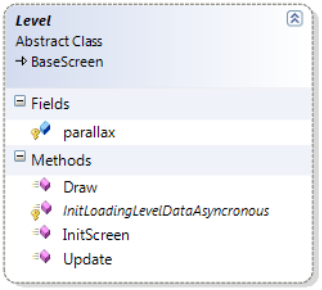
- **Configuration:** Conté elements que s'utilitzen des de la majoria de llocs del joc, com el volum del so, volum de la música, i també s'encarrega d'inicialitzar els gràfics del joc.



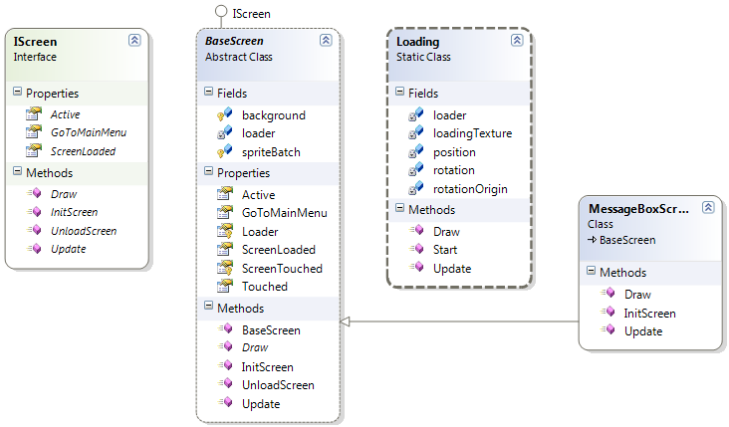
- **Utilities:** Classes genèriques que poden tenir petits usos repetitius en tota la aplicació, com per exemple:
    - GameTimer: Un temporitzador. Pot ser útil si volem que passi un temps entre determinades accions. Per exemple, si volem que al fer click en un botó aquest es vegi d'un color determinat que remarqui que ha estat pitjat durant uns quants mili segons, aquesta classe se n'encarrega de la gestió del temps.
    - Randomizer: En molts punts d'un videojoc es necessita generar números aleatoris. Aquest objecte se n'encarrega.
    - TouchHelper: Un objecte que exposa la interacció multi tàctil (fins a cinc dits simultàniament) del usuari amb el dispositiu mòbil.



- **Levels:** Objecte que té funcionalitats comunes en tots els nivells, com ara el seu renderitzat per pantalla, la execució de la lògica de tots els seus objectes, etc.

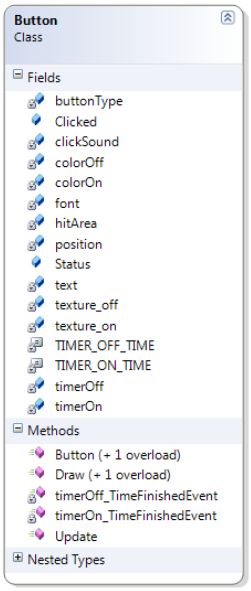


- **Screens:** Conté els objectes necessaris per a gestionar les diferents pantalles del joc, els seus estats i la interacció i transició entre elles. Uns exemples de pantalles serien la pantalla de crèdits, menú principal, menú d'opcions, etc.



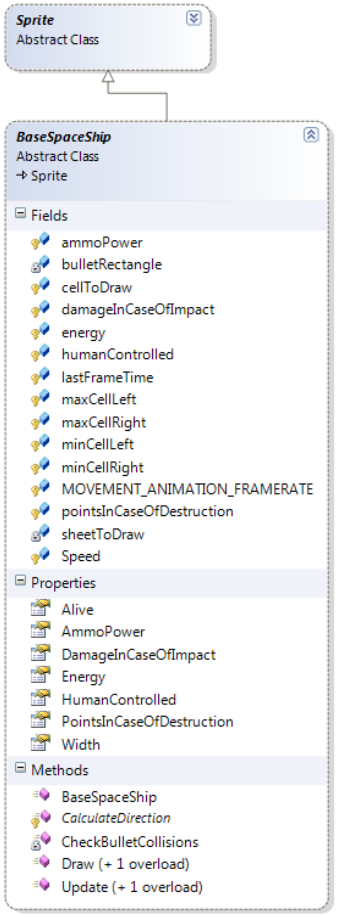
En el diagrama anterior cal recalcar l'ús de tècniques polimòrfiques per a la representació de pantalles ("screens"), de manera que una classe podria llençar una finestra sense conèixer quina és exactament aquesta finestra. Això es demostrarà molt útil al llarg d'aquesta documentació tècnica.

