Índex

Índex de figures

Introducció

El món dels videojocs està lligat al segle XXI. El sector dels videojocs va començar als anys 70 i es podria afirmar que es va començar a identificar com a tal l'any 1971, quan Nolan Bushnell y Ted Dabney van fundar Atari Computers i van llançar al mercat el primer videojocs oficial i comercial.

A partir d'aquest moment, la industria es va començar a desenvolupar i les màquines recreatives, i més tard, les consoles domèstiques i portàtils, varen començar a aparèixer i establint-se en el mercat.

El món dels videojocs és un dels sectors en la industria que més capital general en l'àmbit de l'oci, amb uns ingressos que superen als generats per al cinema i la música junts.

Durant l'any 2013, la industria del videojoc va factura 762 milions d'euros només a Espanya (segons dades de aDeSe[1]) i, tot i el descens del 7.3% respecte 2012, els videojocs continuen sent la primera indústria d'oci audiovisual e interactiu del nostre país (veure figura 1).

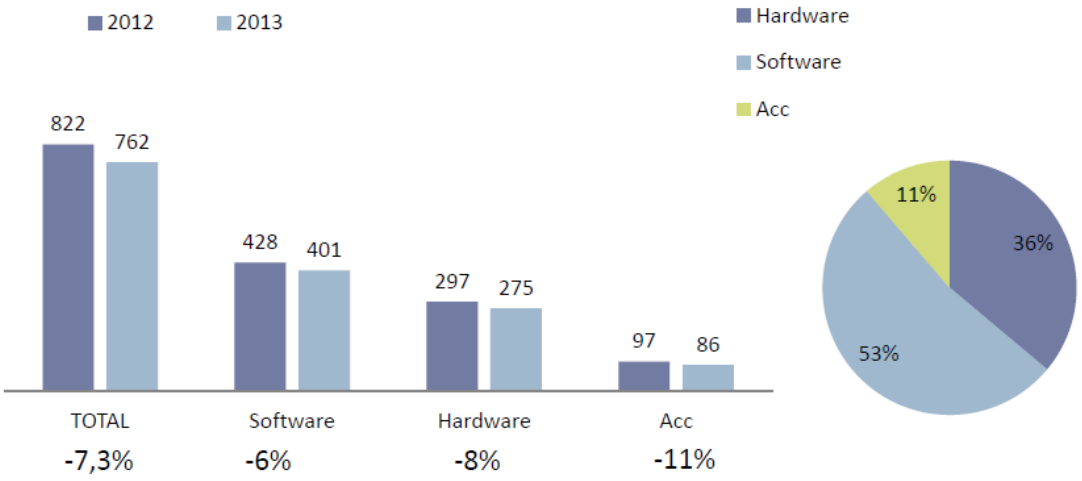


Figura 1: Valor del consum a Espanya els anys 2012 i 2013 en milions d'Euros (Font: Balance Económico 2013 Industria Española del Videojuego - www.aevi.org.es)

Estadístiques i dades rellevants

A més dels 762 milions d'euros generats en consum durant l'any 2013, es poden extreure dades interessants d'aquest balanç econòmic:

- Les ventes de software han generat 401 milions d'euros gràcies a 10.8 milions d'unitats venudes, que suposa un descens d'un 6% respecte l'any 2012 en termes de consum i un 15% en unitats (veure figura 2).

[1] Asociación Española de Distribuidores y Editores de Software de Entretenimiento

	AÑO 2013			
SOFTWARE	UNIDADES	12/13	VALOR	12/13
	En miles	Unidades	Millones €	Valor
TOTAL SOFT	**10.830**	**-15%**	**401**	**-6%**
Juegos PC	948	-37%	19	-35%
Juegos PC educativos	45	-44%	1	-51%
Videojuegos Consola	9.837	-12%	381	-4%

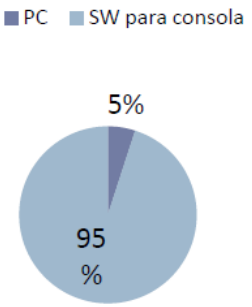


Figura 2: Evolució del software en valor i unitats (Font: Balance Económico 2013 Industria Española del Videojuego - www.aevi.org.es)

- Les ventes de hardware han suposat un total de 275 milions d'euros amb 1.17 milions d'unitats venudes (un 8% menys que 2012).
- Els accessoris han descendit un 11% i han generat 86 milions d'euros amb 4.4 milions d'unitats venudes.
- Les consoles portàtils han experimentat un major descens. Han baixat un 33% en unitats venudes i un 35% en consum. Les consoles de sobretaula han baixat un 11% en unitats i un 12% en valor (veure figura 3).
- Els jocs de sobretaula continuen dominant el mercat, a on el major descens ha estat en PC (tot i que aquest estudi només reflecteix les ventes en format físic i no té en compte les ventes digitals, el gran gruix del mercat d'ordinador)

	AÑO 2013			
HARDWARE	UNIDADES	12/13	VALOR	12/13
	En miles	Unidades	Millones €	Valor
TOTAL CONSOLAS	**1.172**	**-21%**	**275**	**-8%**
Portátil	481	-33%	80	-35%
Sobremesa	692	-11%	194	-12%

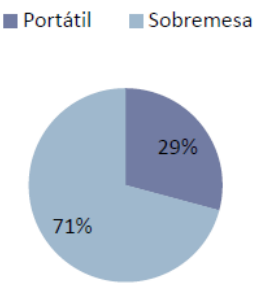


Figura 3: Evolució de les consoles en valor i unitats (Font: Balance Económico 2013 Industria Española del Videojuego - www.aevi.org.es)

Tot i aquestes dades de descens, el descens és molt menor que en altres àmbits de la indústria audiovisual. A més a més, la tendència actual en el sector dels videojocs és un model de negoci anomenat *free to play* o *freemium*. Els videojocs que ofereixen un servei *free to play* són els que, en contra del model clàssic a on es compra el joc i després es pot jugar, ofereixen una part del joc gratuïta i la possibilitat de pagar per més quantitat de joc o per altres experiències de joc, ja sigui amb elements artístics o directament aplicables a la jugabilitat.

Motivacions

La meva motivació personal per a realitzar aquest projecte ha estat la de conèixer i experimentar el desenvolupament dels videojocs des de dins, ja que és en l'àmbit dels videojocs a on vull enfocar la meva vida professional i realitzar aquest projecte pot ser una bona entrada per a aquest sector laboral.

Un dels meus objectius es poder arribar a publicar el projecte una vegada finalitzat. Gràcies als *markets* mòbils i al seu model de negoci és més fàcil distribuir, encara que es tinguin pocs recursos. Per PC també existeix quelcom semblant, *Steam Greenlight*, una plataforma a on es poden penjar els projectes *indie* i, amb els suficients vots de la comunitat, aquest pot sortir a la venta a la plataforma *Steam*.

Objectius del projecte

L'objectiu d'aquest projecte és crear un videojocs de plataformes d'*scroll* automàtic en 2D basat en la música. Això vol dir que tots els elements de l'escenari i els moviments que haurà de fer el jugador per a passar-se'l estaran íntegrament lligats a la banda sonora de la pantalla. A més, el joc ha de ser compatible per als dispositius Android.

Breu història

Un videojoc és un software creat per a l'entreteniment en general i basat en la interacció d'entre una o varies persones i un aparell electrònic que executa el joc. Aquest aparell pot ser un ordinador, una màquina arcade, una videoconsola o un dispositiu mòbil. D'aquests aparells se'n diuen plataformes.

Hi ha videojocs senzills i altres de més complexes, capaços de narrar històries i esdeveniments utilitzant audio i video, demostrant que els videojocs són un altre manifestació de l'art.

El dispositiu d'entrada utilitzat per a manipular un videojoc se'l coneix com a controlador. Aquest varia depenent de la plataforma i poden ser des de palanques amb un sol botó fins al propi dit per a manipular una pantalla tàctil.

Pel que fa a la creació del primer videojoc, ens hauríem de remuntar a l'any 1947 on Thomas T. Goldmith i Estley Ray Mann van patentar un sistema electrònic que simulava el **llançament de míssils** contra un objectiu. Aquest projecte no es podria considerar un videojoc ja que no hi havia imatges en moviment, únicament estàtiques.

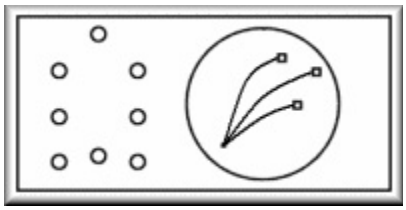


Figura 4: Imatge del joc de míssils de Goldmith i Ray Mann l'any 1947

Més tard va aparèixer el **tres en ratlla** d'Alexander Sandy Douglas, que és considerat el primer joc gràfic per a ordinador, però allò encara no era un videojoc.

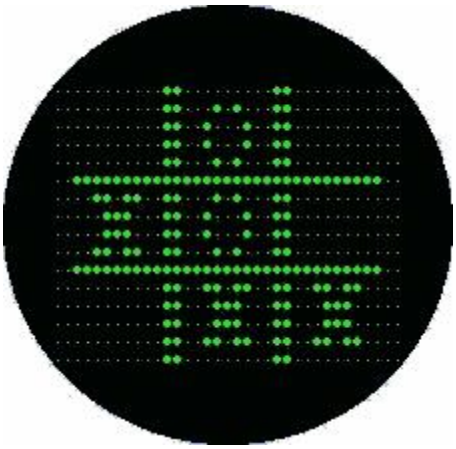


Figura 5: Imatge del primer joc gràfic per a ordinador, el tres en ratlla

Al 1958, William Higinbotham va crear, amb l'ajuda d'un oscil·loscopi i una circuiteria de transistors, un joc que simulava una partida de tenis amb visió lateral, format per una xarxa al mig i dues línies que representaven les raquetes de tots dos jugadors. Aquest joc es manejava amb dos controladors que es van crear específicament per aquest aparell. Tot i el moviment cinemàtic que presentava, el **Tenis for Two** (nom que es va donar al joc) no és considerat encara el primer videojoc ja que utilitzava circuiteria per a funcionar.

Al final, al 1961, va aparèixer **Spacewar**, creat per Russell (juntament amb tres estudiants més del *Tech Model Railroad Club del Massachussets Institute of Technology*) després d'unes 200 hores de treball i molts problemes al llarg del desenvolupament.

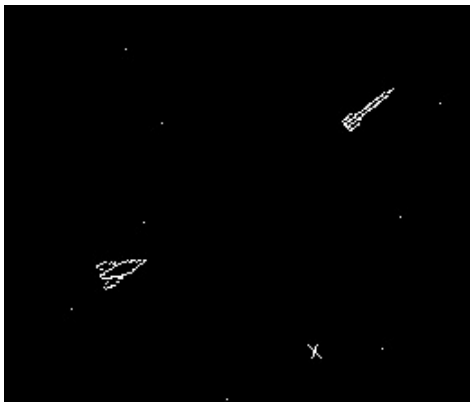


Figura 6: Imatge del Spacewar

El joc consistia en dues naus que s'enfrontaven a l'espai exterior. És considerat el primer videojoc d'ordinador de la història.

Els 70, la crisis dels 80 i l'explosió del sector

Durant la dècada dels 70 es van crear diversos sistemes de joc, però al 1972, Bushnell i Ted Dabney van fundar Atari i van tenir la idea de crear un joc similar a un que ja hi havia al mercat que simulava una partida de ping-pong entre dos jugadors. Tres mesos després va néixer Pong que, una vegada es va

introduir als bars, va provocar tal sensació que les màquines es quedaven col·lapsades perquè no admetien més monedes.

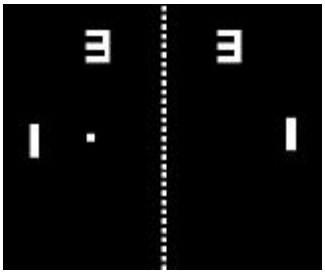


Figura 7: Imatge del Pong

La mateixa Atari va crear altres jocs que actualment son considerats grans clàssics i fins i tot joies dels origens dels videojocs, com per exemple *Asteroids*, *Space Invaders* o el clàssic *Pacman*.

La dècada dels 80 és recordada per la gran crisis que va viure el sector als Estats Units i al Japó al principi d'aquesta causada per la gran saturació del mercat, amb una gran quantitat de jocs clònics[2] i desenes de consoles. Els comerços es trobaven amb una gran quantitat de material que no podien vendre i havien de rebaixar els preus de manera dràstica per a poder treure algun benefici. Com a conseqüència de tot això, en tan sols un any, els videojocs van passar de ser la industria amb major creixement a estar sotmesa a una crisi absoluta.

Però al 1985 la industria es comença a recuperar, gran part gràcies a que Nintendo va llançar **Super Mario Bros**, que va crear un abans i un després en el món dels videojocs, arribant a vendre prop de 10 milions de còpies arreu del món.

Figura 8: Imatge de Super Mario Bros

A partir d'aquí, es podria començar a parlar de la **Batalla de les consoles**.

Els 90 i la batalla de les consoles

La dècada dels 90 es podria considerar com la època daurada dels videojocs. Si avui dia estan tan estesos i son tan populars arreu del planeta és fruit d'aquesta dècada. Durant aquest temps és on apareixen consoles com la Super Nintendo per part de Nintendo, la qual va tenir una de les màximes rivalitats de la història amb la Mega Drive de Sega. Aquests dues companyies col·lapsaven pràcticament tot el mercat del videojoc durant aquesta dècada. Fruit d'aquesta rivalitat van aparèixer grans jocs i

[2] Jocs molt semblants els uns amb els altres, on acostumava a canviar únicament algunes imatges, moviments o cançons

grans consoles que seran recordades durant anys, com per exemple la **Game Gear**, la **Sega Saturn** i la **Nintendo 64**.

També va aparèixer a la industria dels videojocs **Sony**, que va llançar la **Play Station** fruit d'unes negociacions no fructíferes amb Nintendo, que no volia afegir un perifèric que acceptés CDs per a la Super Nintendo. La Play Station va significar un punt d'inflexió en la industria.

Per altra banda, aquesta dècada va ser l'última per a Sega com a creadora de consoles, ja que amb la Dreamcast es va acomiadar del mercat per a centrar-se únicament en la creació de jocs. Durant aquesta època també van aparèixer grans sagues com **Monkey Island**, **Street Fighter II**, **Sonic** o **Lemmings**.

Breu història dels videojocs mòbils

Un videojoc per a mòbil és un videojoc desenvolupat per a que es jugui en un telèfon mòbil, una PDA, un smartphone o qualsevol dispositiu mòbil. Això no inclou les consoles portàtils.

Històricament, els videojocs per a mòbils han tingut sempre un àmbit molt petit i molt sovint es limiten a oferir una bona jugabilitat a falta de bons gràfics. Això és degut a que els mòbils d'abans no tenien prou potència de processadors per a oferir bones experiències gràfiques, encara que moltes vegades era la pròpia tecnologia sobre la qual es programava que limitava l'aplicació.

A finals dels anys 90, els telèfons mòbils eren aparells que només servien per a trucar i rebre trucades. Però alguns fabricants, com Nokia, van decidir oferir algun entreteniment en aquests petits dispositius que disposaven de botons i una pantalla LCD en blanc i negre basats en els primers jocs arcades i consoles dels anys 80. Mai van pensar que això revolucionaria el mercat dels videojocs portàtils.

Aquests jocs van anar evolucionant i alguns oferien la possibilitat de desbloquejar nivells pagant a l'operador de la línia o connectant-se a Internet. Fins a aquest moment, els jocs eren programats en codi màquina i eren afegits directament a la ROM del telèfon, pero amb els mòbils programables, on hi havia una zona per a gravar dades i es podia utilitzar un llenguatge de programació com el JAVA i un cable USB o connexió a Internet per a introduir les dades, van aparèixer multitud d'aplicacions com per exemple calculadores, notes i, al final, videojocs.

L'empresa francesa Gameloft va començar a desenvolupar jocs en blanc i negre per a aquestes petites pantalles i va resultar ser un autèntic èxit. Els mòbils van anar evolucionant i, amb ells, la memòria, la potència i els llenguatges de programació d'aquests (Symbian, J2ME, etc).

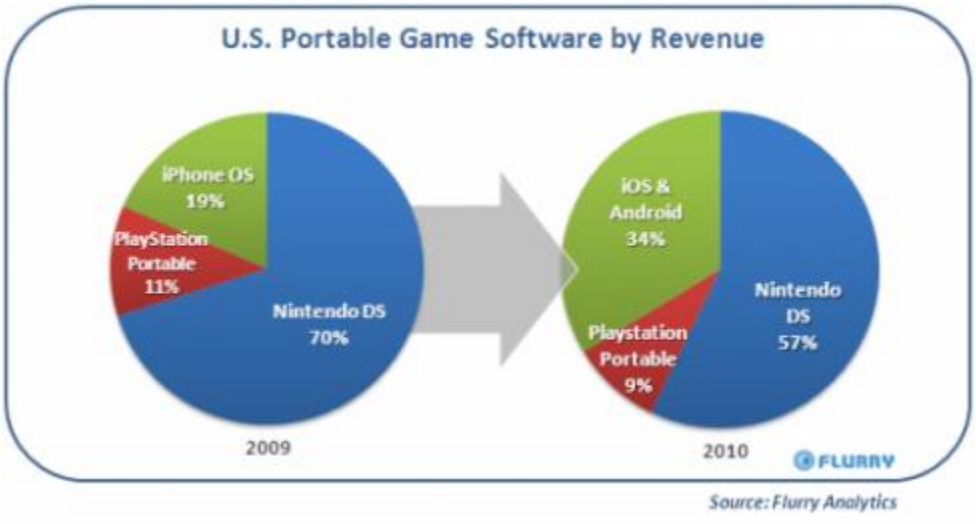


Figura 9: Gràfic comparatiu dels anys 2009 i 2010 en vendes de videojocs portàtils

Actualment el mercat dels videojocs per a mòbils és més gran que qualsevol altre mercat de videojocs portàtils, aconseguint xifres molt elevades. El futur ens prepara videojocs en 3D i videojocs en xarxa, ja sigui en 3G, Wi-Fi o Bluetooth.

Però, el mercat dels videojocs per a smartphones està matant a la industria dels videojocs clàssica? Amb els següents punts podem concloure que si:

- **Emporta't-ho a la butxaca**
  Carregar els videojocs a la nostra butxaca no es exclusiu dels dispositius mòbils, aquests ja existien amb les consoles portàtils de Sony o Nintendo. La diferència és que ara no quedem tancats en el mercat que aquestes dues companyies ens ofereixen. A més, podem jugar sense la necessitat d'utilitzar cap altre dispositiu del que utilitzem més freqüentment: el telèfon mòbil.

- **Altres modes de jugabilitat**
  Jugar amb la pantalla del telèfon, de girar-lo, de jugar amb la gravetat, utilitzar el micròfon i molts altres recursos que els mòbils ofereixen donen al jugador noves experiències de joc que amb les clàssiques consoles portàtils no existien.

- **Revolució en els gràfics**
  Ara per ara, que sigui un videojoc per a mòbil no implica una baixa qualitat de gràfics. Encara que alguns títols encara mantinguin els gràfics bidimensionals, molts d'altres tenen grans gràfics tridimensionals molt ben elaborats. Potser la qualitat no serà la mateixa que les consoles (aparells electrònics dissenyats únicament per aquesta finalitat) però alguns aparells mòbils ja comencen a apropar-se al nivell de gràfics que podia oferir la **PlayStation 2**, com es pot apreciar a la figura 10.

Figura 10: Imatge d'*Infinity Blade*, un joc per iPad

- **El mercat en el núvol**
  Res de cintes, cartutxos, discs ni targetes de memòria. Tot és descarregat al moment i directament des d'Internet. Els títols disponibles són incomptables i només es necessita una connexió 3G o Wi-Fi per a obtenir alguna aplicació gràcies als **Markets**[3].