- **UI:** Representa controls d'interacció persona-màquina, com per exemple botons, scrolls, llistes, caixes d'entrada de text, etc.



3.1.1 Les naus

En un joc de naus, òbviament s'ha de representar l'objecte nau d'alguna manera. Si a més es vol que hi puguin haver naus de molts tipus sense que això suposi un gran esforç de desenvolupament, cal anar amb cura alhora de fer-ne el disseny. En aquest cas he escollit una arquitectura basada en herència. Totes les naus del joc hereten de la classe abstracta BaseSpaceShip.



Aquesta classe abstracta (que vol dir que s'ha d'heretar obligatòriament) conté informació comú en totes les naus. Destacant-ne algunes:

- Alive: Indica si la nau està "viva", i per tant s'executarà la lògica d'actualització (Update) i dibuixat (Draw).

- CalculateDirection: És un mètode abstracte, i que per tan serà implementat per cada nau, que en descriu el moviment. Algunes naus tindran trajectòries lineals, d'altres tindran trajectòries curvilínies, etc. Aquest mètode s'encarrega de descriure la trajectòria corresponent.

- Speed: Velocitat a la que s'anirà desplaçant la nau. És un vector de dues dimensions (X,Y).

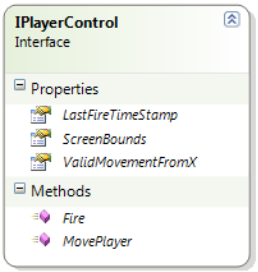
- BulletRectangle: Contè una referència a la textura a renderitzar corresponent a l'armament que dispara la nau.

- MaxCellLeft, MaxCellRigt, MinCellRight, MinCellLeft: Aquestes propietats descriuen la cel·la del spritesheet de la nau en qüestió que correspon a cada posició dins del temps quan la nau descriu un moviment. Per exemple, si estem desplaçant la nau cap a la direcció Y en positiu, i la posició inicial és la normal, posicionarem la cel·la a renderitzar del spritesheet a MinCellLeft, i aquesta s'anirà incrementant, si el moviment prossegueix, fins arribar a MaxCellLeft. El següent diagrama simplificat ho exemplifica:

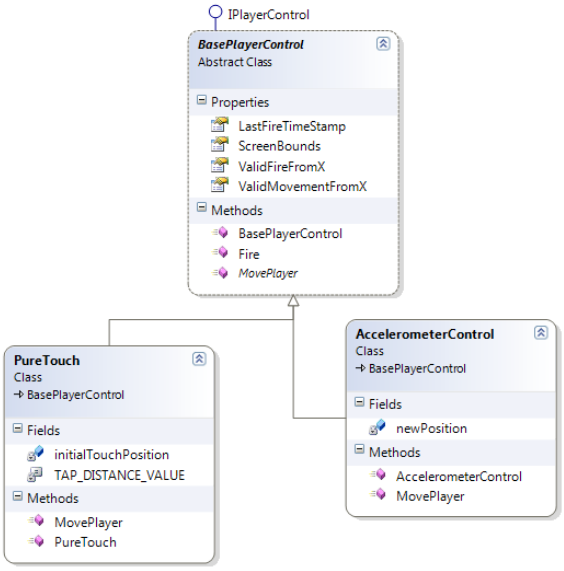
3.1.1 Control de la nau del jugador

Ja s'ha dit que la nau podrà ser controlada per acceleròmetre o per pantalla tàctil. Aquí es defineix el disseny d'aquestes funcionalitats.

Per moure la nau, primerament es defineix una interfície anomenada IPlayerControl, aquesta interfície ens permetrà l'ús de tècniques de polimorfisme per a instanciar el tipus de moviment que vulguem, sempre que es basi en aquesta interfície.



A més, com que tots els controls comparteixen determinades funcionalitats, heretaran tots ells de BasePlayerControl. En aquest cas es mostra aquesta classe base i les classes que n'hereten (PureTouch pel mètode de moviment tàctil i AccelerometerControl pel mètode basat en l'acceleròmetre respectivament). El següent diagrama ho il·lustra:



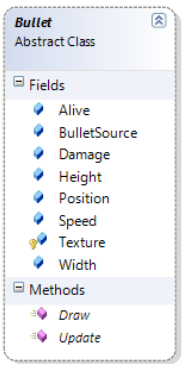
3.1.2 Detecció de col·lisions

En un videojoc la detecció de col·lisions és vital, i més encara en un shooter com el que tenim entre mans, on els projectils són la base del a jugabilitat. La detecció de col·lisions inclou:

- Detectar quan una nau rep l'impacte de l'armament enemic, i per tant s'ha de restar energia. Si la energia és inferior a zero i no hi ha més vides, la nau explota.
- Detectar col·lisions entre naus, el resultat de la qual resta energia en base a la propietat "damageInCaseOfImpact" que tenen totes les naus.
- Recollir objectes de l'escenari (vides, punts, armament, etc).