- **Jocs més econòmics**
  Potser aquest ha sigut el gran factor que ha fet triomfar el mercat: el **preu dels jocs**. Molts d'ells es troben disponibles per tan sols 1€ i aquesta estratègia tan simple està funcionant molt bé. Els desenvolupadors no tenen un gran benefici per cada còpia venuda, però la gran quantitat de còpies que es venen fa que s'aconsegueixin xifres veritablement significatives.
  El jugador també queda satisfet, ja que, al haver pagat un preu tan baix per un joc, li permet adquirir-ne molts més, que garanteix la diversió amb diferents títols i per més temps.

- **Gran quantitat de títols**
  Els smartphones són dispositius més senzills que les consoles comunes, però no per això ha de ser una desavantatge. Justament el disposar d'un hardware i un software accessibles fa que la quantitat de desenvolupadors que estiguin interessats en els smartphones augmenti. Això implica una gran quantitat d'aplicacions noves que apareixen cada setmana als Markets.

[3] Servidors d'aplicacions com per exemple l'Apple Store o el Google Play

- **Tothom pot disfrutar-ho**
  Les grans companyies de la industria dels videojocs intenten forçar al públic a que comprin el seu producte promocionant jocs que només sortiran per aquella plataforma. Això en el món dels mòbils no acostuma a passar, és molt difícil trobar exclusivitats. Molts jocs es troben, per exemple, simultàniament en iPhone i en Android.

- **Interacció amb les xarxes socials**
  Una altra característica que fa que els smartphones siguin grans plataformes per a jocs és la possibilitat de compartir els resultats, parlar amb els amics i interactuar fent servir les xarxes socials.

- **Enfocats en la diversió**
  Finalitzant la nostra llista, tenim que la principal raó de l'èxit dels smartphones com a plataformes de videojocs és la diversió. Molts usuaris reclamen qualitat en els gràfics, no obstant, la gran diferència d'aquests jocs és justament la seva proposta innovadora.

Figura 11: Imatge d'*Angry Birds*

Jocs com **Angry Birds o Fruit Ninja** demostren que no calen grans gràfics per a que existeixi diversió.

Breu història dels videojocs musicals

Segons la **Vikipèdia** són tots aquells jocs "on la jugabilitat està orientada quasi íntegrament sobre l'habilitat del jugar a seguir i mantenir el ritme musical de la banda de so. En un joc de música, el jugador ha de pressionar els botons específics a temps amb la música del joc".

En definitiva, un joc de música és tot aquell que **exigeix al jugar seguir el ritme d'una cançó amb un control**, ja sigui un gamepad o un perifèric a mida. El tipus de controls varia enormement, des de maraques, guitarres, bateries, pistes de ball, micròfons, etc.

El primer joc musical que es coneix és **Otocky**, de 1987 i que va sortir amb una expansió de la NES coneguda com a **Famicon Disk System**, un afegit que només es va vendre al Japó. Otocky era, bàsicament, un joc de naus i trets però amb l'afegit musical. El personatge podia disparar en 8 direccions diferents, cadascuna corresponent a una nota musical. El jugar havia d'apretar el botó i improvisar la música. Així es "componia" música matant enemics.

Figura 12: Imatge d'*Otocky*

Deixant de banda el **Mario Paint** (un perifèric per a poder crear partitures que va sortir el 1992) i algun altre joc educatiu, no va ser fins al 1995, any que va sortir **RockBuster: Quest for Flame,** on podem trobar el primer joc musical pròpiament dit. Aquest joc per a ordinador venia acompanyat d'una pua de guitarra que es connectava al port de l'ordinador (figura 13) i que, com si una guitarra d'aire es tractés, s'havia d'anar movent la pua al ritme de les cançons de la pantalla.

Figura 13: Joc *RockBuster: Quest for Flame amb el seu perifèric*

Cap a l'any 1996, la PlayStation 1 va decidir apostar per l'originalitat amb **PaRappa the Rapper**: un joc musical que podria considerar-se com el pare de tots els jocs moderns del gènere. PaRappa era un gosset que tenia que rapejar juntament amb els seus amics. En aquest cas, rapejar consistia en pressionar una successió de botons d'acord amb el ritme.

Benami (divisió de Konami destinada a desenvolupar videojocs musicals) és la pionera del gènere. Aquesta divisó va començar a desenvolupar jocs de música l'any 1997 amb **Beatmania** per a recreatives, que consistia en una taula de DJ amb set botons a mode de control. Usant aquesta configuració, el jugador tenia que seguir les notes de la pantalla, girant la taula ocasionalment.

Aquesta mateixa divisió va contribuir en altres llicències com **Dance Maniax** (de ball, amb sensors de moviment), **Guitar Freaks** (de guitarra, precursor de Guitar Hero), **Dance Dance Revolution** (de ball, amb una "pista" de ball com a control, figura 14), **DrumMania** (de bateria), etc.

Figura 14: "Pista" de ball de *Dance Dance Revolution*

Jocs com PaRappa van obrir camí a jocs musicals sense perifèrics, com perfectament podria ser **Gitaroo Man** per a PS2 i PSP, **Elite Beat Agents** per a DS i **Amplitude** per a PS2, entre molts altres. En Elite Beat Agents el nostre objectiu és seguir el ritme de la música mitjançant la pantalla tàctil de la consola Nintendo DS

Figura 15: Captura de *Elite Beat Agents*

El gènere de jocs musicals es tan llarg que només exposarem un petit llistat dels més coneguts:

- SingStar (karaoke)
- Guitar Hero / Rock Band (guitarra, bateria, teclat i veu)
- Wii Music (instruments en general)
- Donkey Konga (bongos)
- Samba de Amigo (maraques)
- Etc

Estructura del document

Aquest document està organitzat en un conjunt capítols que són els següents:

1. **Introducció, motivació i objectius del projecte.** En aquest capítol s'explicarà el perquè del desenvolupament d'aquest projecte, quins han estat els objectius proposats i com s'ha organitzat el desenvolupament.
2. **Estudi de viabilitat.** En aquest capítol es justificaran els paràmetres que fan possible el desenvolupament del projecte.
3. **Metodologia.** Aquest capítol conté una explicació de la metodologia emprada per al desenvolupament.
4. **Planificació.** En aquest capítol es defineix l'estratègia seguida per a assolir els objectius plantejats.
5. **Marc de treball i conceptes previs.** En aquest capítol es descriuen els diversos aspectes relacionats amb el desenvolupament general del projecte, que ajudaran a entendre millor els següents capítols. També es tractaran les principals accions desenvolupades durant les primeres etapes de la realització del projecte. S'inclouran els passos d'estudi i aprenentatge de conceptes que s'hagin emprat en el desenvolupament.
6. **Requisits del sistema.** En aquest capítol es defineixen els requeriments de software i hardware que recullen els objectius de l'aplicació, juntament amb les funcionalitats que es volen obtenir.
7. **Estudis i decisions.** Aquest capítol conté una descripció de les eines utilitzades, amb les característiques i l'ús que se'ls li ha donat.
8. **Anàlisis i disseny del sistema.** Aquesta secció proporciona una comprensió precisa de les necessitats del sistema. En aquesta secció es tradueixen els requeriments esmentats en els capítols anteriors a un llenguatge més formal. La part del disseny permet augmentar el nivell d'especificació i realitzar un esquema d'implementació mitjançant diverses eines de programació orientada a objectes.
9. **Implementació.** En aquest capítol es donen a conèixer com s'ha construït l'aplicació, les classes i els mètodes implementats que resulten més significatius per a la comprensió del funcionament del videojoc.
10. **Implantació i resultats.** En aquest capítol es mostraran proves d'execució de l'aplicació, es mostren d'aquest, incloent interfícies, imatges de la partida i tot allò que es pugui visualitzar del videojoc, a més del procés de penjar l'aplicació a Google Play per a la seva descàrrega.
11. **Conclusions.** En aquest apartat s'exposaran les conclusions extretes una vegada finalitzat el projecte.
12. **Treball futur.** En aquesta secció s'exposa tot allò que es podria millorar o ampliar del videojoc.
13. **Bibliografia.** Aquest capítol conté totes les referencies emprades per al desenvolupament del videojoc.

14. **Annexos.** En aquesta secció s'inclouen els elements que no han tingut cabuda en els anteriors
15. **Manual d'usuari i d'instal·lació.** Aquesta secció inclou tots les instruccions per a poder instal·lar i utilitzar l'aplicació.

Estudi de viabilitat

Per a desenvolupar aquest projecte no es requereix d'una gran infraestructura i els costs són limitats.

El material amb el que es compta inicialment és el següent:

- Ordinador amb Windows 7 com a sistema operatiu
- Llicència gratuïta d'estudiant per a Unity3D
- Dispositiu mòbil amb Android.

Al iniciar el projecte, ja es disposava dels recursos necessaris per poder desenvolupar-lo, pel que no ha suposat cap tipus de cost addicional. Cal esmentar que existeix una llicència Pro de Unity, que és de pagament, però aquest projecte està dut a terme amb la versió gratuïta del motor.

Recursos humans

Tot videojoc, per senzill que sembli, requereix de com a mínim tres elements essencials: gràfics, so i programació. Si un d'aquests components falla, el videojoc o pateix de forma notable. En el desenvolupament de qualsevol joc comercial, una gran multitud de persones es dedica exclusivament a cada un d'aquests elements.

Com es pot comprovar en la Figura 16, per molt bona banda sonora o programació realitzada, es necessita una gran capacitat de abstracció per poder arribar a pensar que el conjunt d'un rectangle i un quadrat representen realment un escenari.

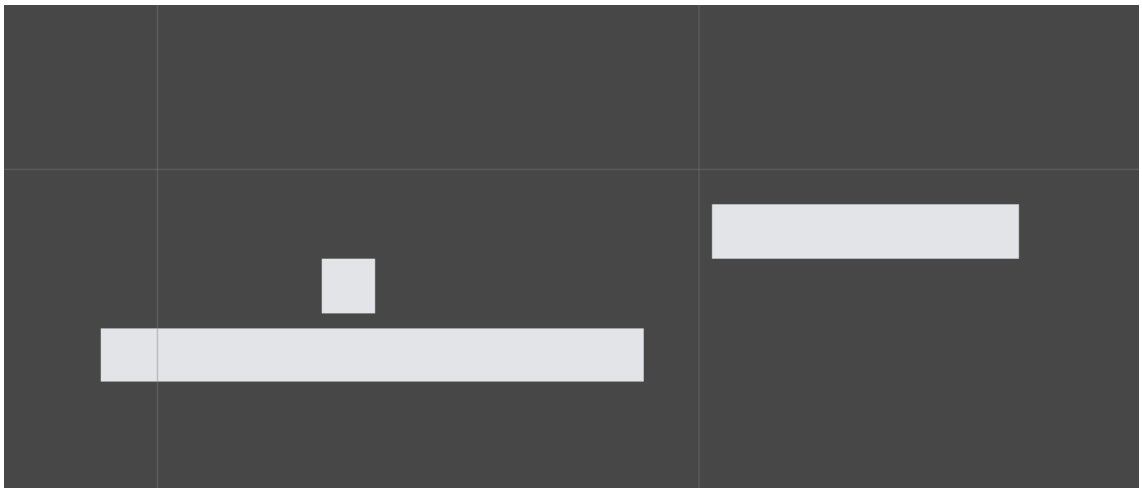


Figura 16: Exemple de joc sense gráfics

Una solució a aquesta problemàtica es demanar ajuda a un dissenyador gràfic. En el procés de crear un videojoc, un dissenyador gràfic té la responsabilitat de realitzar els dibuixos i temes necessaris de la interfície gràfica del joc (fons i cada model del joc – enemics, escenaris, armes, dispars, etc.) que interaccionen amb el comportament del programa. En aquest projecte no s'ha fet ús d'un dissenyador gràfic, el que s'ha fet és obtenir *sprites* i dibuixos lliures de drets d'autor oferts per a la comunitat amb

l'única obligació d'esmentar als autors de les sobres. Pel que fa a la música, s'ha realitzat el mateix tipus de recurs que per a les imatges: música lliure de drets d'autor o música clàssica.

Una altra opció pel que fa a la música hauria estat comprar els drets d'alguna cançó específicament per a la utilització en una app o videojoc, però no ha estat al cas. Tot i això, en el pressupost final si que s'han afegit els costos de comprar una cançó.

Avaluació prèvia de costos i mitjans

La valoració de l'anàlisis econòmic ha estat realitzar dues vegades: el real i el cas en que s'hagués hagut de comprar tot el maquinari i les llicències.

Cas real

En aquest estudi, es planteja el cas real, on no s'hagut de comprar maquinari expressament per al desenvolupament del projecte i aquest s'ha realitzat amb eines gratuïtes.

- Hores d'estudi: 14 dies * 4 hores/dia * 10€/hora = 560€
- Hores de desenvolupament: 4 mesos * 22 dies/mes * 4 hores/dia * 10€/hora = 4400€
- Hores de recerca de material: 7 dies * 4 hores/dia * 10€/hora = 280€
- **TOTAL = 5.240€**

En total, en feina feta real, el projecte està estimat en uns 5240€

Cas ideal

En aquest estudi es planteja un cas hipotètic més pròxim a la realitat, on s'han hagut de comprar tot el necessari per al desenvolupament del projecte i s'ha hagut de subcontractar personal específic per algunes de les tasques del desenvolupament. D'aquesta manera, tenim 4 perfils diferents:

- Analista/dissenyador: 30€/hora
- Programador: 25€/hora
- Il·lustrador/animador: 25€/hora
- Compositor: 25€/hora

TASCA	PERFIL	HORES	COSTOS
LECTURA I INVESTIGACIÓ	Analista/dissenyador	20	600 €
ESTUDI DE UNITY	Programador	30	750 €
DISSENY D'ALGORITMES	Analista/dissenyador	30	900 €
IMPLEMENTACIÓ	Programador	110	2750 €
PROVES I OPTIMITZACIONS	Programador	30	750 €
CREACIÓ DE MELODIES I EFECTOS SONORS	Compositor	30	750 €
CREACIÓ DE MODELS	Il·lustrador/Animador	20	500 €
CREACIÓ D'ANIMACIONS	Il·lustrador/Animador	40	1000 €
MEMÒRIA	Analista/dissenyador	80	2400 €
TOTAL			**10400 €**

Pel que fa als costos de maquinària, s'hauria de tenir en compte els costos dels ordinadors, i del material necessari per a realitzar tots els passos. S'inclou el material que haurien d'usar tots els membres de l'equip per a la realització del projecte. S'ha considerat que el preu mitjà dels ordinadors és de 1.000 €.

COMPONENT	UNITATS	PREU UNITARI	PREU
ORDINADOR	4	1000 €	4000 €
TECLAT MUSICAL AMB ENTRADA USB	1	250 €	250 €
TAULETA DIGITALITZADORA	1	140 €	140 €
TOTAL			**4390 €**

Pel que fa a costos de software, s'han inclòs tots els tipus d'eines que l'equip necessitaria:

SOFTWARE	PREU UNITARI	PREU
UNITY PRO	1500 €	**1500 €**
UNITY ANDROID	1500 €	**1500 €**
UNITY iOS	1500 €	**1500 €**
ADOBE PHOTOSHOP	69.99 € mes * 5 mesos	**245.95 €**
AUDACITY	Gratuït	**0 €**
MICROSOFT PROJECT	1369 €	**1369 €**
MICROSOFT VISIO	399 €	**399 €**
TOTAL		**6513.95 €**

D'aquesta manera, el cost total del projecte està estimat en **21303.95€**.

Metodologia

Scrum és una metodologia de desenvolupament àgil, que està basat en el desenvolupament iteratiu i incremental, que es caracteritza per:

- Adoptar una estratègia de desenvolupament incremental vers una planificació i execució completa del producte.
- Solapament de les diferents fases de desenvolupament i no realitzar una rere l'altre amb cicles seqüencials o en cascada.

Scrum és un model de referència que defineix un conjunt de pràctiques i rols i que pot prendre com a punt de partida per a definir els procés de desenvolupament que s'executarà durant tot el projecte.

Els rols principals en Scrum són:

- **ScrumMaster**: Manté els processos i treballa de forma similar a un director de projecte.
- **ProductOwner:** Representa els interessats externs i interns. Representa la veu del client i s'assegura de que l'equip Scrum treballi de forma adequada des de la perspectiva del negoci. Escriu les tasques, les prioritza i les col·loca al *Product Backlog*.
- **Team**: Desenvolupadors del projecte encarregat d'entregar el producte. Un petit equip de 3 a 9 persones amb les habilitats necessàries per a realitzar la feina (anàlisis, disseny, desenvolupament, testeig, documentació, etc).

Durant cada *sprint* (període d'entre una i quatre setmanes), l'equip crea un increment de software "potencialment entregable". Aquestes característiques són un subconjunt del *Product Backlog*, que és un conjunt de requisits d'alt nivell prioritzats que defineixen la feina a fer. Els elements del *Product Backlog* que formen part de l'sprint es determinen a la reunió d'*Sprint Planning*.

Durant aquesta reunió, el *ProductOwner* identifica els elements que han d'estar completats i l'equip determina la quantita de feina que es pot comprometre a fer durant el següent sprint. Durant aquest, ningú pot canviar l'*Sprint Backlog*, el que significa que els requisits estan congelats durant aquest sprint.

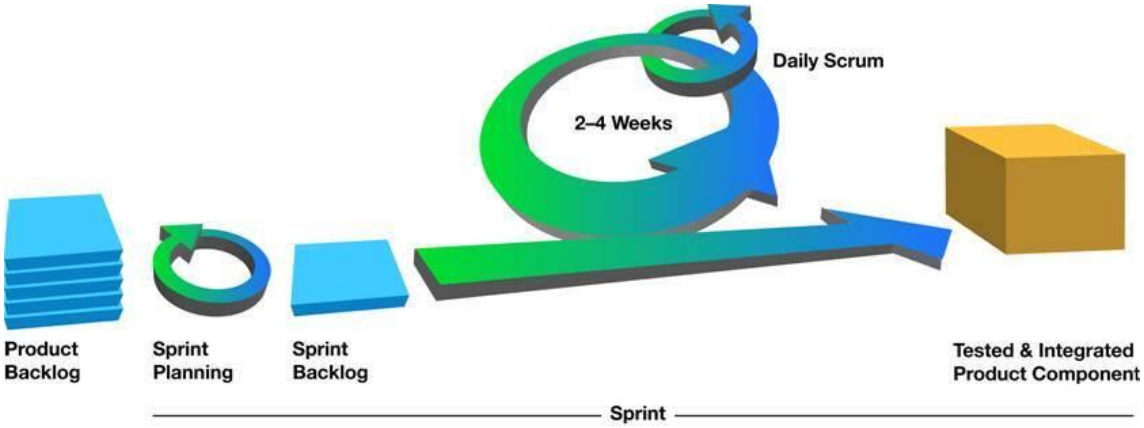


Figura 17: Marc de treball SCRUM

Cada dia, es realitza una *Daily Meeting* que consisteix en que cada membre del grup respongui a tres preguntes:

- Què vaig fer ahir?
- Que faré avui?
- Tinc algun problema que m'impedeixi continuar? I si es així, algú em pot ajudar?

Això permet una alta comunicació de l'equip per a detectar problemes i poder solucionar-los el més aviat possible. El temps òptim d'aquesta reunió és de 15 minuts, ja que en aquesta reunió no s'ha de parlar de desenvolupament o metodologia sinó únicament de tasques realitzades o per realitzar. Aquesta reunió ha de començar sempre a la mateixa hora i fer-se al mateix lloc.

Després de cada sprint, es duu a terme una retrospectiva de l'sprint, a on tots els membres de l'equip deixen les seves impressions sobre l'sprint superat per a poder realitzar una millor continua del procés.

Beneficis de l'Scrum

- **Flexibilitat contínua als canvis:** Gran capacitat de reacció davant els requeriments canviants generats per les necessitats del client o l'evolució del mercat.
- **Reducció del Time to Market:** El client pot començar a utilitzar les característiques més importants del projecte abans de que estigui completament acabat.
- **Millor qualitat del software:** El treball metòdic i la necessitat d'obtenir una versió del treball funcional després de cada iteració ajuda a l'obtenció d'un software d'alta qualitat.
- **Major productivitat:** S'aconsegueix, en altres raons, degut a l'eliminació de la burocràcia i la motivació de l'equip proporcionada pel fet de que es poden estructurar de manera autònoma.
- **Predicció dels temps:** A través d'aquest marc de treball es coneix la velocitat mitjana de l'equip per sprint, amb els que es possible estimar de manera fàcil quan es podrà fer ús d'una determinada funcionalitat que encara es troba en el Backlog.
- **Reducció de riscos:** El fet de dur a terme les funcionalitats de més valor al principi i saber la velocitat en la que avança l'equip permet detectar riscos innecessaris de manera anticipada.

Planificació

Per a qualsevol projecte, sempre és important fer una valoració del temps dedicat. En aquest capítol es descriu una planificació que s'ha seguit durant la elaboració del projecte.

Tasques planificades

Planificació del projecte

En la primera tasca es va definir la planificació del projecte i les diferents parts que el componen, així com el temps estimat per a realitzar-lo i les principals eines que s'utilitzaran.

Definició del joc

En aquesta segona fase es va definir el guionatge i els diferents elements d'interacció que intervenen en el joc. Es va definir el document de disseny del joc (GDD) i les diferents tasques del projecte per a poder seguir correctament una metodologia SCRUM.

Estudis previs

En aquesta fase es va estudiar el funcionament del motor de videojocs Unity, que permet desenvolupar videojocs per a multitud de plataformes i en diferents llenguatges.

Primeres implementacions

En aquesta fase es va començar a implementar un senzill personatge que seguir els indicacions del jugador i que detectés col·lisions.

Primera implementació d'un escenari senzill

En aquesta fase es va començar a desenvolupar un senzill nivell que contingués alguns dels elements que podrien sortir en el joc.

Disseny de artístic del joc

En aquesta etapa es va investigar sobre les llicències d'autor i què es podia utilitzar i què no sense la necessitat d'un artista gràfic o d'un music. Una vegada finalitzada, es van escollir els elements necessàries per al desenvolupament del joc.

Adaptació pantalla i implementació completa de personatge

En aquesta part, la més llarga del desenvolupament, es va crear la pantalla i es va implementar i depurar els elements de la pantalla i del personatge, per a poder tenir una demostració del joc que segueixi el ritme de la música.

Efectes especials

En aquesta fase es varen afegit efectes especials a la pantalla, tals com sons, sistemes de partícules o sistemes d'il·luminació.

Post-producció

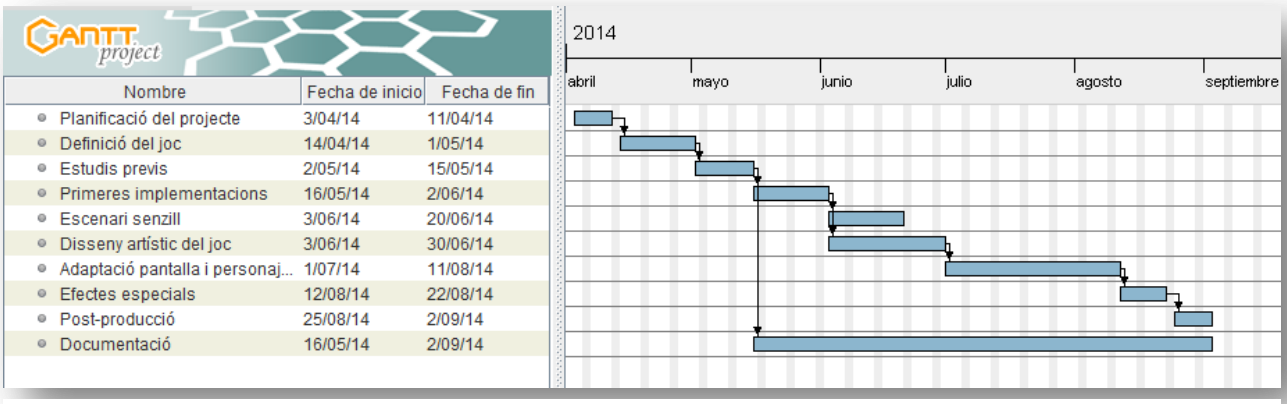
En aquesta fase es va creat tot el que no tenia relació directa amb el desenvolupament: pujada als Markets, inclusió d'anuncis, etc.

Documentació

La documentació és una tasca constant que s'ha de portar a terme durant tot el projecte. Els diferents mètodes, funcions i idees que van apareixent durant el desenvolupament s'han de plasmar a la documentació.

Temps estimat

Es va planejar una durada aproximada d'uns 5 mesos que queda reflectida en el diagrama de Gantt de la figura 18:



Nombre	Fecha de inicio	Fecha de fin
Planificació del projecte	3/04/14	11/04/14
Definició del joc	14/04/14	1/05/14
Estudis previs	2/05/14	15/05/14
Primeres implementacions	16/05/14	2/06/14
Escenari senzill	3/06/14	20/06/14
Disseny artístic del joc	3/06/14	30/06/14
Adaptació pantalla i personaj...	1/07/14	11/08/14
Efectes especials	12/08/14	22/08/14
Post-producció	25/08/14	2/09/14
Documentació	16/05/14	2/09/14

Figura 18: Diagrama inicial de Gantt

Resultats estimats

S'espera crear un videojoc totalment funcional i mínimament divertit amb una pantalla que faci la funció de demostració, al mateix temps que compleixi totes les expectatives i aspectes esmentats en els anteriors apartats.

Marc de treball y conceptes previs

En aquest apartat s'explicarà tot el treball realitzat durant les primeres etapes del projecte:

- Aprendre conceptes previs
- Escollir un motor de joc
- Aprendre els conceptes de la programació de videojocs

Introducció als motors de videojocs (Game engines)

En el desenvolupament de videojocs, a menys que es compti amb un molt de personal i un gran pressupost que permeti implementar-ne un de propi, s'utilitza un motors de jocs (*Game Engine*). La raó és que un motor serveix per abstraure (depenent de la plataforma) els detalls de com fer les tasques més comunes en el desenvolupament de videojocs, com renderitzar, el motor de física, l'entrada/sortida, així, els desenvolupadors (i artistes, dissenyadors, etc) poden centrar-se en els detalls que fan que el joc sigui únic.

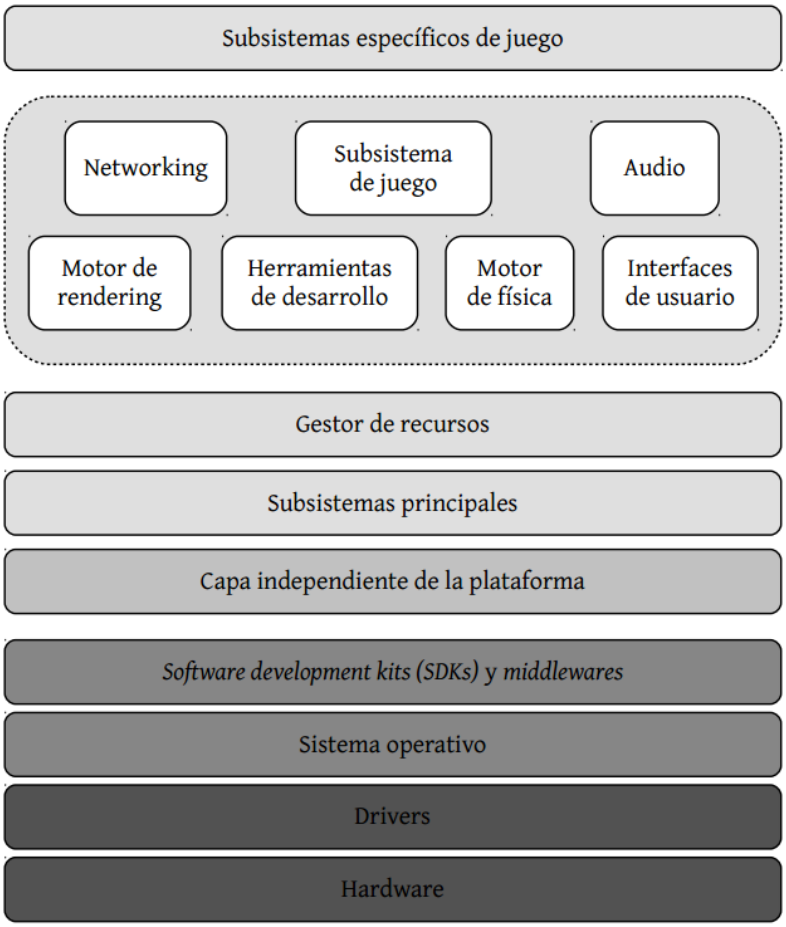


Figura 19: Arquitectura general d'un motor de videojocs

Els motors ofereixen components reutilitzables que poden ser manipulats per a portar un joc a terme:

- Les càrregues, visualització i animació dels models, les deteccions de col·lisions, la física, les comandes d'entrada/sortida, les interfícies gràfiques i parts d'intel·ligència artificial poden ser components d'un motor.
- En canvi, el contingut del joc, els models i textures específics, el comportament darrera les col·lisions i l'entrada/sortida i el mode en que els objectes interaccionen amb el món són components del videojoc en qüestió.

Per parlar d'un motor cal parlar també de l'API i les SDK:

- **API** (application programming interface). Interfície que opera amb el sistema, llibreries i serveis que permeten treure avantatge de característiques particulars.
- **SDK** (software development kit). Col·lecció de llibreries, APIs i erines que fan que sigui possible la programació amb el sistema i els seus serveis.

La majoria de *game engines* proveeixen d'APIs amb les respectives SDKs.

Tipus de motors

Els motors es poden dividir en 3 grups:

- **Programació a baix nivell**

Aquí trobaríem els motors de més baix nivell, aquells que treballen directament amb la targeta gràfica i la seva API, a on hem d'implementar nosaltres mateixos el motor de física i la interacció amb l'entrada/sortida (o obtenir-la mitjançant llibreries externes). En aquest apartat trobem les llibreries **DirectX** (de Microsoft) i **OpenGL** (*open source*). Com a llibreries externes trobaríem **Havox**, **ODE** o **PhysX** per a temes de física i col·lisions, **OSG** per a temes d'escenes, **SpeedTree** per a temes de vegetacions o **FMOD** per a temes de so.

En aquest tipus de motors obtenim una programació completament lliure, il·limitada, però més farregosa.

- **Mig nivell**

En aquest apartat trobaríem tots aquells motors que ofereixen interfícies per a no haver de treballar directament amb comandes que cridin la targeta gràfica, com per exemple **Ogre**.

- **Point-and-click engines**

Son els motors que treballen a més alt nivell. Tots proporcionen una interfície de treball més o menys amigable amb varies opcions, que proporcionen també un IDE sobre el qual treballar i que són més a l'abast de la gent que no sigui tan especialitzada en l'entorn dels videojocs. En aquest apartat tenim la majoria dels motors comercials com podrien ser **Unity3D**, **Unreal Engine** o **GameMaker**. Molts d'aquests motors tenen un inconvenient, i és que alguns estan molt centrats en un tipus específic de joc, com podria ser els FPS (*First Person Shooter*).

Amb els motors *point-and-click* només hem de preocupar-nos del mètode *Update*[4], no pas del renderitzat, el control de les físiques o d'altres detalls perquè d'això ja s'encarrega el motor. Però això provoca un gran inconvenient, i és que es limita força la capacitat de control i personalització.

Un motor es pot adquirir (la majoria dels casos pagant una llicència) o construint-ne un de propi. Totes dues opcions tenen avantatges i inconvenients:

- Si es compra:
  - S'obté un alt nivell de suport.
  - És la solució més còmoda.
  - Facilita el desenvolupament de l'aplicació.
- Si es desenvolupa
  - Es poden implementar funcionalitats a gust.
  - Fa falta temps.
  - Fa falta experiència.

Descripció d'alguns game engines

OGRE

Ogre és un motor multiplataforma de codi obert per a poder treballar amb gràfics de manera professional i fàcil. Té l'opció de programarse en C++ o Python i acostuma a ser l'alternativa a *OpenGL.* Ogre ha estat desenvolupat per a facilitar la programació de videojocs o per a poder treballar amb gràfics complexes, detall que ha estat molt útil per a renderitzar escenes en 3D.

Cal destacar que Ogre no és un motor dissenyat només per a fer jocs, sinó que és un motor de gràfics 3D en general. A causa d'això, Ogre no porta suport natiu per a física ni so, però això no és cap problema, ja que gràcies a l'enorme comunitat existent a Internet, es poden trobar mòduls especialment dissenyats per a Ogre que permeten aquestes funcionalitats.

No s'ha escollit perquè requereix de força preparació prèvia i no permet una visualització *instantània* de l'escena, factor determinant per a aquest joc ja que no es fa la música per a la pantalla sinó la pantalla

[4] El mètode Update és aquell que es calcula a cada frame de l'execució del joc. Veure figura 26

per a la música. A més a més, la integració amb dispositius mòbils requereix d'una preparació prèvia i treballar directament amb llibreries gràfiques per a smartphones.

Unreal Engine

El motor Unreal Engine, creat per Epic Games l'any 1998, és un dels motors de videojocs més utilitzats en la industria amb prop de 200 jocs creats a partir d'aquest.

Actualment la última versió és de maig de 2014 i permet desenvolupar jocs per PC (Windows i Linux), Mac OS X, Xbox 360, PlayStation 3, PlayStation Vita, WiiU, iOS, Android, etc. Té dos tipus de llicència: la gratuïta i la comercial que es tracta d'una llicència que costa 99$ + un 25% dels beneficis a partir dels 50.000$.

Punts forts

- Funcionalitats molt completes.
- Molta documentació si es busca bé.
- La instal·lació ve amb varies escenes i demostracions amb tot el codi i eines.
- S'han realitzat jocs de tot tipus amb aquest motor.

Contres

- Funcionalitats poc intuïtives d'utilitzar
- La documentació es troba dispersa i és difícil de trobar.
- La instal·lació per defecte està orientade a jocs en primera persona.
- Molts tutorials d'UDK Editor però pocs de programació en UnrealScript.

No s'ha escollit aquest motor per estar massa enfocat als jocs en primera persona i ser la dificultat realitzar jocs en dues dimensions.

Ansca Corona

Ansca Corona es un motor enfocat en el desenvolupament de jocs per a plataformes mòbils (Android, iOS) que està proveït de la seva pròpia llibreria de física i que es desenvolupa mitjançant scripts amb el llenguatge Lua per sobre d'una capa en C++/OpenGL.

Disposa de vàries eines auxiliars de les quals es podria destacar **Kwik**, un plugin que permet als usuaris de Photoshop a crear aplicacions buides però permet veure gràficament com quedarien els dissenys en un dispositiu real sense necessitat de tenir coneixements de programació.

Tot i la facilitat que té per a compilar per a dispositius mòbils, no s'ha escollit pel mateix motiu que Ogre: perquè no permet una visualització instantània de la pantalla.

Gideros Mobile i Wave Engine

Motors molt i molt semblants a Corona que es provenen del llenguatge d'scripting Lua en el cas de Gideros i del llenguatge C# i VisualBasic en el cas de Gideros. Tots dos estan provistos també d'un gran nombre de Plugins per a poder realitzar diferents tasques i tots dos són cross-platform (permeten el desenvolupament simultani per a diferents plataformes amb només una implementació).

Pel mateix motiu que Corona, no s'han escollit perquè no contenen un entorn "d'escena" que permeti visualitzar instantàniament el joc, sinó que requereixen de compilar contínuament per a poder visualitar-la i testejar-la.

Unity3D és una eina integrada d'edició 3D per a crear tan videojocs o altres continguts interactius, com per exemple, visualitzacions arquitectòniques o animacions 3D en temps real. En el sentit de que l'entorn gràfic integrat és el mètode principal de desenvolupament, és similar a altres programes com el Director o el Blender game engine. Tot i que el Unity3D té l'entorn gràfic integrat, també té integrades la física i altres elements per fer que, més que un motor gràfic, sigui un motor de videojocs. Com a motor gràfic, és possiblement el millor que està per sota dels 10.000€.

Unity té una versió per a Windows i una altra per Mac OS X i permet produir jocs per a Windows, Mac, iPad, iPhone i Wii. També permet produir jocs de navegador que utilitzen el plug-in de Unity per a reproductors web, compatibles per a Windows i Mac però no per Linux. El reproductor web també s'utilitza per al desenvolupament de widgets de Mac.

Les principals característiques d'aquest motor graphic són:

- Entorn de desenvolupament integrat.
- Desplegament sobre múltiples plataformes (ja comentades anteriorment).
- Admet diversos formats d'arxiu d'editors 3D com per exemple el 3DMax, el Maya o el Blender.
- El motor gràfic utilitza Direct3D (en Windows), OpenGL (tant en Windows com en Mac), OpenGL ES (per a iOS) i llibreries de propietat ATI (per a Wii).
- Scripting de joc basat en Mono (la implementació de codi obert del .NET) amb la possibilitat d'utilitzar fins a tres llenguatges diferents de programació: Javascript, C# o Boo (un dialecte del Python).
- Disponibilitat d'un editor de terrenys, vegetació i creadors d'arbres.
- Multijugador en xarxa amb Raknet.

Figura 20: Imatge de la interfície de Unity3D

En quan obrim Unity podem veure diferents apartats:

- **Scene (vista d'escena)**
  L'escena és l'àrea de construcció del Unity a on construirem visualment cada escena del joc.
  Els jocs de Unity estan dividits en “scenes". Una escena pot ser un menú, un nivell o qualsevol altre cosa semblant.
  La forma més senzilla de treballar amb la vista d'escena seria arrastrant un objecte des de la vista de projecte (més endavant) a la vista d'escena que col·locaria aquest objecte a l'escena. Llavors podríem posicionar-lo, escalar-lo i rotar-lo, tot sense sortir d'aquesta vista.
  La vista d'escena també és el lloc a on s'editen els terrenys (ja sigui pintant-los, esculpint-los o col·locant-hi elements), es col·loquen les llums, les càmeres i altres objectes.

    - **El posicionament dels GameObjects**
      Per a poder posicionar tots els objectes que queden repartits per l'escena hem de fer ús del Gizmo (veure figures 21 i 22). Cada objecte en té un, que es pot veure quan el cliquem.

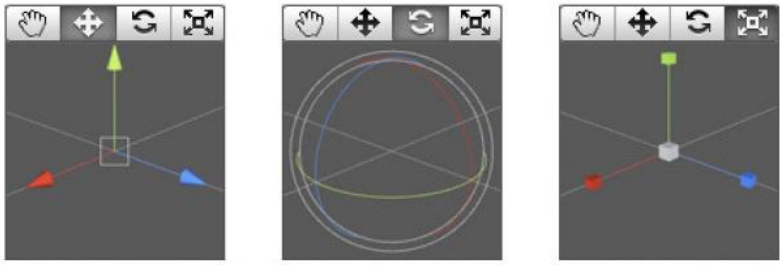


Figura 21: Gizmos de translació, rotació i escalat

La pròpia escena també conté un Gizmo que serveix per posicionar-nos la nostra vista dintre de l'escena.

Figura 22: Gizmo de l'escena

A partir d'aquest Gizmo podem observar l'escena ràpidament des de diferents vistes, ja sigui alçat, planta, perfil, vista isomètrica o vista en perspectiva.

- **Game (vista de joc):** En aquesta vista obtindrem una previsualització del nostre joc. En qualsevol moment podem reproduir-lo i jugar-lo en aquesta vista. Per al funcionament d'aquesta vista, caldrà utilitzar els botons d'iniciar/parar el joc que veurem més endavant a l'apartat de la barra d'eines. Aquesta vista també permet fer una visualització de com quedaria el nostre projecte en diferents aspectes, com per exemple 5:4 o 16:9, entre d'altres.

- **Project (vista de projecte):** Aquesta és la nostra llibreria d'Assets(actius) per al joc. Aquí és on guardarem tots els assets del projecte. Les escenes, els scripts, els models 3D, les textures, els arxius d'àudio i els prefabricats són un exemple. Per a importar un asset nou
al projecte, només caldrà arrastrar-lo.