En aquest sentit, destacar dues coses:

- Cada nivell té un array de Bullets, o bales, que contenen informació de les bales que hi ha "vives" o no al joc. Les que estan vives seran les que es comprovarà si xoquen amb l'entorn.

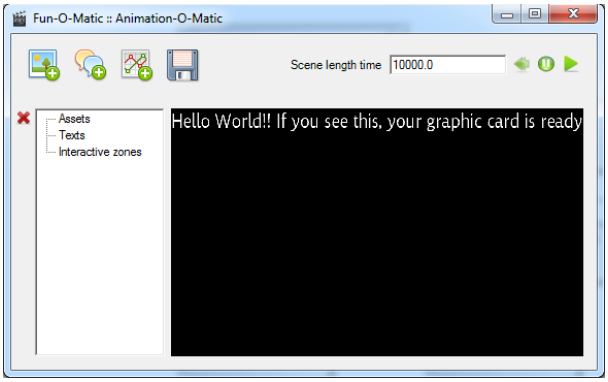


El mètode CheckBulletCollision de la classe base de les naus, que hem vist anteriorment, s'encarrega de recórrer l'array i testejar les possibles col·lisions entre cada nau i les bales. Si la bala l'ha disparat una màquina, només es comprovaran les col·lisions amb la nau humana, i si per contra, la bala l'ha disparat una nau humana, les col·lisions només es testegen sobre les naus enemigues.

3.1 Animations creator

Ja he parlat una mica en els punts anteriors del creador d'animacions. No obstant la seva importància dins del projecte es mereix majors explicacions. Aquesta eina permet crear animacions, que poden ser interactives o no, a partir d'elements multimèdia: text, imatge, so i música. I el que és més important, permet que qualsevol persona sigui capaç de crear aquestes animacions, sense coneixements de programació (la qual cosa es tradueix en un estalvi de recursos qualificats i creació ràpida d'animacions).

Aquest és l'aspecte de la pantalla principal de la eina de creació d'animacions:



S'hi pot observar un menú:

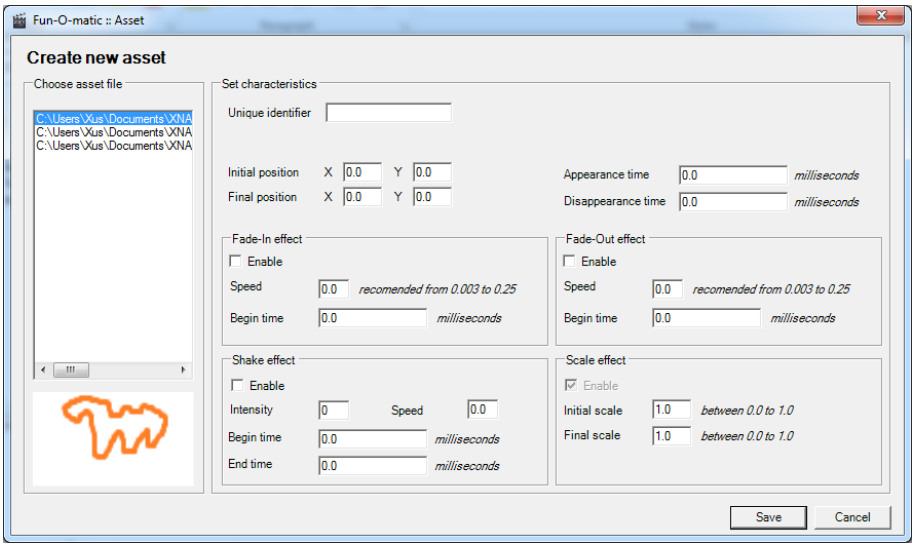
- Nou asset: Permet afegir gràfics a la animació
- Nou text: Permet afegir texts a la animació
- Nova zona interactiva: Permet afegir una zona interactiva a la animació

També hi ha altres elements a la interfície:

- Longitud de la escena (en milisegons)
- Rebobinar
- Pausa

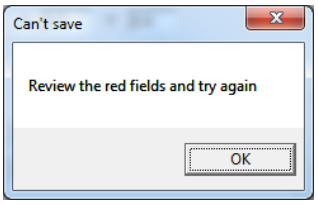
- Play (o continuar després de pausa)
- Llista d'elements afegits a la animació (banda esquerra), que són eliminables i editables.
- Zona de preview (quadre de color negre). En aquesta zona el creador de la animació podrà previsualitzar, en temps real, el resultat de la animació que està creant, tal com es veuria en el mòbil. La única diferència és les majors dimensions de la pantalla, però la resolució i la proporcionalitat són les que existeixen a Windows Phone 7. Aquest preview pot ser pausat, rebobinat, etc.

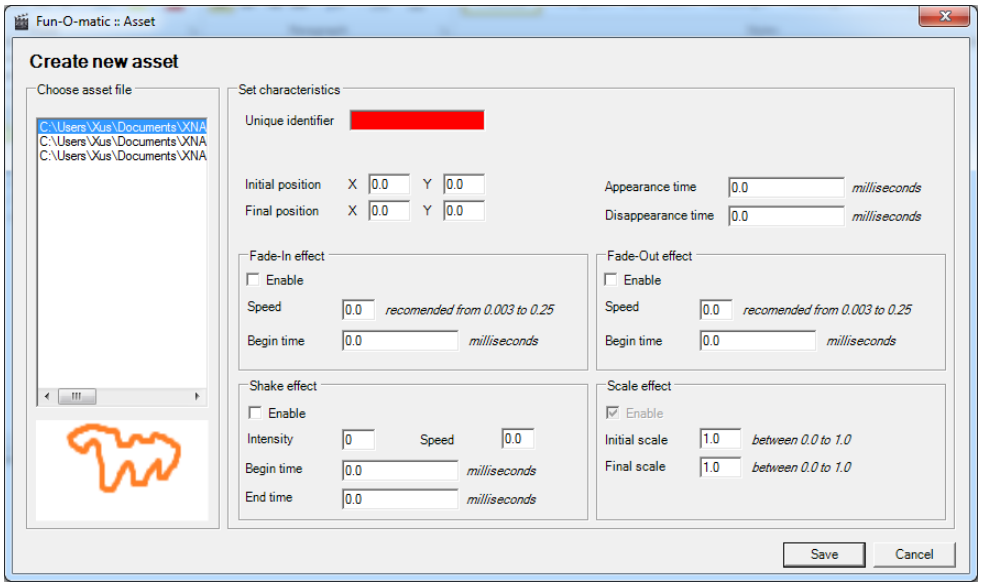
La pantalla següent mostra la informació que es sol·licita quan volem afegir un nou asset a la animació:



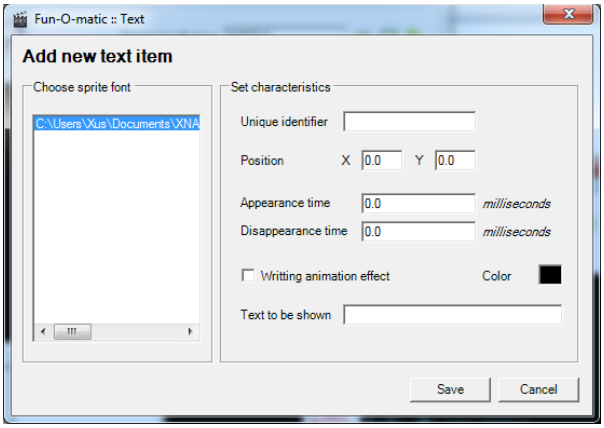
Gran part d'aquesta informació és opcional (la que està dins dels quadres, que consisteixen en diferents efectes). Cada asset ha de tenir un identificador únic, una posició inicial, final (que de ser diferents produiran una interpolació lineal), moment d'aparició i desaparició dins la línia del temps de la animació principal. A la banda esquerra de la pantalla podem escollir els gràfics que tenim disponibles i previsualitzar-los.

En el cas de no completar la informació de forma correcta, el formulari és validat automàticament i s'informa al usuari de la necessitat de corretgir la informació introduïda:



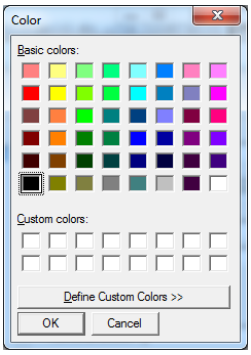


Afegir un text a la animació és una operació similar:

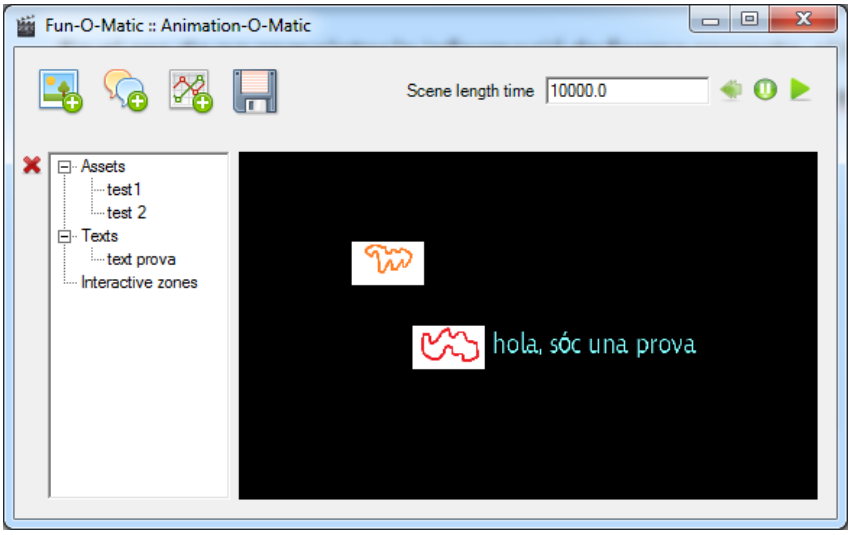


Com es pot veure a la figura, podem escollir la font de les que tinguem carregades al projecte, i al igual que abans, definir les característiques del text. Entre elles destaquen que podem escollir-ne el color, i posar-li un efecte d'escriptura (les lletres van apareixent progressivament com si estiguessin escrites per algú).

En aquest diàleg veiem com es pot seleccionar el color en que volem el text:



Aquest seria el resultat d'una animació de prova –extremadament simple però correcta com a exemple-.

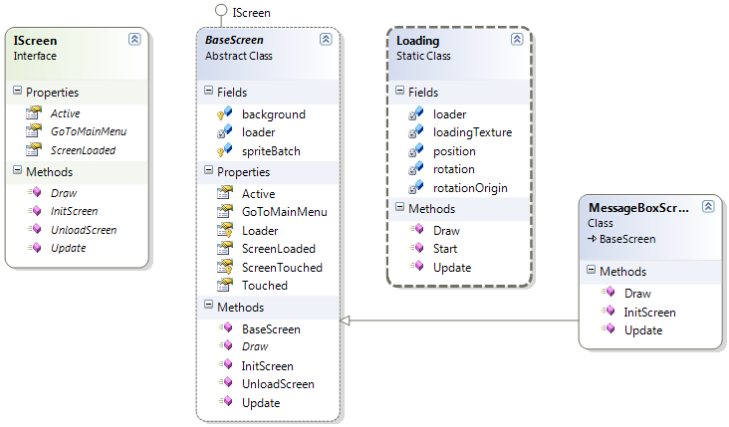


També es pot veure un exemple de la aplicació en funcionament en aquest vídeo que he pujat al Youtube: http://www.youtube.com/watch?v=Mk9y8L0wj5w

3.1 Gestió de penatalles (Screens)

Una bona gestió de les pantalles i la transició entre elles no és trivial en un videojoc. I la seva correcta implementació pot arribar a influir molt negativament en la mantenibilitat del joc, i acabar repercutint en el cost de desenvolupament de cadascuna d'aquestes pantalles. Per aquest motiu, en Ultimatum to Earth se li ha donat una importància molt elevada.

Anteriorment ja s'ha parlat del paquet Screens, que il·lustra la figura següent:



Les classes que conté són molt importants, existirà una classe estàtica anomenada ScreenManager que gestionarà totes les pantalles i s'encarregarà d'enviar a la GPU per a ser renderitzada la pantalla actual. Amb un bon disseny, basat en polimorfisme, canviar d'una pantalla a l'altra serà tan senzill com escriure una línia de codi:

```
LaunchScreen(new MainMenuScreen());
```

En aquest cas estem obrint la pantalla anomenada MainMenuScreen. Gràcies a la herència totes les pantalles que vulguem podran fer ús de la classe Loading, que s'encarrega de pintar una pantalla que indiqui que s'està carregant informació. Una pantalla que doni aquesta informació és necessària per evitar que l'usuari tingui la sensació que la aplicació no respon, s'ha penjat, o que és lenta. Un simple "carregant" serà suficient per dissipar els dubtes de l'usuari.