- **Hierarchy (vista de jerarquia):** Aquesta vista conté tots els objectes de l'escena actual. Alguns poden ser instàncies d'assets i d'altres poden ser models 3D. Quan un objecte és seleccionat a la jerarquia, també és seleccionat a la vista d'escena, permetent modificar un objecte més ràpidament.

- **Inspector (vista d'inspector):** Aquesta vista serveix per a vàries coses. Si es selecciona un objecte es mostraran totes les seves característiques i es podran modificar i personalitzar al nostre gust. Aquesta vista també conté el generador de terrenys. Si un objecte té un script associat, i aquest script té algun atribut públic, aquest es podrà modificar directament des de l'inspector sense necessitat de modificar l'script.

- **Toolbar (barra d'eines):** Aquestes són algunes de les opcions principals del Unity i consta de 5 controls bàsics que fan referència a diferents parts de l'editor.

Figura 23: Toolbar de Unity

Aquest conjunt de botons s'anomenen botons de control i la funció és modificar els objectes que apareixen a la vista d'escena. Començant des de l'esquerra la funció de cada botó és:

- **Hand Tool:** Aquest control permet moure'ns lliurement per l'escena. Si es manté el botó ALT del teclat ens permetrà rotar. CTRL(Windows)/CMD(Mac) ens permetrà fer zoom. SHIFT incrementar la velocitat de moviment.
- **Translate Tool**: Permet moure qualsevol objecte seleccionat en els eixos X, Y i Z.
- **Rotate Tool**: Permet rotar qualsevol objecte seleccionat en l'escena.
- **Scale Tool**: Permet escalar qualsevol objecte seleccionat en l'escena.

A aquests botons s'hi pot accedir ràpidament mitjançant els botons Q,W, E i R del teclat.

Figura 24: Controls de reproducció de Unity

Aquest conjunt de botons s'anomenen controls de reproducció i serveixen per entrar en la previsualització del joc, per pausar-lo o per saltar endavant.

Figura 25: Controls de visualització de Unity

Aquest conjunt de botons i desplegables (disponible sobre la vista d'escena) s'anomenen botons de visualització i, d'esquerra a dreta, són:

- **Render mode** Per defecte es trobarà en "Textured" però conté altres opcions.
  - **Textured**: Les textures es renderitzaran a la vista.
  - **Wireframe**: Les superfícies no es renderitzaran, només es veurà la malla de filferros.
  - **Textured Wireframe**: Les textures es renderitzaran, però també veurem la malla d'arestes.
- **Color modes**: La segona opció és el mode de color, apareix "RGB" per defecte. Però també tenim les opcions Alpha, Overdraw i Minimaps.
- **Llums**: Aquest interruptor encén o apaga la il·luminació de l'escena.
- **Efectes**: Aquest últim interruptor activa o desactiva els efectes de *skybox*, *lense flare* i boira.

- **Console (consola):** Encara que no estigui present a la imatge anterior, cal recordar que el Unity disposa d'una consola a on informarà de qualsevol error de compilació o d'execució i de qualsevol warning que produeixi el codi (també serveix com a eina de debug).

Totes aquestes vistes no estan subjectes a la posició inicial, sinó que es poden moure, redimensionar o tancar les necessàries per a poder oferir un bon ambient de treball.

Elecció de Unity3D

S'ha escollit Unity3D perquè, a part de l'experiència prèvia amb el motor, amb la nova versió 4.3 permet realitzar jocs directament en dues dimensions, que ocasionen que el motor "ignori" la tercera dimensió i faciliti la implementació del joc. Un altre motiu és que permet una visualització instantània de l'escena i que, gràcies a la "*Assets Store"* podem trobar multitud d'exemples, tutorials i eines per a facilitar el desenvolupament. Sobre l'elecció del llenguatge s'ha escollit C# davant de Javascript per la millor documentació que es pot trobar a la xarxa i la facilitat de programació amb les funcions d'autocompletar de l'editor de Unity, a més de la familiaritat amb el llenguatge. Boo s'ha descartat des del principi per tractar-se d'un llenguatge inferior i, per tant, que té molta menys documentació que els altres dos.

A la figura 26 podem observar un esquema de com funciona l'execució d'un videojoc. El seguiment de l'execució comença amb un mètode **Start** que inicialitza totes les dades i eines que s'utilitzin (variables, elements, connectors amb la targeta gràfica si es necessitessin, etc). A cada tic de rellotge, el sistema llegirà els inputs del jugador, actualitzarà l'estat del joc (col·lisions, intel·ligència artificial, física, etc) i pintarà el resultat per pantalla. En cas de que el joc provoqui un game over, el sistema s'encarregarà de guardar les puntuacions, tancar els fitxers que s'hagin obert i de finalitzar el que sigui necessari.

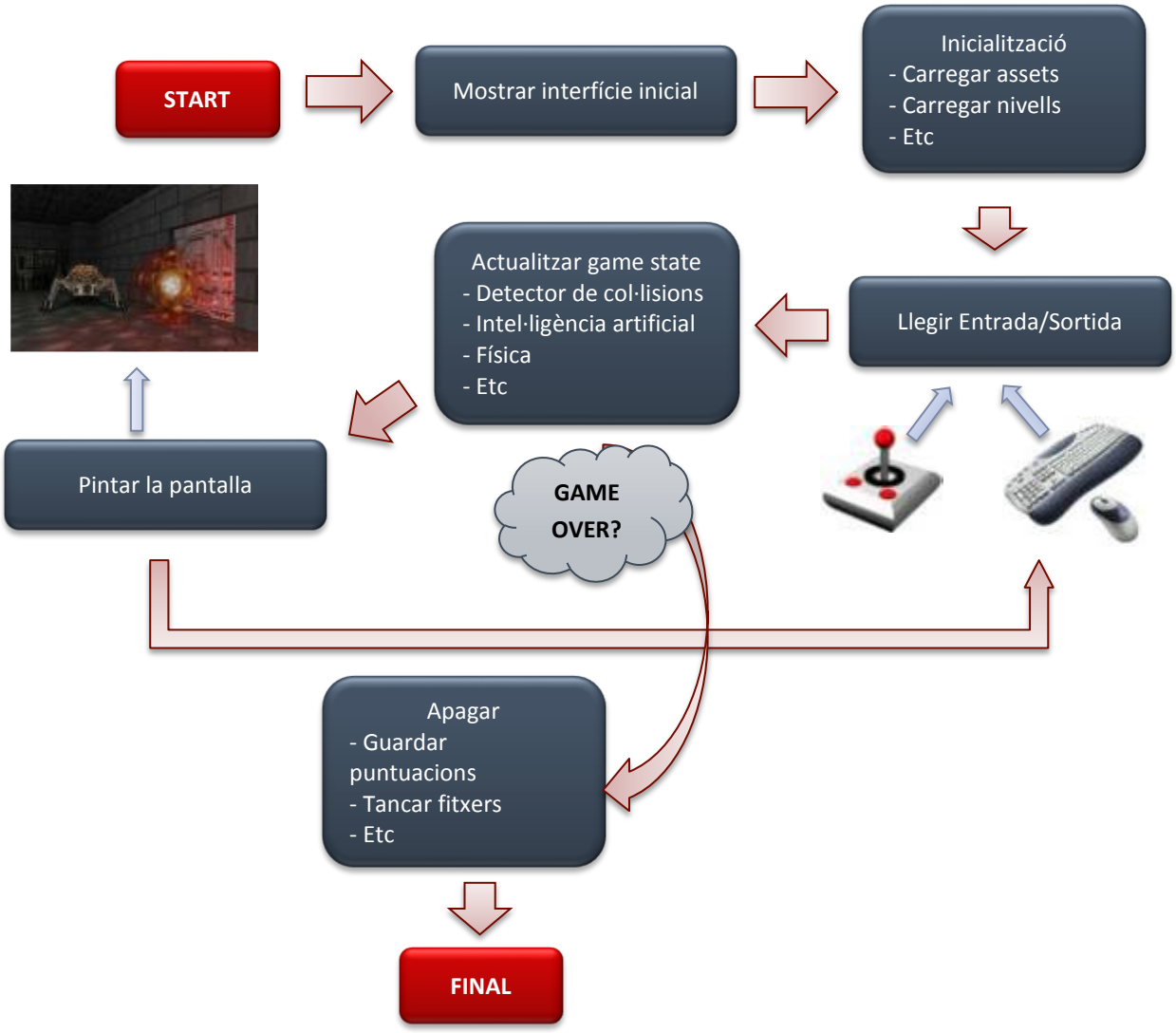


Figura 26: Workflow d'un videojoc

Requisits del sistema

En aquest capítol seran descrits els requeriments, que expliquen a grans trets els objectius de l'aplicació, juntament amb les funcionalitats desitjades.

Dins de qualsevol aplicació hi ha dos tipus de requeriments:

- **Requeriments funcionals:** Descriuen quins són els serveix que ens oferirà l'aplicació, independentment de la implementació.
- **Requeriments no funcionals:** Ens informen de les restriccions que venen imposades pel client o pel propi problema.

Requeriments funcionals

Els principals requeriments que ha de satisfer el joc són:

- Mostrar el menú principal.
- Iniciar una partida.
- Pausar una partida.
- Reiniciar una partida.
- Finalitzar l'aplicació
- Veure els controls de joc.
- Utilitzar les accions del personatge dins de l'escenari.

Mostrar el menú principal

Es mostrarà un menú principal que permetrà al jugador iniciar una partida nova, mirar quins són els controls i sortir de l'aplicació.

Iniciar una partida

Començar una partida nova del nivell de la demostració.

Pausar una partida

Es podrà pausar la partida mitjançant una icona de la interfície, que activarà el menú de joc.

Reiniciar una partida

Es podrà reiniciar una partida a partir del menú del joc.

Finalitzar l'aplicació

Es podrà sortir de l'aplicació mitjançant un botó del menú principal.

Veure els controls del joc

Es podrà veure quins són els controls bàsics del joc.

Utilitzar les accions del personatge dins de l'escenari

El personatge principal podrà saltar i colpejar, amb els que podrà interactuar amb l'escenari.

Requeriments no funcionals

En tot projecte s'ha de prestar especial atenció a tots els aspectes que han de tenir-se en compte quan és té que dissenyar un sistema, més allà de la explicació funcional presentada anteriorment. Els requeriments no funcionals del sistema són aquells que fan referència a restriccions del tipus de disponibilitat dels recursos, seguretat o interfícies externes (hardware i software), entre altres. Aquestes condicions ens permeten executar l'aplicació sense problema.

En quan a la aplicació implementada, s'ha de dir que, des del punt de vista de seguretat, no hi ha cap requisit de control d'accés al programa, ja que es tracta d'una aplicació on els usuaris que hi accedeixen només tenen un rol. Tampoc disposa de dades confidencial o d'alt risc, pel que no es tindrà en compte un sistema de protecció de dades.

Cal afegir que l'aplicació ha estat implementada sobre la plataforma Windows, ja que per a desenvolupar amb el motor Unity, aquest és necessari.

La implementació i les proves de l'aplicació s'han portar a terme sobre un equip amb les següents característiques:

- Inter Core i7
- nVidia GT650M
- 8Gb de RAM
- Windows 7 Home Edition
- Diversos smartphones amb diferents tamanys i densitats de pantalla i diferents processadors.

Estudis i decisions

Aquí anomenarem els programes i llibreries utilitzades per realitzar aquest projecte. Es mostren les eines que han permès la implementació dels prototips, des de les més bàsiques fins aquelles que han servit simplement de suport.

Sistema operatiu

El sistema operatiu amb el qual s'ha desenvolupat aquest projecte ha estat un Windows 7 Home Edition, amb la corresponent versió per arquitectures de 64 bits.

Software utilitzat

MonoDevelop

MonoDevelop és un entorn de desenvolupament integrat lliure i gratuït, dissenyat primordialment per C# i altres llenguatges .NET com Nemerle, Boo i Python. MonoDevelop originalment va ser una adaptació de SharpDevelop per Gtk#, però des de llavors s'ha desenvolupat per respondre les necessitats dels desenvolupadors del Projecte Mono. El IDE inclou maneig de classes, ajuda incorporada, completament de codi, Stetic (dissenyador de GUI) integrat, suport per a projectes, i un depurador integrat des de la versió 2.2.

MonoDevelop pot executar-se en les diferents distribucions de Linux i Mac. Des de la versió 2.2, MonoDevelop ja disposa del suport complet per GNU/Linux, Windows i Mac, completant així una fita per ser un verdader IDE multi plataforma. A més de ser l'IDE que acompanya a Unity.

C#

C# es un llenguatge de programació orientat a objectes desenvolupat i estandarditzat per Microsoft, com part de la seva plataforma .NET.

La sintaxis bàsica deriva del llenguatge C/C++ i utilitza el model de objectes de la plataforma .NET, similar al de Java, tot i que inclou millores derivades d'altres llenguatges.

El nom de C Sharp va ser inspirat per la notació musical, on # "sostingut", (en anglès sharp) indica que la nota (C és la nota do en anglès) és un semitò més alt, suggerint que C# és superior a C/C++. Addicionalment, el signe '#' es pot considerar com un conjunt de quatre '+'.

Tot i que C# forma part de la plataforma .NET, aquesta es una interfície de programació d'aplicacions (API), mentre que C# es un llenguatge de programació independent dissenyat per a generar programes sobre la plataforma. Existeix un compilador implementat per generar els programes per a diferents plataformes com Windows, Unix i GNU/Linux.

Unity

Motor de videojocs utilitzat per a desenvolupar el joc. Per més informació, veure l'apartat Marc de treball y conceptes previs.

Microsoft Word 2013

Microsoft Word és un software destinat al processament de textos. Va ser creat per la empresa Microsoft, i actualment ve integrat en la suite informàtica Microsoft Office.

Originalment va ser desenvolupat per Richard Brodie pel computador de IBM sota el sistema operatiu DOS en 1983. Seguidament van ser programades altres versions per a moltes altres plataformes, incloent les computadores IBM que corrien en MS-DOS (1983). Tot i que és un component de la suite ofimàtica Microsoft Office, també és vent de formar independent i inclòs a la suite de Microsoft Works. Les versions actuals són Microsoft Office Word 2013 per a Windows i Microsoft Office Word 2011 per a Mac. És actualment el processador de textos més popular del món. S'ha utilitzat per a l'edició d'aquesta memòria.

GIMP 2

GIMP és un programa en forma de taller de pintura i fotografia que treballa sobre un "llenç" per capes i que està destinat a l'edició, retoc fotogràfic i pintura a base d'imatges de mapa de bits. Disposa d'una gran quantitat d'eines i se'n fan plug-ins per tenir major compatibilitat amb altres programes. S'ha utilitzat per a l'edició dels sprites i d'imatges del joc.

Audacity

Audacity és un programa d'edició d'àudio que permet realitzar multitud d'operacions sobre arxius de so. Des de retallar, ampliar, atenuar, aplicar-hi un filtre, etc. S'ha utilitzat per a retallar la música de fons del joc i poder aplicar-hi filtres.

yUML

yUML és una eina online de creació de diagrames de classe, d'activitat i de cas d'ús a partir de text pla. Una eina suficientment potent com per crear diagrames de casos d'ús i d'activitat complexos sense necessitat d'instal·lació de cap software. Disposa també de nombrosos codis de colors per a poder personalitzar els diagrames a més de 3 tipus d'estètica: plana de color blanc, plana de color gris i una tercera més semblant a l'escriptura a mà. S'ha utilitzat per a crear els diagrames de casos d'ús del projecte. La url es http://yuml.me/.

Cacoo

Eina semblant a yUML però amb més potència i versatilitat. A diferència de yUML, que funciona a partir d'un text pla, cacoo funciona exactament igual que els altres editors de diagrames convencionals, però amb la diferència de que no cal instal·lar res, ja que és totalment online. Permet crear diagrames UML, diagrames elèctrics, de xarxa, mapes de núvols, etc. La versió gratuïta permet crear fins a 30 diagrames i emmagatzemar-los a la compta, a partir d'aquests, cal comprar "punts cacoo" per a poder seguir creant diagrames. S'ha utilitzat pels diagrames d'activitat i pels diagrames de classe. La seva url és https://cacoo.com.

Gantt Project

Gantt project és una aplicació de software lliure que permet realitzar diagrames de Gantt.

Un diagrama de Gantt és una eina gràfica que té per objectiu mostrar el temps de dedicació previst per les diferents tasques o activitats al llarg del desenvolupament d'un projecte.

El diagrama de Gantt no indica les relacions existents entre activitats, però la posició de cada tasca al llarg del temps respecte les altres fa que es puguin identificar aquests relaciones i dependències. S'ha utilitzat per a l'edició dels diferents diagrames de Gantt que apareixen en aquesta memòria.

Android SDK

L'SDK (Software Development Kit) d'Android, inclou un conjunt d'eines de desenvolupament. Comprèn un depurador de codi, llibreries, un simulador de telèfons, documentació, exemples de codi i tutorials. La plataforma integral de desenvolupament (IDE) oficial és Eclipse juntament amb el complement ADT (Android Development Tools plugin), per crear i depurar aplicacions. A més a més es poden controlar dispositius Android que estiguin connectats.

Les actualitzacions de l'SDK estan coordinades amb el desenvolupament general d'Android. També suporta versions antigues d'Android, per si els programadors necessiten instal·lar aplicacions en dispositius ja obsolets o més antics.

Aquesta eina només s'ha utilitzat per a la compilació del joc per a les plataformes Android.

Google Analytics

Google Analytics és un servei gratuït d'estadístiques de llocs web i aplicacions mòbils per part del buscador Google. Ofereix informació agrupada segons els interessos. Es poden obtenir informes com el seguiment d'usuaris, rendiment d'aquests, resultats d'alguna campanya de marketing, les proves de versió d'anuncis, rendiment de contingut, anàlisis de navegació, etc.

En aquest projecte l'hem afegit com a plugin de Unity per a poder visualitzar les estadístiques de quanta gent accedeix a l'aplicació.

Anàlisi i disseny del sistema

En aquest apartat es descriurà la temàtica i com ha de ser el videojoc, i s'exposarà a nivell conceptual i sense entrar en cap moment en la implementació, els elements més importants per la comprensió del disseny del videojoc i del projecte.

Descripció general

El videojoc es basa en una pantalla en dues dimensions en el que es controla a un personatge que té l'habilitat de saltar i de colpejar. El moviment d'aquest personatge és automàtic, és a dir, es mou automàticament cap endavant, així que el jugador no té cap control sobre la velocitat i la direcció que el personatge té.

La única manera de poder passar el nivell es no caure ni xocar amb res, perquè això provocaria la mort del personatge i obligaria al jugador reiniciar la partida des de zero. L'objectiu del joc és arribar fins al final de la cançó, a on s'indicarà al jugador que ha guanyat i que s'ha aconseguit passar el nivell amb èxit.

Disseny dels elements visuals

A continuació s'explicaran els diferents elements que composen l'apartat visual del joc, tant en la interfície de joc com els sprites[5] dels personatges.