En aquest sentit, totes les pantalles, que hereden de BaseScreen (i aquesta, al seu temps, implementa la interfície IScreen) tenen la propietat ScreenLoaded. Per a que es pugui renderitzar la pantalla de "Loading", fins i tot amb alguna animació, mentre s'estan carregant textures, sons, fitxers XML, etc, hem d'utilitzar tècniques de programació paral·lela –encara que molt senzilles-. Windows Phone 7 suporta múltiples fils d'execució, en aquest sentit la solució adoptada consisteix en llençar un nou fil que carregui la informació necessària, quan aquest acabi informa al Loader i aquest deixa de pintar-se, mostrant ja la pantalla a la qual volíem accedir. El següent en seria un exemple molt simplificat, però que deixa clara la idea:

```
protected override void InitLoadingLevelDataAsyncronous()
{
    Loading.Start();

    ScreenLoaded = false;
    Threading.ThreadStart starter = new System.Threading.ThreadStart(LoadContent);
```

```
        Threading.Thread thread = new System.Threading.Thread(starter);

        thread.Start();
    }

    private void LoadContent()
    {
         System.Threading.Thread.Sleep(10000);
         ScreenLoaded = true;
    }
```

En aquest cas es pot veure com informem a Loading que estem iniciant la càrrega (i per tant s'haurà de renderitzar ell mateix enlloc de la pantalla que estem inicialitzant, i en el cas que apliqui s'inicialitzaran les seves pròpies animacions i gràfics).

Dins "LoadContent" senzillament estem aturant la execució durant 10.000 milisegons (10 segons) com a prova demostrativa del funcionament del sistema. Finalment informem de que la càrrega ha finalitzat.

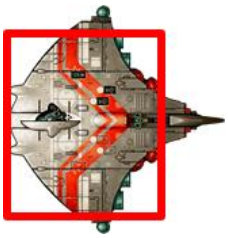
A partir d'aquí podem veure com les pantalles renderitzaràn la pantalla de Loading o bé la pantalla actual, depenent de si aquesta s'està carregant o si per contra, ja s'ha acabat de carregar. És un sistema ràpid i senzill per aplicar la pantalla de Loading a qualsevol pantalla existent amb l'objectiu de millorar la experiència d'usuari.

```
    public override void Draw(GameTime gameTime)
    {
      spriteBatch.Begin();

      if (ScreenLoaded)
      {
          GameManager.Graphics.GraphicsDevice.Clear(Color.Black);
          parallax.Draw(gameTime, spriteBatch);
      }
      else
      {
          Loading.Draw(spriteBatch);
      }

      spriteBatch.End();
    }
```

CAPÍTOL 4. Proves

En aquest projecte s'han realitzat dos tipus de proves:

- Proves realitzades per l'autor del TFG amb l'objectiu de la depuració d'errors.
  En aquestes proves s'ha fet especial incidència a la detecció de col·lisions. En aquest sentit ha estat necessari introduir petites modificacions al programa. Aquestes modificacions consisteixen en renderitzar per pantalla el objecte sobre el qual es testeja una col·lisió –un rectangle-. La col·lisió no es calcula sobre la dimensió total de la textura que pugui tenir un objecte (per exemple una nau). La següent imatge il·lustra la idea:



  Això es fa per donar un major realisme al joc.
  Així doncs, és evident que si la detecció de col·lisions no s'aplica sobre la dimensió real de la textura, és fàcil que hi hagin errors. I depurar els errors d'una aplicació de temps real, i en aquest cas un videojoc, pot esdevenir una tasca molt complexa. Per això la importància de la capacitat de depurar visualment els possibles errors, en aquest cas dibuixant els rectangles, com ja s'ha dit.

  La captura de pantalla següent mostra aquest renderitzat dels rectangles. És útil perquè ajuda a detectar problemes quan els projectils impacten amb les naus o hi ha col·lisions entre naus. Un problema típic –i que he trobat i he pogut solucionar amb aquesta tècnica- és que les naus explotin quan el projectil amb el que han xocat està massa lluny. El problema és que el rectangle, que existeix de manera lògica però no visual, no es troba on creiem que hauria d'estar. Renderitzar aquests rectangles m'ha ajudat a detectar i corretgir el problema.

- Proves realitzades per usuaris, amb la observació del autor del TFG, amb l'objectiu d'avaluar la usabilitat.

  És molt important que els usuaris provin el joc, sobre la atenta però discreta observació del autor. Això permet detectar errors, coses que l'autor pot creure òbvies, però que el jugador no entén. O a la inversa, l'usuari detecta que hem posat botons en pantalla que potser no són necessaris.

  Un exemple de millora que he pogut realitzar gràcies a aquesta observació és que a la pantalla d'opcions no és necessari un botó "guardar canvis". En el seu lloc, la majoria de jugadors de prova entenen que després d'aplicar els canvis desitjats, es pot prémer el botó "endarrera". Aquest botó l'inclouen obligatòriament tots els mòbils que es basin en el sistema operatiu Windows Phone 7. La imatge següent mostra aquesta situació:

CAPÍTOL 5. Conclusions

Desenvolupar un videojoc és una tasca d'alta complexitat que requereix una gran disparitat de coneixements relacionats amb el món multimèdia: des de grafisme a programació, passant per la creació d'arguments i guions, disseny de personatges, i creació d'una història que permeti al joc ser alguna cosa més que una munió de gràfics en moviment.

En un videojoc professional doncs es requereixen molts recursos si es vol arribar a bon port. Aquest treball és una petita representació del que un projecte d'aquestes característiques podria semblar.

Els mòbils acostumen a tenir uns jocs més simples dels que podem trobar en les videoconsoles tradicionals. Aquestes màquines tenen jocs en el seu catàleg que basen la seva qualitat en els seus gràfics. Encara que també contenen fortes línies argumental, aquests jocs no són res sense unes textures extremadament detallades, i una il·luminació i una física que pràcticament sembli que simuli la realitat. En canvi els smartphones tenen jocs que, donats els recursos limitats del dispositiu, necessàriament s'han de basar més en la innovació, i no tant en els gràfics.

Tot plegat obra la porta a equips de desenvolupament relativament petits, que poden arribar a competir amb empreses molt més grans, si el seu equip té la qualitat i creativitat suficients.

D'aquesta manera, doncs, estimo que s'han complert els objectius plantejats inicialment en aquest treball. Però el camí no acaba aquí, aquest projecte m'ha ensenyat moltes coses i és possible que Ultimatum to Earth acabi veient la llum al Marketplace algun dia.

CAPÍTOL 6. Glossari

Array

Terme de programació. És un objecte que conté una llista delimitada d'objectes emmagatzemats en memòria.

Acceleròmetre

És un aparell que permet mesurar l'acceleració aplicada sobre un objecte, i mesurar-ne les conseqüències. En el cas que ens ocupa, els smartphones d'última generació incorporen tots ells dispositius d'aquest tipus.

Briefing

És un resum de requeriments que pot tenir el desenvolupament d'una aplicació, o un entregable multimèdia. Pot contenir apunts dels requeriments que s'han recollit mantenint entrevistes amb un possible client del producte que s'està desenvolupant.

C#

És un llenguatge de programació desenvolupat per Microsoft com a llenguatge de programació que forma part del .Net Framework. El podríem considerar multi-paradigma, ja que és orientat a objectes, funcional, genèric, imperatiu i declaratiu.

Detecció de col·lisions

En termes de computació, i més concretament, del desenvolupament de videojocs, la detecció de col·lisions fa referència a la resolució del problema computacional de detectar la intersecció entre dos objectes.

Marketplace

És el nom que es dona als punts de venda de les aplicacions que es desenvolupen per als Smartphones d'última generació (iPhone i Windows Phone, entre altres). En aquests punts de venda els usuaris poden cercar i comprar les aplicacions i jocs disponibles. Qualsevol programador pot publicar les seves aplicacions en aquests Marketplaces, a canvi de pagar una quota d'entrada.

Geolocalització assistida

Consisteix en utilitzar diverses tecnologies, de forma paral·lela, i en el menor temps possible, per a localitzar geogràficament la posició d'un dispositiu mòbil. Per exemple, el sistema operatiu Windows Phone 7 utilitza una conjunció d'informació de les xarxes Wi-fi que té al voltant, triangulació de la posició mitjançant les antenes de comunicació dels teleoperadors, i GPS. Aquesta tecnologia combinada té raó de ser perquè el GPS per si sol pot resultar molt lent (cal que el dispositiu busqui i trobi el satèl·lit), i no funciona correctament en l'interior dels edificis o sota terra (on no arriba la cobertura dels satèl·lits).

iPhone

És un dels primers dispositius que va combinar les tecnologies multimèdia d'última generació amb un smartphone. Tot plegat combinat amb una pantalla multi-tàctil d'alta sensibilitat. És un producte fabricat per l'empresa nord americana Apple. Actualment hi ha fins a quatre generacions d'aquest mòbil, per la qual cosa és el que té més recorregut històric.