Interfície de joc

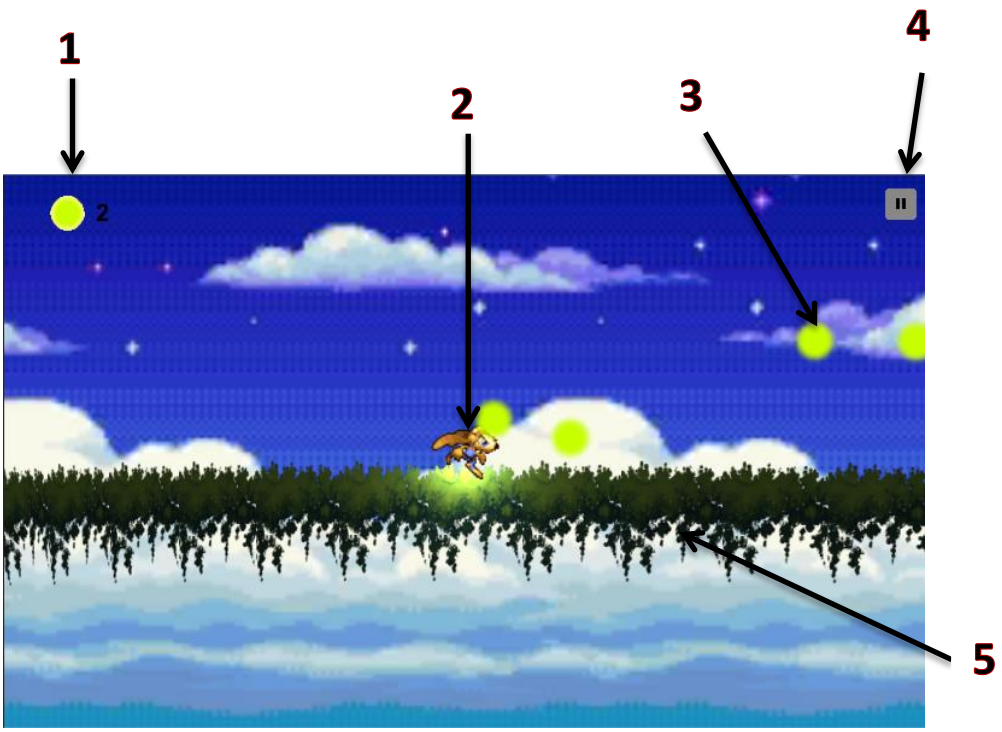


Figura 27: Captura amb els principals elements del joc

[5] Un gràfic mòbil utilitzat en videojocs, mapa de bits que representa algún objecte.

La interfície està formada pels elements que es poden veure a la figura 27:

1. Monedes obtingudes
2. Personatge
3. Moneda
4. Botó de pausa
5. Escenari

Disseny de personatge principal

Per a reproduir qualsevol element en 2D, simplement es requereix un arxiu d'imatge en format ".png" ja que permet, amb el sistema de compressió, no tenir pèrdues de detall i utilitzar el canal "alfa" (RGBA) per obtenir transparències, i així poder mostrar siluetes definides i no només rectangulars.

Per a la reproducció d'animacions el que es requereix són els mateixos arxius d'imatge, però amb la peculiaritat que haurien de contenir un dibuix per a cada instant de la seqüència d'animació que es vol realitzar.

Animació de córrer

Figura 28: Spritesheet de córrer

Animació de saltar

Figura 29: Spritesheet de saltar

Animació de caure

Figura 30: Spritesheet de caure

Animació de colpejar

Figura 31Spritesheet de colpejar

Animació de guanyar

Figura 32: Spritesheet de guanyar

Animació morir

Figura 33: Spritesheet de morir

Disseny del control del personatge

Pel control del personatge ens centrarem en determinar els moviments que volem que realitzi. Així doncs, un cop tenim una llista de tot el que el personatge podrà fer, es determina un diagrama d'estats per tal de saber com es realitzarà la transició d'un estat a un altre.

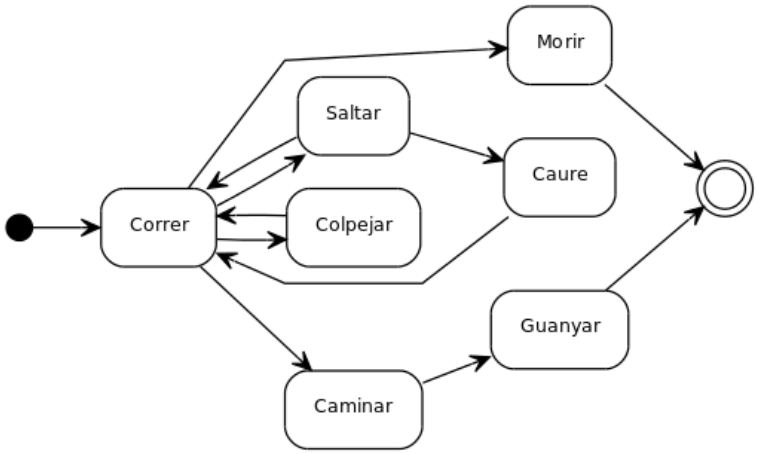


Figura 34: Diagrama d'estats de les diferents animacions del personatge

Disseny de l'escenari

L'escenari ha estat construït mitjançant un paquet de Unity anomenat **Ferr2D** que permet, mitjançant una sèrie de polígons, recrear un escenari en dues dimensions que funciona de la manera il·lustrada a la figura 35.

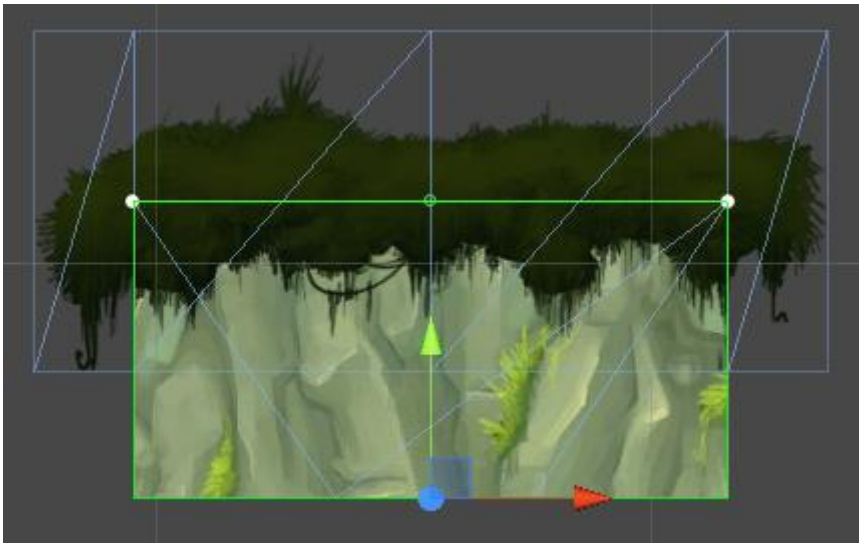


Figura 35: Escenari bàsic creat a partir del plugin Ferr2D

A partir d'una sèrie de polígons i dues textures (una per la gespa i una altra per la roca), el plugin recrea un escenari (físic o decoratiu). Mitjançant punts clau, podem crear un escenari més o menys complicat i complex. Veure figura 36.

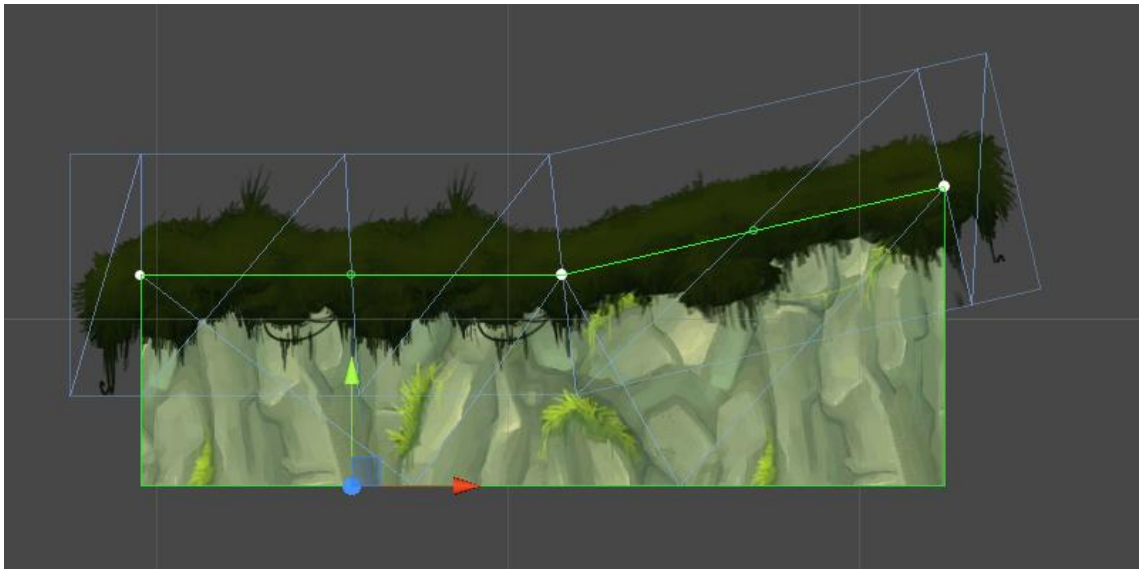


Figura 36: Escenari amb tres vèrtexs

Aquesta ha estat la manera com s'ha creat l'escenari.

Objectes destruïbles

Els obstacles destruïbles de l'escenari estan formats d'un cub allargat amb un sistema de col·lisions adherit per a poder tenir constància de quan el personatge hi ha xocat. Depenent si en el moment de xocar el personatge està colpejant o no, es decidirà si el resultat de la col·lisió és una mort o que l'obstacle queda destruït.

Disseny de les monedes

Les monedes que hi ha a l'escenari estan compostes d'un sistema de partícules que conté una sola partícula de color groc:

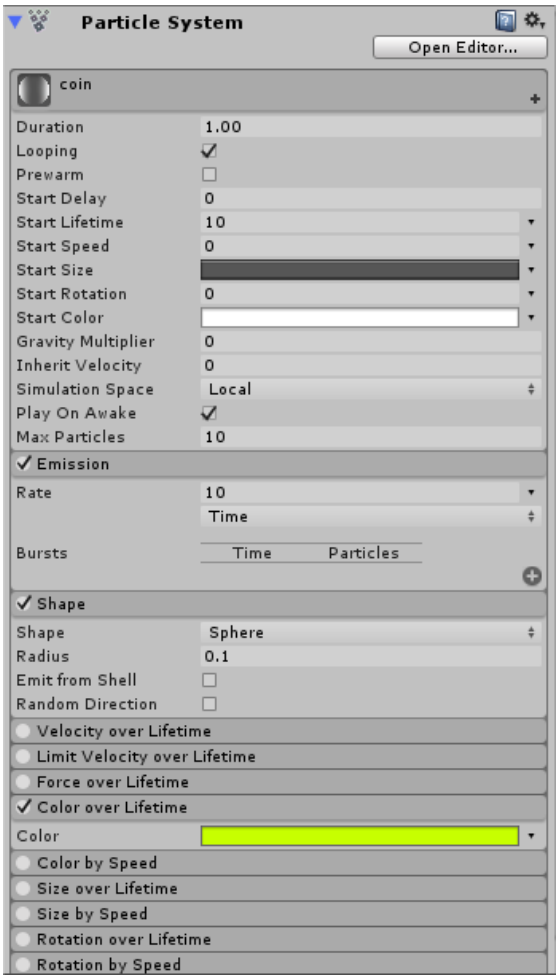


Figura 37: Paràmetres del sistema de partícules de les monedes

Diagrames de casos d'ús

Els casos d'ús descriuen el comportament d'un sistema des del punt de vista de l'usuari. Permeten definir els límits del sistema i les relaciones entre aquets i l'entorn, es pot dir que són les descripcions de les funcionalitats del sistema, independentment de la implementació. Estan basats en el llenguatge natural, de manera que poden ser accessibles per a tots els usuaris.

Menú principal

A l'iniciar el videojoc, apareix un menú que conté 3 opcions: iniciar partida, veure controls i sortir.

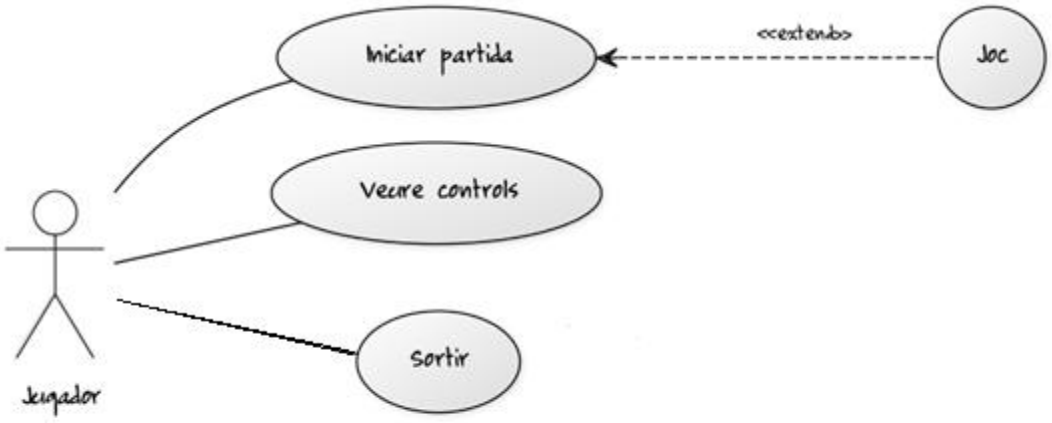


Figura 38: Diagrama de cas d´ús de menú principal

Joc

Una vegada el jugador hagi seleccionat **Iniciar partida** el sistema començarà una nova partida a on el jugador se li permetrà saltar i colpejar amb el personatge.

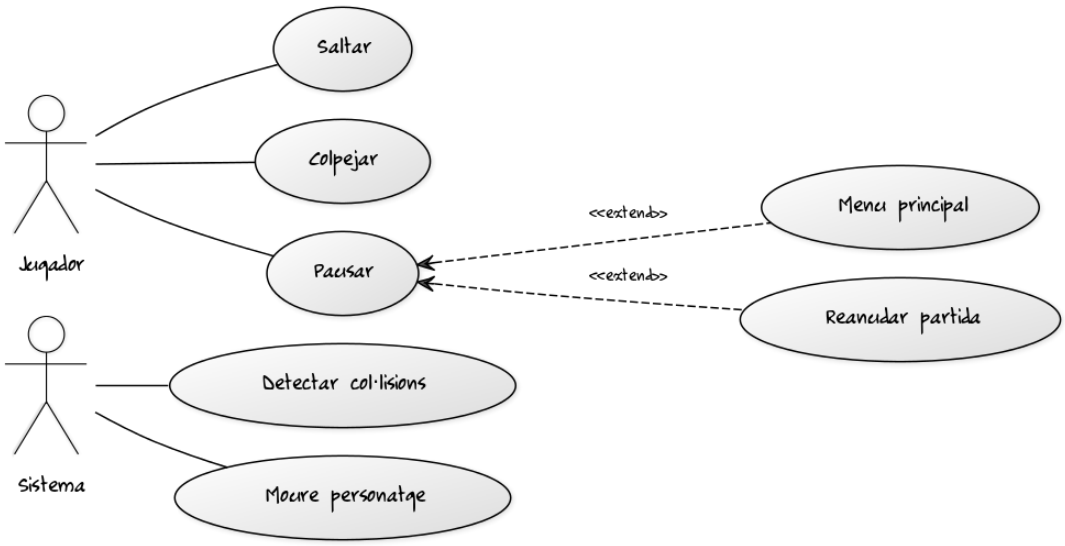


Figura 39: Diagrama de cas d'ús de joc

Fitxes de casos d'ús

Menú principal

Fitxa de cas d'ús: Menú principal	
Descripció	El jugador tria una opció del menú al iniciar-se el programa
Actor	Jugador
Pre-condició	S'ha iniciat el programa
Flux Principal	1. Es tria una opció del menú
Flux alternatiu	Cap
Post-condició	S'executa l'opció escollida
Observacions	Cap

Saltar

Fitxa de cas d'ús: Saltar	
Descripció	El jugador realitza la acció de saltar i el personatge realitza l'acció corresponent
Actor	Jugador
Pre-condició	El jugador ha premut el control de saltar
Flux Principal	1. El jugador prem el control de saltar 2. Si el personatge és a terra 2.1. El sistema dóna la instrucció de saltar 2.2. El personatge realitza l'animació de saltar 2.3. Si en acabar l'animació de saltar, el personatge no és a terra 2.3.1. S'executa l'animació de caure 2.4. Quan el personatge caigui a terra 2.4.1. S'executa l'animació de córrer
Flux alternatiu	1. Si el personatge no és a terra, no s'executarà l'acció de saltar.
Post-condició	S'ha realitzat l'acció de saltar
Observacions	Cap

Colpejar

Fitxa de cas d'ús: Colpejar	
Descripció	El jugador realitza la acció de colpejar i el personatge realitza l'acció corresponent
Actor	Jugador
Pre-condició	El jugador ha premut el control de colpejar
Flux Principal	1. El jugador prem el control de colpejar 2. S'executa l'animació de colpejar 3. El sistema controla si ha xocat amb alguna cosa i realitza els càlculs corresponents
Flux alternatiu	Cap
Post-condició	S'ha realitzat l'acció de colpejar
Observacions	Cap

Menú de pausa

Fitxa de cas d'ús: Menú de pausa	
Descripció	El jugador pausa el joc
Actor	Jugador
Pre-condició	El jugador a premut el botó de pausa
Flux Principal	1. El jugador prem el botó de pausa 2. El sistema pausa el joc i mostra el menú de pausa 2.1. Si el jugador prem el botó de reprendre la partida 2.1.1. El joc continua 2.2. Si el personatge prem el botó de tornar al menú principal 2.2.1. El sistema tanca la pantalla i torna al menú principal
Flux alternatiu	Cap
Post-condició	S'ha mostrat el menú de pausa
Observacions	Cap

Diagrames d'activitat

Diagrama d'activitat de la partida

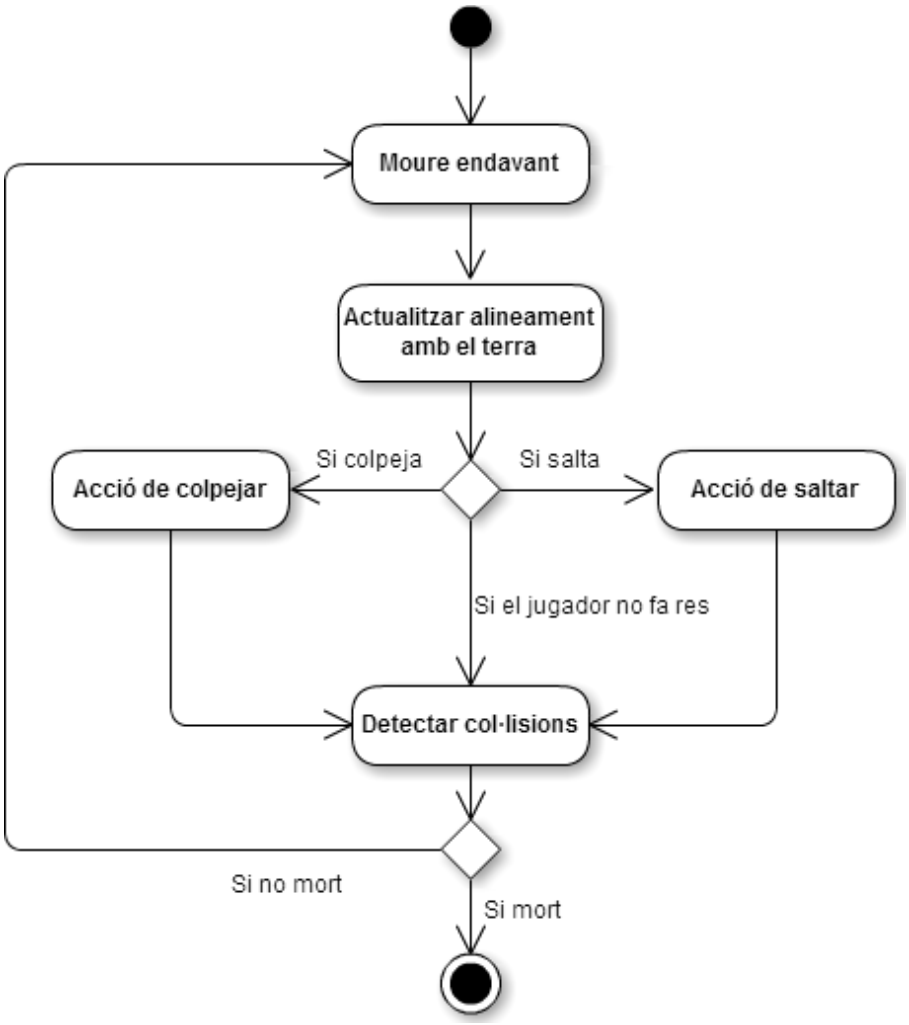


Figura 40: Diagrama d'activitat de la partida

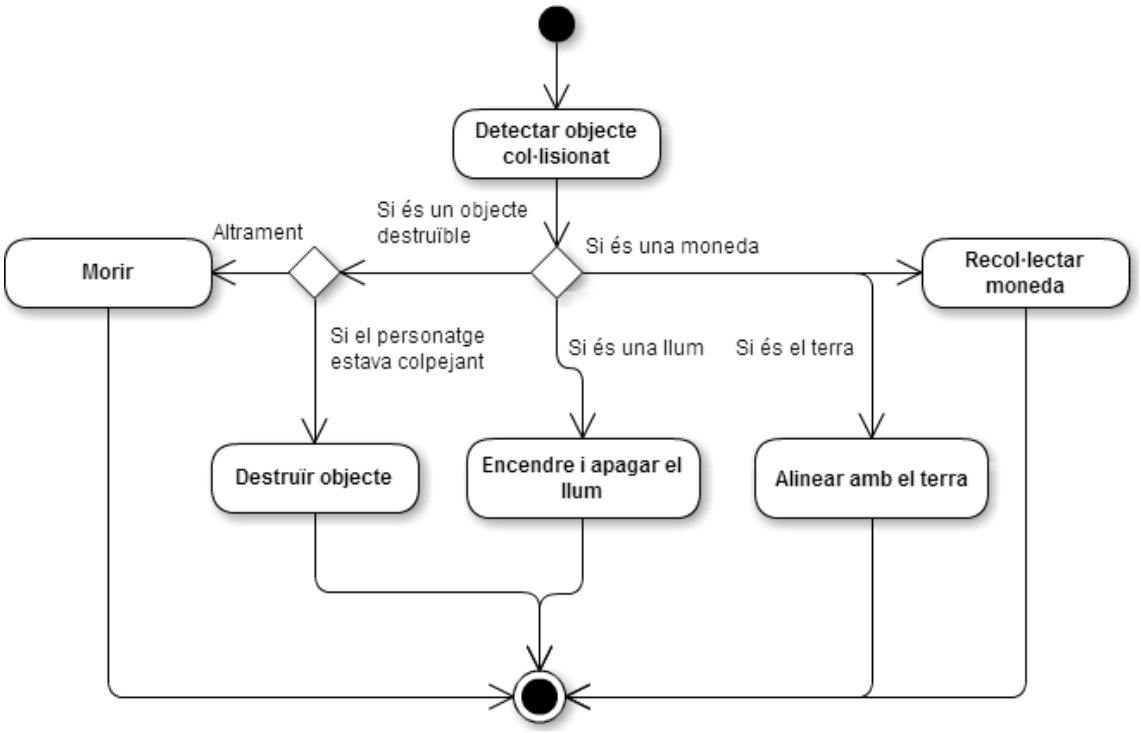


Figura 41: Diagrama d'activitat del càlcul de col·lisions

Implementació

API de Unity

Unity ofereix una API (figura 43) que permet, a través dels scripts, interactuar amb els GameObjects i els seus components, presents a l'escena actual. Cal comentar que totes les classes (scripts) que s'implementen hereten de la classe MonoBehaviour.