Objective C

És el llenguatge de programació que s'utilitza per a desenvolupar aplicatius per als sistemes d'Apple.

Pantalla tàctil o multi-tàctil

És un dispositiu electrònic de visualització que pot detectar la presència i la localització del contacte d'un dit, quan parlem d'una pantalla tàctil, o de diversos dits alhora, quan parlem de pantalles multi-tàctils.

Render

Terme tècnic que prové de l'anglès , que consisteix en la generació d'un gràfic a partir d'estructures de dades processades per un ordinador.

Smartphone

Es consideren smartphones els dispositius mòbils de màxima capacitat computacional, i connectivitat.

Space shooter

És un sub-gènere dels shooters. Un shooter és un terme anglès, que prové del gènere d'acció. Aquest tipus de jocs basen la interacció amb l'usuari en l'habilitat i els reflexos d'aquests.

Sprite

Un sprite, en termes de computació gràfica, és una imatge en dues dimensions, que s'ha d'integrar en una escena més grant.

Sprite sheet

Un sprite sheet és un fitxer que conté nombrosos sprites. Per a la tarja gràfica –dispositiu electrònic encarregat de processar els gràfics- és més senzill carregar una sola imatge i separar-ne les parts en memòria, que no pas carregar molts gràfics petits. D'aquí a que molts cops els sprites s'unifiquin en una sola imatge.

Gràfics en temps real

És el conjunt de tècniques de computació gràfica que permeten la generació i anàlisi d'imatges en temps real.

UML

Sigles provinents de Unified Modeling Language. És un llenguatge de propòsit general utilitzat en l'enginyeria del software. Inclou una sèrie de notacions gràfiques que permeten representar fàcilment sistemes computacionals complexos.

Windows Phone 7

És el sistema operatiu per a telèfons mòbils d'última generació desenvolupat per l'empresa nord americana Microsoft. És el successor de Windows Mobile, encara que ha estat reescrit completament. Com a novetats ofereix una interfície d'usuari molt més simple i usable que els seus competidors. Un altra diferencia competitiva és que totes les aplicacions i videojocs es poden provar abans de comprar-los. També destaca la seva integració amb Office i les xarxes socials.

El sistema operatiu va ser presentat a Barcelona el febrer de 2010, en el marc del World Mobile Congress, i llençat a la venda a Europa el 21 novembre de 2010.

XNA

És un conjunt d'eines desenvolupades per Microsoft, orientades a la programació de videojocs. Intenta evitar que els programadors hagin d'ocupar-se de tasques repetitives, i permet que aquests es centrin en el que és verdaderament important, en l'essència dels seus jocs.

.Net Compact Framework

És una versió reduïda del .Net Framework. Plataforma desenvolupada per Microsoft per a permetre la execució d'aplicacions basades en C# als dispositius mòbils.

CAPÍTOL 7. Bibliografia

- *Game Development Essentials*, Jeannie Novak.
  Thomson Delmar Learning, 2008. ISBN 978-1-418-04208-0

- *Microsoft XNA Game Studio Creator's Guide*, Stephen Cawood i Pat McGee.
  Mc Graw Hill, 2009. ISBN 978-0-07-161406-1

- *Physics for Game Programmers*, Grant Palmer.
  Apresss, 2005. ISBN 1-59059-472-X

- *Learning XNA 3.0,* Aaron Reed
  O'Reilly, 2009. ISBN 978-0-596-52195-0

- *Programming Game AI by Example,* Mat Buckland
  Wordware Publishing, Inc, 2005. ISBN 1-55622-078-2

- *Artificial Intelligence for Games,* Ian Millington
  Morgan Kaufmann, 2006. ISBN 978-0-12-497782-2

- *XNA 3.0 Game Programming Recipes,* Riemer Grootjans
  Apress, 2009. ISBN 978-1-4302-1855-5

- *Mathematics and Physics for Programmers,* Danny Kodicek
  Charles River Media, 2005. ISBN 978-1-58450-330-9

- *Game Graphics Programming,* Allen Sherrod
  Sherrod, 2008. ISBN 978-1-58450-516-7

- *Game Engine Architecture,* Jason Gregory
  AK Peters, 2009. ISBN 978-1-56881-413-1