En aquesta documentació ens limitem a comentar els atributs i mètodes més utilitzats durant el desenvolupament d'aquest projecte.

GameObject

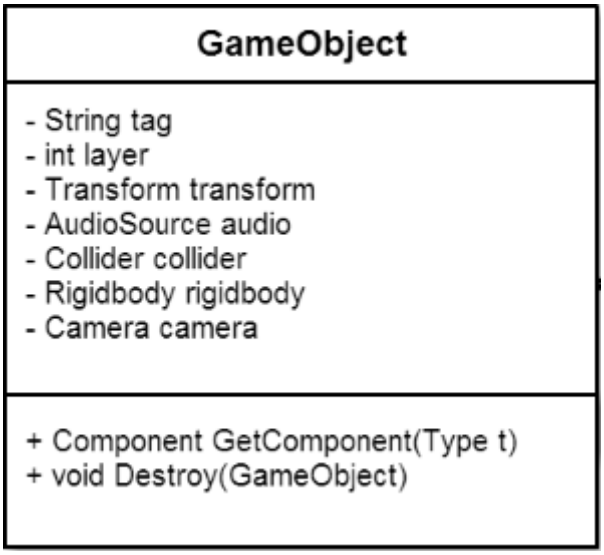


Figura 42: Classe GameObject

Classe bàsica per a totes les escenes de Unity. Conté els atributs necessàries per a la identificació dels objectes dins l'escena més tots els tipus de components que aquest pugui tenir, amb el component Transform com a únic obligatori. Els diferents tipus de components s'aniran explicant a continuació.

- **String tag**: Es pot fer servir per a identificar un objecte.
- **int layer**: Indica a quina capa pertany l'objecte.
- **Transform transform**: Apunta al component del tipus Transform d'aquest objecte.
- **AudioSource audio**: Apunta al component del tipus AudioSource (si l'objecte en té un).
- **Renderer renderer**: Apunta al component del tipus Renderer (si l'objecte en té un).
- **Collider collider**: Apunta al component del tipus Collider (si l'objecte en té un).
- **Rigidbody rigidbody**: Apunta al component del tipus Rigidbody (si l'objecte en té un).
- **Camera camera**: Apunta al component del tipus càmera (si l'objecte en té un).

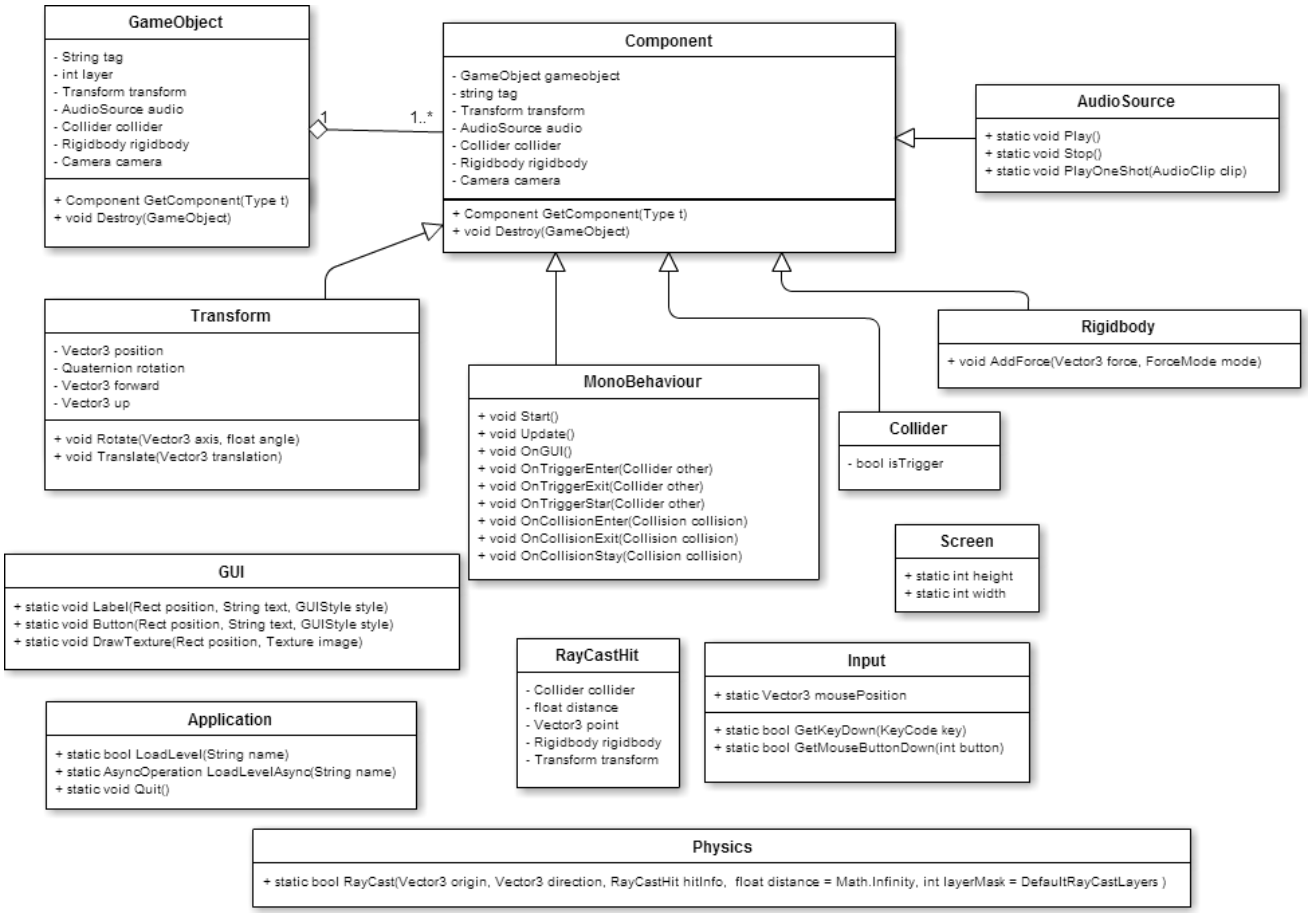


Figura 43: Diagrama de l'API de Unity

- **Component GetComponent(Type t)**
  Permet obtenir un component del tipus especificat.
- **void Destroy(GameObject object)**
  Destrueix el GameObject que rep com a paràmetre.

Component

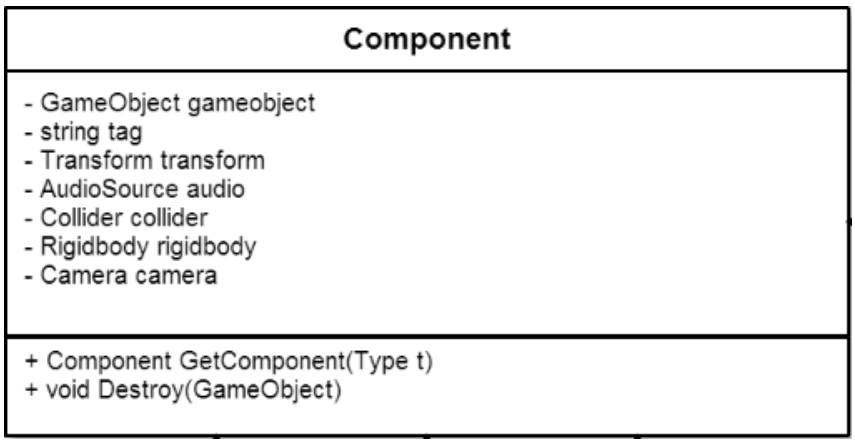


Figura 44: Classe Component

Component és la classe bàsica de qualsevol cosa adjunta a un GameObject. Conté els mateixos mètodes i atributs que el GameObject amb l'afegit del propi GameObject al que està adjunt el Component.

MonoBehaviour

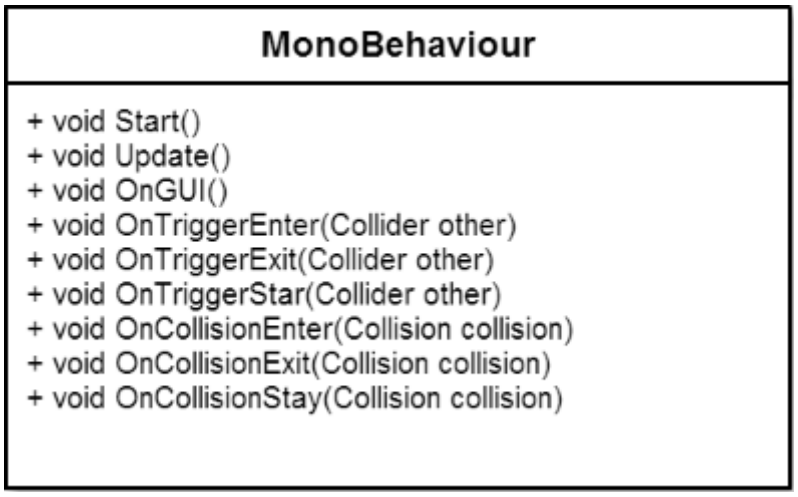


Figura 45: Classe MonoBehaviour

Totes les classes que s'implementen (Scripts) hereten de MonoBehaviour. Conté els mètodes que controlen el flux d'execució del programa. El codi del nostre programa anirà dins d'aquests mètodes.

- **void Start()**
  Aquest mètode es cridat només al primer frame posterior a la creació de l'script.
- **void Update()**
  Update es cridat a cada frame de l'execució, és a on es trobar la major part de l'script.
- **void OnGUI()**
  Mètodes que serveix per a renderitzar i capturar els esdeveniments de la interfície d'usuari.
- **void OnTriggerEnter(Collider other)**
  Aquest mètode és cridat quan el collider entra a la zona del trigger.
- **void OnTriggerExit(Collider other)**
  Aquest mètode és cridat quan el collider surt de la zona del trigger.
- **void OnTriggerStay(Collider other)**
  Aquest mètode és cridat mentre el collider està dins a la zona del trigger.
- **void OnCollisionEnter(Collision collision)**
  Aquest métode és cridat quan el collider/rigidbody ha començat a tocar un altre rigidbody/collider.
- **void OnCollisionExit(Collision collision)**
  Aquest métode es cridat quan aquest collider/rigidbody ha deixat de tocar un altre rigidbody/collider.
- **void OnCollisionStay(Collision collision)**
  Aquest métode es cridat quan aquest collider/rigidbody està tocant altre rigidbody/collider.

Transform

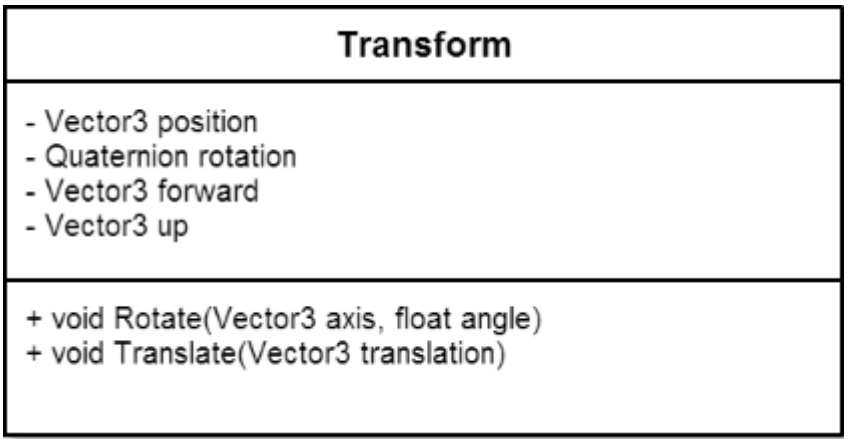


Figura 46: Classe Transform

Cada objecte de l'escena té un Transform. S'usa per guardar i manipular la posició, escala i rotació del objecte al que està adjunt.

- **Vector3 position** : La posició del objecte en el world space.
- **Quaternion rotation** : La rotació del objecte en el world space.
- **Vector3 forward** : L'eix local cap endavant de l'objecte en el world space.
- **Vector3 up** : L'eix local cap a dalt de l'objecte en el world space.
- **void Rotate(Vector3 axis, float angle)**
  Rota l'objecte al voltant de l'eix especificat, el número de graus desitjat.
- **void Translate(Vector3 translation)**
  Realitza una translació a l'objecte d'acord amb el vector especificat.

Collider

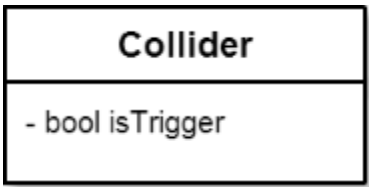


Figura 47: Classe Collider

És la classe bàsica de tots els colliders primitius (BoxCollider, SphereCollider i CapsuleCollider) i del MeshCollider.

Aquests colliders poden ser del tipus trigger. Si són trigger, no poden col·lisionar amb altres objectes rígids, però criden els mètodes OnTriggerEnter(), OnTriggerExit() i OnTriggerStay() ja comentats a la secció del MonoBehaviour quan aquests entren a la zona de detecció.

- **bool isTrigger** : Si és cert, aquest collider es el tipus trigger

GUI

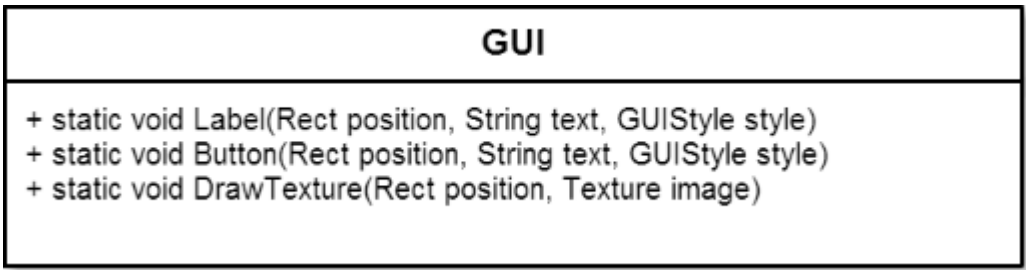


Figura 48: Classe GUI

La classe GUI serveix per realitzar la GUI de l'aplicació, és a dir, la interfície gràfica de l'usuari.

- **static void Label(Rect posicio, string text, GUIStyle style)**
  Mostra un rètol a la pantalla amb un text i estil especificat. El rectangle que rep com a paràmetre representa la posició a la pantalla.
- **static bool Button(Rect posicio, string text, GUIStyle style)**
  Mostra un botó a la pantalla amb un text i estil especificat. Retorna cert quan aquest ha estat premut, altrament retorna fals.
- **static void DrawTexture(Rect position, Texture image)**
  Mostra una textura per pantalla a la posició especificada.

Physics

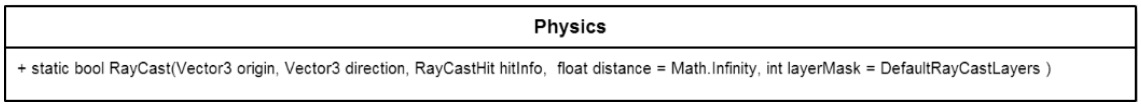


Figura 49: Classe Physics

La classe conté propietats globals de les físiques i alguns mètodes de suport. El que s'ha emprat en l'aplicació és el següent:

- **static bool Raycast(Vector3 origin, Vector3 direction, RaycastHit hitInfo, float distance = Mathf.Infinity, int layerMask = DefaultRaycastLayers, int layerMask = DefaultRayCastLayers)**
  Genera un raig que impacte amb els colliders de l'escena. S'ha d'especificar l'origen d'aquest raig, la direcció, la llargada, i amb quines layers pot interactuar (ignora col·lisions amb colliders que no pertanyen al layer especificat). El paràmetre hitInfo és de sortida i conté la informació de la possible col·lisió que ha tingut el raig.

RayCastHit

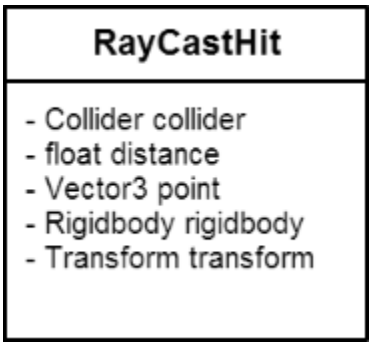


Figura 50: Classe RayCastHit

És una estructura que conté la informació retornada després d'usar un Raycast.

- **Collider collider** : El collider que ha estat impactat.
- **Float distance** : La distància respecte al punt d'origen del raig al punt d'impacte.
- **Vector3 point** : El punt d'impacte on el raig ha impactat el collider.
- **Rigidbody rigidbody** : El Rigidbody del collider impactat (si en té).
- **Transform transform** : El Transform del collider que ha estat impactat.

Input

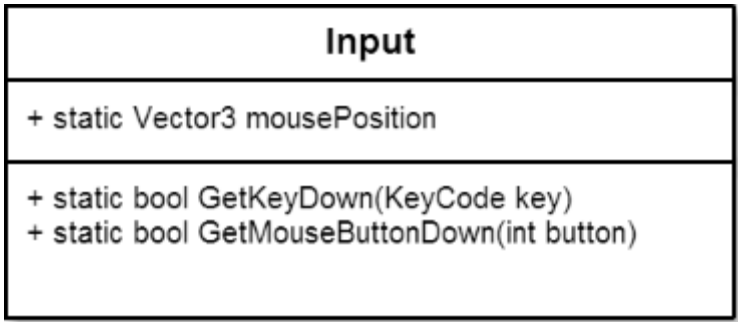


Figura 51: Classe Input

Interfície per consultar l'entrada del sistema.

- **static Vector3 mousePosition** : La posició actual del ratolí en coordenades de píxel.
- **static bool GetKeyDown(KeyCode key)**
  Retorna cert si la tecla especificada del teclat acaba de ser pressionada. Si es vol consultar la tecla Q per exemple, el KeyCode tindria la forma de KeyCode.Q.
- **static bool GetMouseButtonDown(int button)**
  Retorna cert si el botó especificat del ratolí acaba de ser pressionat. El 0 identifica el botó esquerra, el 1 el botó dret i el 2 el botó del mig.

Screen

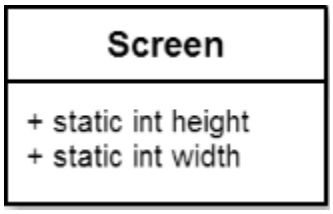


Figura 52: Classe Screen

La classe conté informació de la pantalla.

- **static int height** : Indica l'altura de la resolució de la pantalla.
- **static int width** : Indica l'amplada de la resolució de la pantalla.

Application

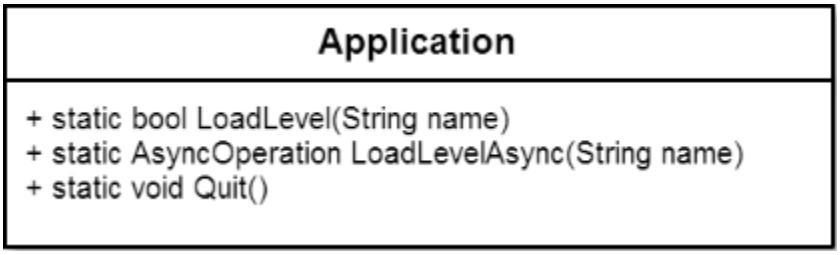


Figura 53: Classe Application

La classe conté mètodes per controlar el flux d'execució de l'aplicació.

- **static void LoadLevel(String name)**
  Permet substituir l'escena actual per un altre de l'aplicació especificada.
- **static AsyncOperation LoadLevelAsync(String name)**
  Permet carregar una escena en segon pla
- **static void Quit()**
  Tanca l'aplicació.

Diagrama de classes de l'aplicació

Com s'ha comentat abans, tot el codi de l'aplicació hereta de MonoBehaviour, per tant, poden utilitzar tots els mètodes d'aquesta classe. En aquest apartat només explicarem els mètodes addicionals que s'han implementat per al desenvolupament del projecte. La figura 54 conté el diagrama de classes de l'aplicació.

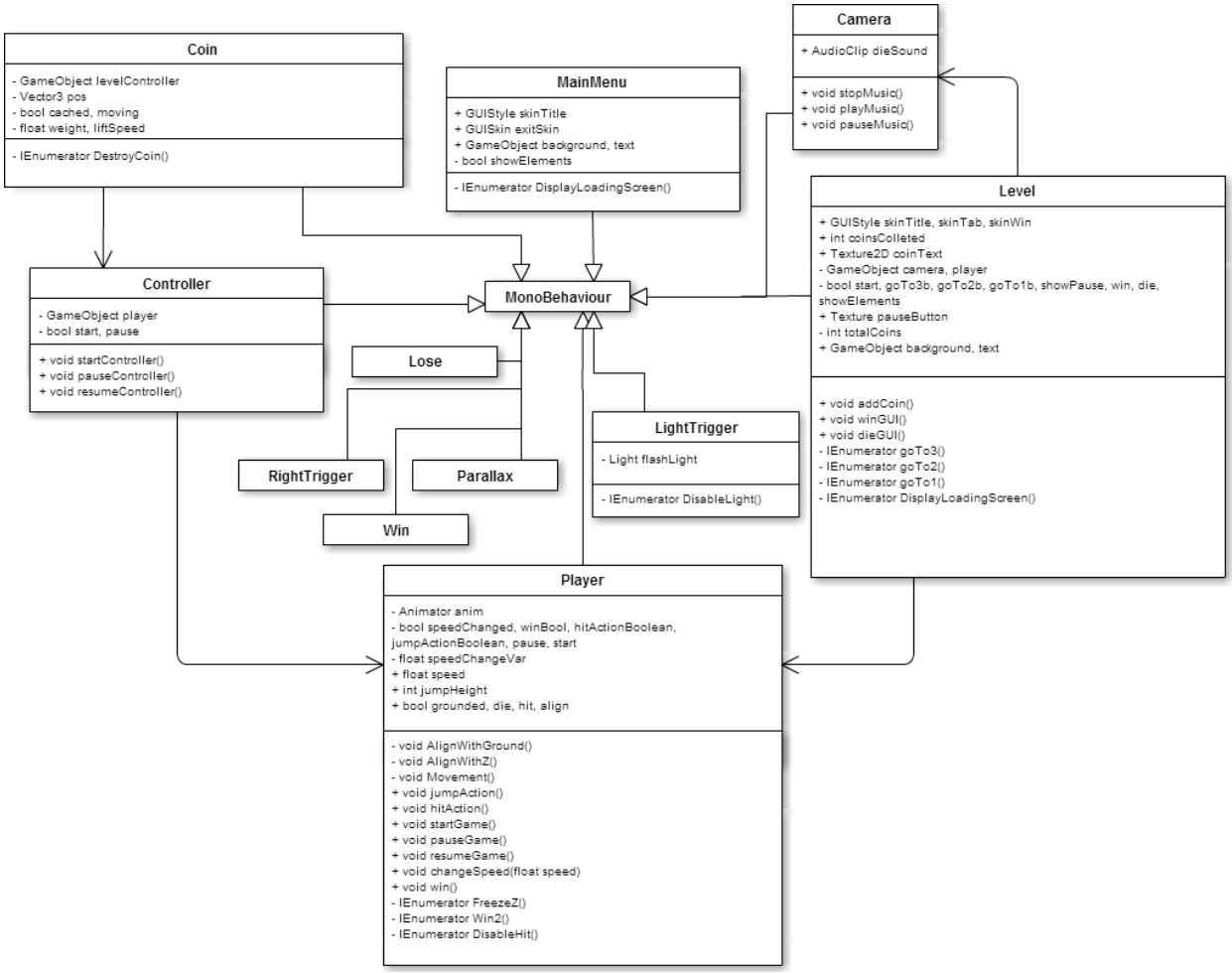


Figura 54: Diagrama de classes de l'aplicació

Player

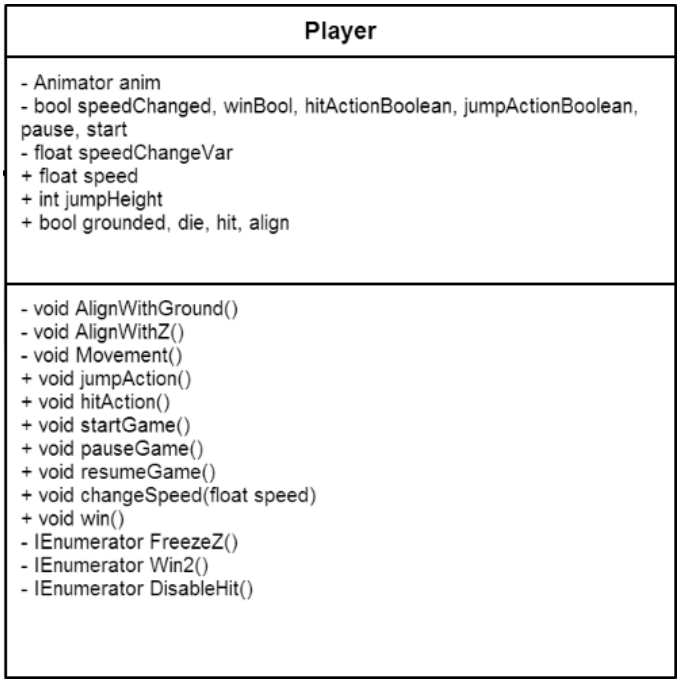


Figura 55: Classe Player

Classe principal amb tota la lògica del personatge. Conté tots el mètodes que permeten al personatge realitzar totes les accions possibles i és una de les classes pivot de tot el projecte. Moltes altres classes accedeixen a aquesta per a executar algun dels mètodes públics. El mètode **Update** d'aquesta classe bàsicament mou el personatge cap endavant i executa els mètodes **Movement** i **AlignWithGround** o **AlignWithZ** depenent del valor de la variable **align**.

- **Animator anim**
  Referència al Component Animator que conté totes les animacions del personatge principal, amb els estats i el diagrama d'estats.
- **bool speedChanged**
  Indica si la velocitat del personatge ha estat canviada o si en canvi manté encara la velocitat inicial.
- **bool winBool**
  Indica que el jugador ha guanyat la partida.
- **bool hitActionBoolean**
  Indica que el jugador ha realitzat l'acció de colpejar.
- **bool jumpActionBoolean**
  Indica que el jugador ha realitzar l'acció de saltar.
- **bool pause**
  Indica que el jugador ha pausat el joc.
- **bool start**
  Indica que el jugador ha iniciat o reprès el joc.
- **float speedChangeVar**
  Conté la velocitat modificada en casa de que s'hagi modificat durant l'execució de la partida.

- **float speed**
  Velocitat inicial del personatge.
- **int jumpHeight**
  Potència de salt del personatge.
- **bool grounded**
  Indica que el personatge està tocant a terra.
- **bool die**
  Indica que el personatge ha mort
- **bool hit**
  Indica que el personatge està en estat de "colpejar"
- **bool align**
  Indica que s'ha de alinear el personatge amb la normal del terra.
- **void AlignWithGround()**
  Alinea el personatge amb la normal del terra.
- **void AlignWithZ()**
  Alinea el personatge amb l'eix Z (vertical).
- **void Movement()**
  Realitza una acció segons el control que estigui activat (saltar o colpejar).

```
si s'ha iniciat la partida
      si no guanyat i no perdut
            si control==saltar i jugador_a_terra
                  saltar()
            altramentsi control==colpejar
                  colpejar()
            fsi
      fsi
fsi
```

- **void jumpAction()**
  Indica al mètode Movement que ha de realitzar l'acció de saltar.
- **void hitAction()**
  Indica al mètode Movement que ha de realitzar l'acció de colpejar.
- **void startGame()**
  Inicia el joc.
- **void pauseGame()**
  Pausa el joc.
- **void resumeGame()**
  Repren el joc després d'una pausa
- **void changeSpeed(float speed)**
  Canvia la velocitat del personatge segons el paràmetre speed.
- **void win()**
  Indica al personatge que ha guanyat la partida.

- **IEnumerator FreezeZ()**
  Mètode asíncron que congela el personatge en l'eix de les Z. (S'executa després de la mort del personatge).
- **IEnumerator Win2()**
  Mètode asíncron que executa la segona part de l'animació de victòria.
- **IEnumerator DisableHit()**
  Mètode asíncron que deshabilita l'animació i l'estat de colpejar.

A part d'aquests mètodes, la classe Player també implementa la funció **OnCollisionEnter(Collision collision)** que permet controlar amb que ha xocat el personatge:

```
si no mort
      si objecte_col·lisionat=terra o moneda o lightTrigger
            si objecte_col·lisionat = terra
                  personatge_a_terra = cert
                  executa_animacio(córrer)
            fsi
      altramentsi objecte_col·lisionat = objecte_destruïble
            si acció = colpejar
                  destruïr_objecte(objecte_col·lisionat)
            altrament
                  morir()
            fsi
      altrament
            morir()
      fsi
fsi
```

La primera comprovació que es realitzat és si el personatge és mort o no, ja que si és aquest cas, el joc queda "aturat" i no permet cap tipus d'intervenció del jugador excepte la d'iniciar una partida nova. En cas de que no sigui així, un condicional decidirà quina és la lògica a seguir.

Primerament es comprova si l'objecte col·lisionat és **potencialment mortal** o no, entenem com a potencialment mortal aquells que poden ocasionar la mort del personatge. Si no ho és, comprovarem si l'objecte és el terra ja que, en aquest cas, el que s'executa és un canvi d'animació, de caure a córrer. En qualsevol altre cas, el propi objecte (monede o llum) és qui s'encarrega de les funcions que s'han d'executar.

Tot seguit, es comprova si l'objecte (que en aquesta branca i les següents només pot ser un objecte potencialment mortal) és un objecte destruïble que es troba enmig de l'escenari. Si és així, es comprova si el personatge estava colpejant ja que, si és així, el que s'ha de fer és destruir aquest objecte i seguir endavant. En cas de que el personatge no estigués colpejant, es provocaria la mort del personatge.

Finalment, només queden objectes que provoquen la mort directa del personatge, que en aquest cas són uns objectes invisibles que es troben per sota del nivell del mapa que ocasionen la mort quan el personatge cau de l'escenari.

Coin

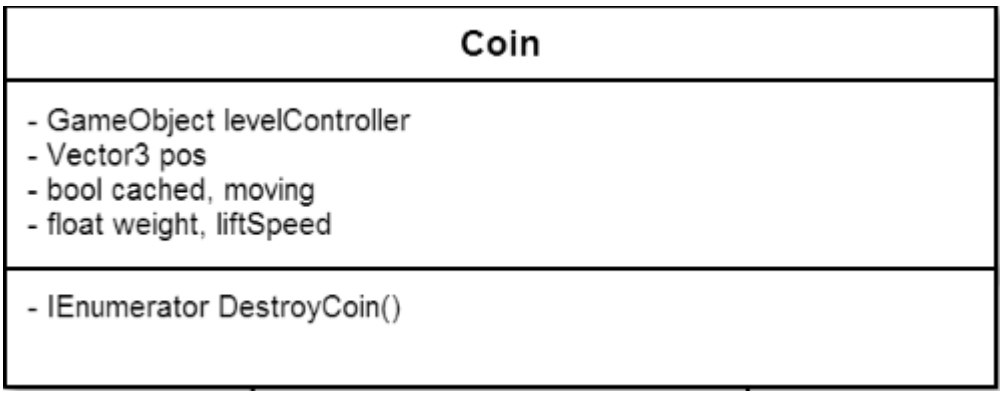


Figura 56: Classe Coin

Aquesta classe controla tot el que està relacionat amb les monedes del joc.

- **GameObject levelController**
  Referència al controlador del joc (Controller) per a poder indicar-li que s'ha aconseguit una moneda.
- **Vector3 pos**
  Posició de la moneda.
- **bool cached**
  Indica si la moneda ha estat agafada o no.
- **bool moving**
  Indica si la moneda està executant l'animació de moviment cap al comptador.
- **float weight**
  Indica la quantitat de distància que s'ha de moure la moneda en l'animació cap al comptador.
- **float liftSpeed**
  Indica la velocitat amb la que s'ha de moure la moneda en l'animació cap al comptador.
- **IEnumerator DestroyCoin()**
  Mètode asíncron que destrueix la moneda, per a alliberar espai de memòria i evitar confusions gràfiques.

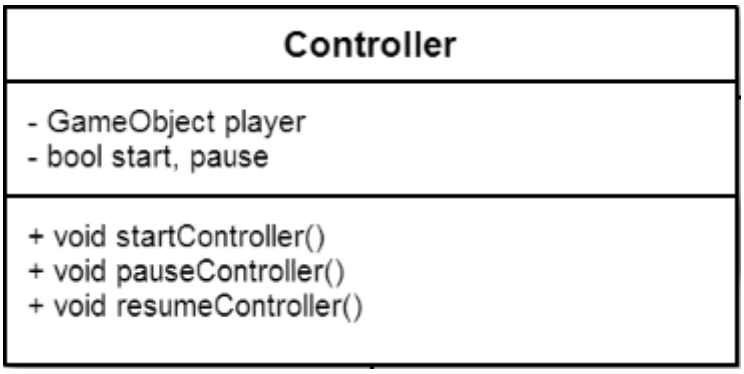


Figura 57: Classe Controller

Classe que s'encarrega d'administrar el control de la pantalla del dispositiu sobre el personatge. El gruix d'aquesta classe es troba dins el mètode **OnGUI** que captura la zona on el jugador ha premut per decidir quina acció ha de realitzar el jugador, que es pot resumir en el següent tall de pseudocodi:

```
si no pausa i partida_iniciada
      si pantalla_premuda
            x = posició_ratolí(x)
            y = posició_ratolí(y)
            si x < ample_pantalla() div 2
                  saltar()
            altramentsi boto_pausa
                  pausa_joc()
            altrament
                  colpejar()
            fsi
      fsi
fsi
```

Si la partida no es troba pausada i a més a més ha estat iniciada és quan es poden capturar events de la pantalla ja que, d'altra manera, no s'executarien i provocarien conflictes amb els botons de la GUI.

Primerament es capturen les coordenades de la posició premuda i, segons aquesta, s'executaria una funció o una altra.

- Si s'ha premut a la part esquerra de la pantalla (entenent com a part esquerra el costat esquerra de la línia imaginaria que divideix l'ample de la pantalla en dues meitats) s'ha de realitzar l'acció de saltar.
- Per altra banda, si es prem el botó de pausa (situat a la part dreta de la pantalla), cal pausar el joc.
- Finalment, tota la zona que queda per cobrir (la part dreta de la pantalla exceptuant la zona del botó de pausa) s'ha de realitzar l'acció de colpejar.

- **GameObject Player**
  Referència al jugador, per a poder realitzar les accions sobre aquest.
- **bool start**
  Indica si el joc ha sigut iniciat o no.
- **bool pause**
  Indica si el joc ha sigut pausat o no.
- **void startController()**
  Inicia el controlador.
- **void pauseController()**
  Pausa el controlador.
- **void resumeController()**
  Reprèn el controlador.

MainMenu

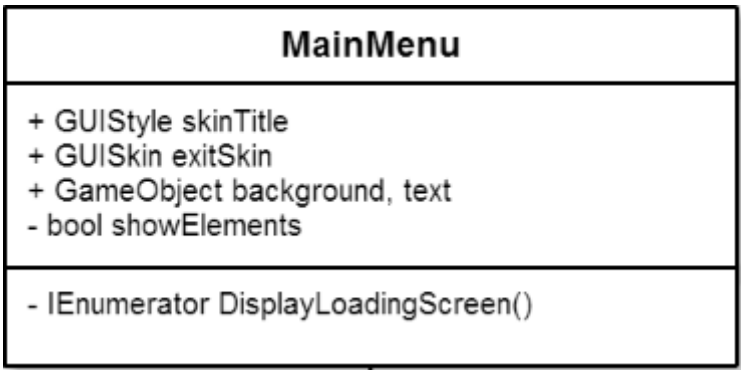


Figura 58: Classe MainMenu

Classe encarregada d'administrar el menú principal del joc. El gruix d'aquesta classe també es troba dins el mètode **OnGUI**.

- **GUIStyle skinTitle**
  Estil que tindrà el títol del menú.
- **GUISkin exitSkin**
  Skin que tindran els altres botons del menú.
- **GameObject background**
  Referència al fons de color negre de la pantalla de càrrega.
- **GameObject text**
  Referència a la paraula *Loading* de la pantalla de càrrega.
- **bool showElements**
  Indica si s'ha de mostrar el menú principal o al pantalla de càrrega.

- **IEnumerator DisplayLoadingScreen()**
  Mètode asíncron que mostra la pantalla de càrrega i carrega el nivell de joc.

Camera

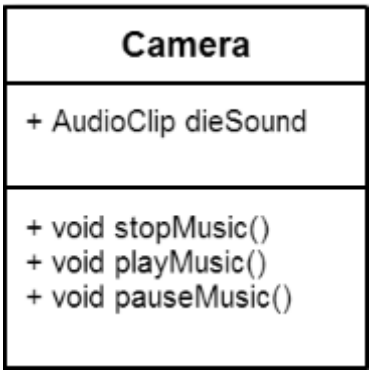


Figura 59: Classe Camera

Classe que administra els sons del joc, que es reprodueixen des de la càmera.

- **AudioClip dieSound**
  Referència al clip de so que es reprodueix quan el personatge mor.
- **void stopMusic()**
  Atura la música de fons.
- **void playMusic()**
  Reprodueix la música de fons.
- **void pauseMusic()**
  Pausa la música de fons.

Level

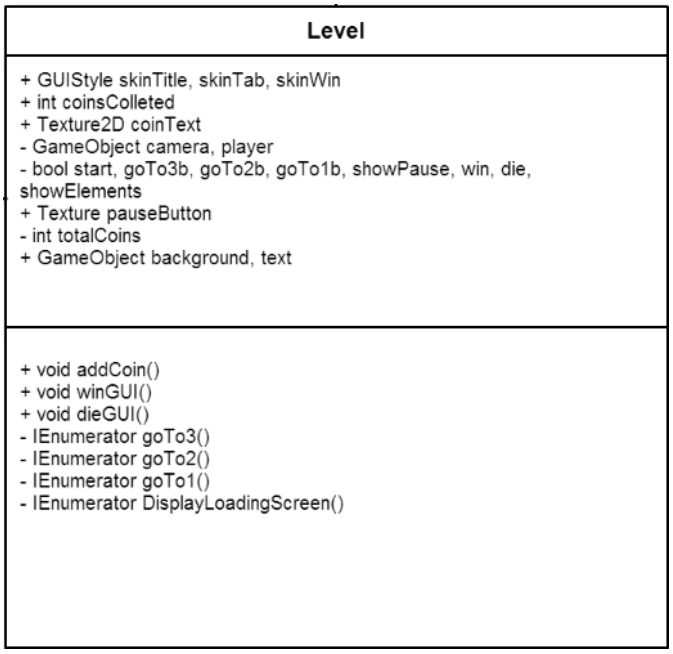


Figura 60: Classe Level

Aquesta classe s'encarrega d'administrar algunes de les funcions més genèriques del joc, com executar la interfície de victòria/derrota o iniciar la partida.

- **GUIStyle skinTitle, skinTab, skinWin**
  Diversos estils per als elements de la interfície d'usuari.
- **int coinsCollected**
  Número de monedes obtingudes.
- **Texture2D cointext**
  Textura de la moneda per a la GUI.
- **GameObject camera**
  Referència a la càmera principal del joc.
- **GameObject Player**
  Referència al personatge.
- **bool start**
  Indica que el joc ha estat iniciat.
- **bool goTo3b, goTo2b, goTo1b**
  Conjunt de variables booleanes que administren el compte enrere previ a l'inici de la partida.
- **bool showPause**
  Indica que el jugador ha premut el botó de pausa.
- **bool win**
  Indica que el jugar ha guanyat la partida.
- **bool die**
  Indica que el personatge ha mort.
- **bool showElements**
  Indica si s'ha de mostrar la interfície d'usuari o la pantalla de càrrega.
- **Texture pauseButton**
  Imatge del botó de pausa.
- **int totalCoins**
  Total de monedes de la pantalla.
- **GameObject background**
  Referència al fons de color negre de la pantalla de càrrega.
- **GameObject text**
  Referència a la paraula *Loading* de la pantalla de càrrega.
- **void addCoin()**
  Afegeix una moneda al comptador.
- **void winGUI()**
  Mostra la interfície de victòria.
- **void dieGUI()**
  Mostra la interfície de derrota.
- **IEnumerator goTo3(), goTo2(), goTo1()**
  Conjunt de mètodes que mostren el compte enrere previ a l'inici de la partida.
- **IEnumerator DisplayLoadingScreen()**
  Mètode asíncron que mostra la pantalla de càrrega i carrega el nivell de joc.

Win / Lose

Aquestes dues classes s'encarreguen de detectar quan el jugador ha arribat a la meta o ha xocat amb un element que provoca la derrota. Consten únicament del mètode **OnTriggerEnter** que s'executa quan el jugador activa el trigger que conté la classe.

RightTrigger

Aquesta classe s'encarrega de canvia el mètode d'alineament del personatge principal, canviant entre alinear-lo amb la normal del mapa o amb l'eix Z. Conta únicament del mètode **OnTriggerEnter** que canvia de normals a Z i de **OnTriggerExit** que canvia a la inversa. Es fa servir per els salts entre plataformes.

LightTrigger

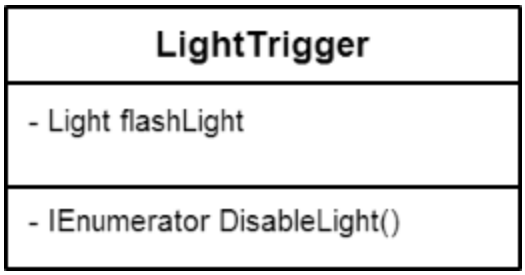


Figura 61: Classe LightTrigger

Aquesta classe s'encarrega d'encendre i apagar els llums de l'escenari.

- **Light flashLight**
  Referència al GameObject *Light* que il·lumina l'escenari.
- **IEnumerator DisableLight()**
  Mètode asíncron que apaga el llum al cap d'un temps.

```
esperar(0.2 segons)
flashLight.encesa = fals
```

Parallax

Aquesta classe s'encarrega única i exclusivament del moviment de la imatge de background, que dóna una sensació de moviment i profunditat. A les figures 62 i 63 es pot veure aquest moviment del fons en dos instants de temps diferents

Figura 62: Parallax 1

Figura 63: Parallax 2

Implantació i resultats

Legislació i normativa vigent

El projecte desenvolupat no presenta cap problema en aspectes legislatius. No s'ha tingut en compte la llei orgànica de protecció de dades de caràcter personal (LOPD) ja que el sistema en cap moment tracta cap tipus de dades relatives a l'usuari. Des del punt de vista de seguretat, no hi ha cap requisit de control d'accés al programa, ja que es tracta d'una aplicació on els usuaris que accedeixen només tenen un rol.

I pel que fa a la llei de serveix de la societat de la informació i comerç electrònic (LSSICE), el projecte no constitueix una activitat econòmica tot i haver-hi afegit un mòdul d'AdSense de Google per a poder visualitzar anuncis.

Screenshots del videojoc

A continuació es mostren captures del videojoc desenvolupat.

Figura 64: Pantalla principal

A la figura 64 es pot veure el menú principal amb els diferents botons per accedir a les diferents funcionalitats.

Figura 65: Pantalla d'inici de partida

A la figura 65 es pot veure un inici de partida, a on se li demana al jugador que premi la pantalla per a iniciar una partida.

Figura 66: Imatge d'una partida en curs

A la figura 66 es pot veure una captura d'una imatge en curs, a on es pot veure clarament els diferents elements del joc.

Figura 67: Captura de derrota

A la figura 67 es pot observar el que es mostra per pantalla en cas de que el jugador hagi perdut la partida.

Figura 68: Menú de pausa

A la figura 68 es pot observar com és el menú de pausa, a on es pot visualitzar clarament que hi ha la possibilitat de reprendre la partida o de tornar al menú principal.

Figura 69: Captura de victòria

A la figura 69 es pot observar una captura de victòria on, malauradament, no s'ha aconseguit cap moneda.

Figura 70: Captura del joc en dispositiu mòbil

A la figura 70 es pot veure com queda el joc en un dispositiu mòbil a on sí que apareixen anuncis per part de Google. La posició d'aquests és la idònia per a no molestar al jugador durant la partida.

Implantació a Google Play

Com que es tracta un joc principalment dissenyat per a dispositius mòbils, la inclusió a algun dels *Markets* d'apps mòbils es tasca obligada. En aquest apartat s'explicarà quins són els passos per a una correcta pujada a aquest.

El primer que cal és inscriure's amb un compte de Google com a desenvolupadors i per això cal accedir a la url https://play.google.com/apps/publish/signup/ i loguejar-nos amb el nostra compte de Google.

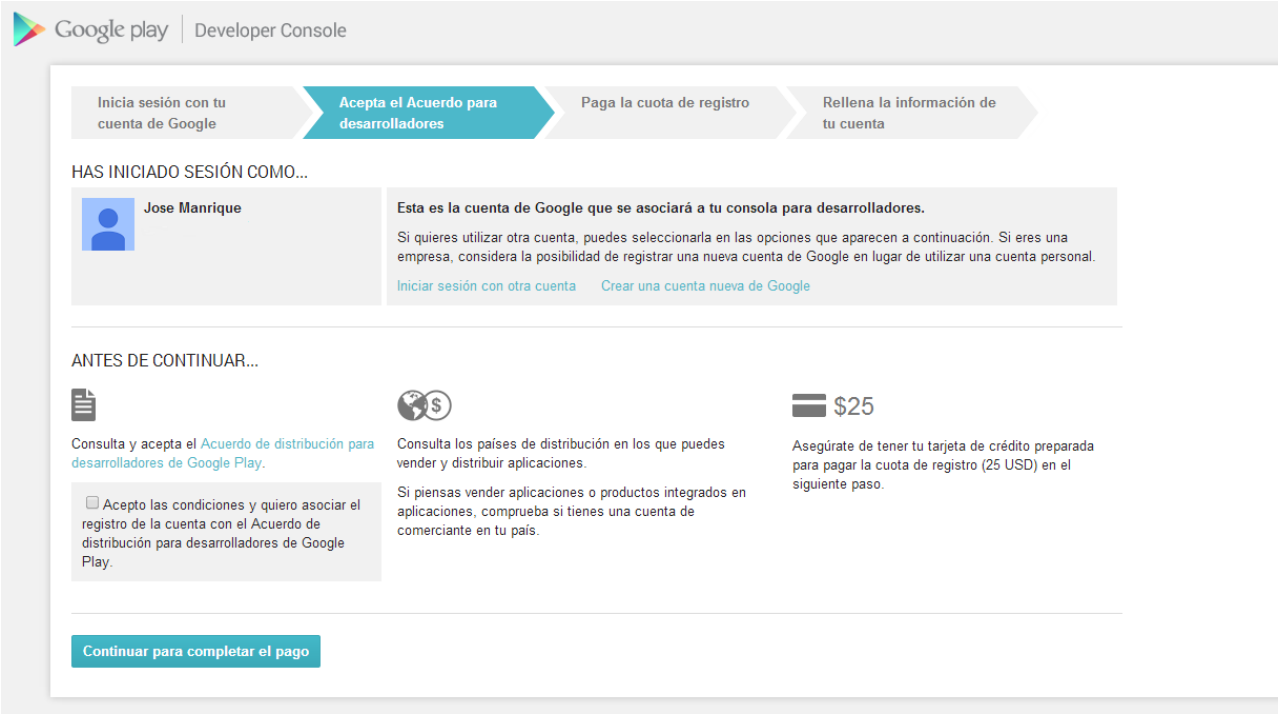


Figura 71: Pàgina d'inici de la consola de desenvolupadors de Google

En cas de que sigui la primera vegada (figura 71), caldrà pagar 25$ per a poder crear-nos la nostra compta de desenvolupadors (el pagament és únic i vàlid per sempre). Quan hàgim realitzat el pagament, accedirem a la consola de desenvolupadors (figura 72).

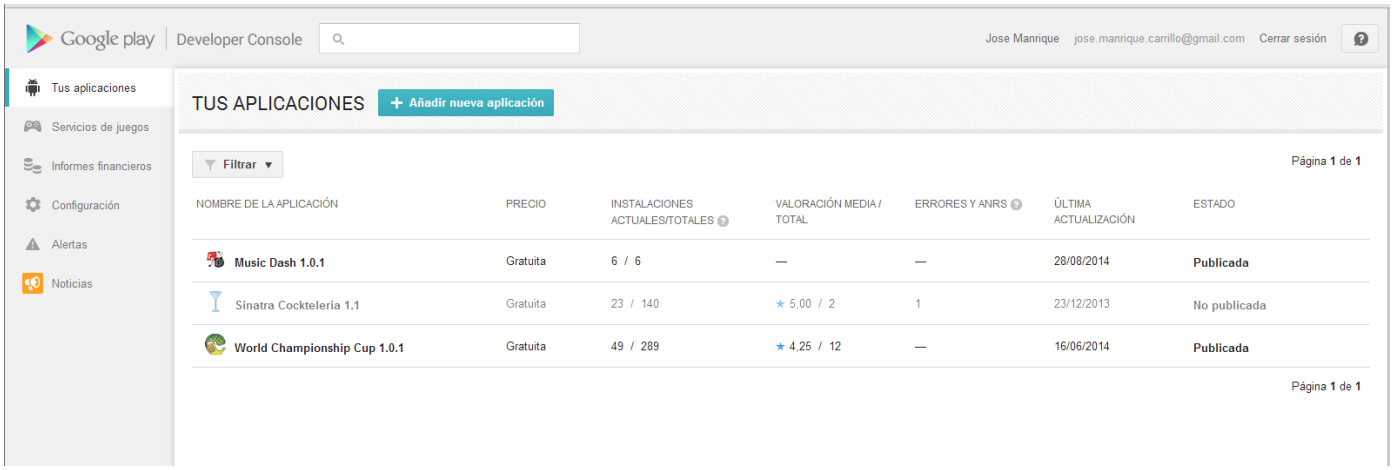


Figura 72: Consola de desenvolupadors de Google Play

En cas de que sigui la primera vegada que accedim, el llistat d'aplicacions serà òbviament buida. Per a pujar una nova aplicació, caldrà clicar sobre el botó **+Añadir nueva aplicación** (o la seva traducció en l'idioma que tinguem al compte de Google) i accedirem a la següent figura (número 73).

Figura 73: Imatge de creació d'una nova aplicació

En aquest pop-up haurem d'escollir l'idioma predeterminat de l'aplicació (es poden afegir tants idiomes com vulguem més endavant), un nom per a aquesta i, si tenim ja el fitxer .apk preparat (fitxer compilat d'Android) el podem pujar. Sinó, aquest pas es pot realitzar més endavant. Tot seguit emplenarem les dades de la nostra aplicació.

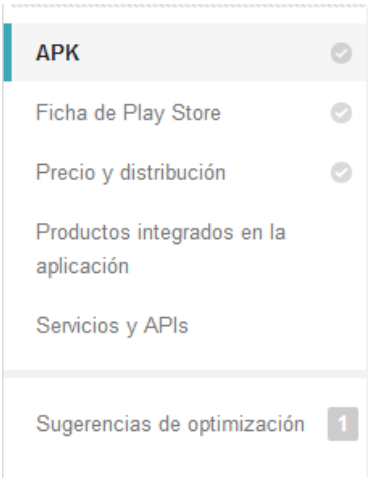


Figura 74: Menú esquerra de la consola de Google Play

Figura 75: Menú interior de la consola de Google Play

El menú de més a l'esquerra (figura 74) no ens interessa de moment més enllà de la primera opció (**Tus aplicacions**), ja que es tracta de configuracions de compta, serveis de jocs (accedir als cercles de Google+) i d'informes financers. En aquesta memòria no es tractaran aquests apartats.

En el següent menú (figura 75) hi ha tots els apartats que cal emplenar per a que la aplicació pugui ser pujada al market, aquests són:

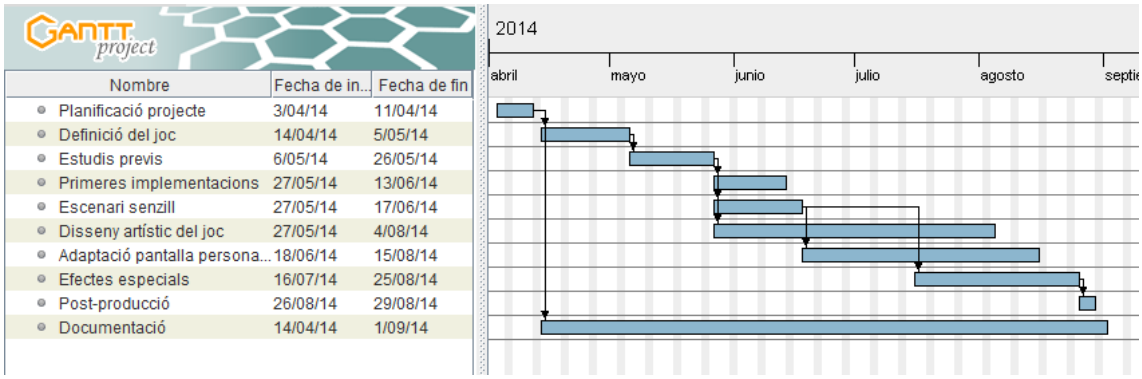
- **APK**: Aquí es on pujarem el nostra arxiu APK per primera vegada o l'actualitzarem si fos el cas. Existeix la possibilitat de crear una aplicació Beta o Alpha per a que només un grup de persones (indicades en la consola amb l'e-mail) puguin descarregar-se-la i probar-la.
- **Ficha de Play Store**: En aquest apartat hi afegirem totes les dades i imatges que es mostraran a la fitxa:
    - Títol de l'app (obligatori).
    - Descripció curta (obligatori).
    - Descripció llarga (obligatori).
    - Captures de pantalla de mòbil i tauletes de 7 i 10 polzades (obligatòria almenys una).
    - Icona d'alta resolució de 512x512 píxels (obligatori).
    - Imatge destacada de 1024x500 píxels (obligatori).
    - Imatge promocional de 180x120 píxels.
    - Video promocional (link a Youtube).
    - Seleccionar un tipus d'aplicació (Aplicació o Joc) (obligatori).
    - Categoria de l'aplicació (en el cas de Joc, indicar si és de carreres, de música, arcade, educatiu, etc) (obligatori).
    - Classificació de contingut (indicar per a quin tipus de persona està recomanada, per a més informació https://support.google.com/googleplay/android-developer/answer/188189) (obligatori).
    - Dades de contacte del desenvolupador (obligatori excepte el telèfon).
    - Política de privacitat (cal indicar-ne una o indicar que no per ara no n'hi ha cap).
- **Precio y distribución:** Aquí caldrà indicar si la aplicació serà gratuïta o de pagament i indicar en quins països es podrà descarregar, poden filtra per país i per companyia de telèfon. També cal indicar-li el consentiment pel contingut i les lleis d'exportació d'EE.UU.
- **Productos integrados en la aplicación:** Aquest apartat només serà útil en el cas de que la nostra aplicació disposi d'in-app purchases[6].
- **Servicios y APIs:** Com indica el nom, es un apartat a on s'administraran els diferents serveis que pot tenir una aplicació, des del servei GCM (Google Cloud Messaging, per a l'enviament de missatges push) fins a llicències i facturacions integrades a aplicacions.

[6] Micropagaments, possibilitat de comprar codis, monedes o crèdits per a jugar amb diners reals

Conclusions

Temporalització

Difícilment la temporització planificada inicialment en un projecte es correspon fidelment a la que s'aplica a la realitat. La temporització real del temps dedicat és tal com a es mostra en la Figura 76.



Nombre	Fecha de in...	Fecha de fin
Planificació projecte	3/04/14	11/04/14
Definició del joc	14/04/14	5/05/14
Estudis previs	6/05/14	26/05/14
Primeres implementacions	27/05/14	13/06/14
Escenari senzill	27/05/14	17/06/14
Disseny artístic del joc	27/05/14	4/08/14
Adaptació pantalla persona...	18/06/14	15/08/14
Efectes especials	16/07/14	25/08/14
Post-producció	26/08/14	29/08/14
Documentació	14/04/14	1/09/14

Figura 76: Planificació final

Tot i que la llargada total del projecte respecte el temps no s'ha allargat. Es pot observar com algunes tasques s'ha invertit més temps de l'esperat i algunes altres es solapen entre elles.

Conclusions

Durant l'elaboració d'aquest Projecte Final de Carrera s'han arribat a varies conclusions:

- És molt important usar les eines adequades i realitzar un bon anàlisis dels problemes que poden sorgir. Tot i que suposi una inversió de temps extra, es nota en el resultat final i ajuda a solucionar molts problemes.
- Realitzar un videojoc no és només modelar i picar codi. Possiblement el moment més important en la creació d'un videojoc és la fase de dissenyar els elements i les regles que el regeixen, o sigui, realitzar un bon GDD (Game Document Design). Aquesta feina en un desenvolupament professional la porta a terme un Game Designer.
- Una persona sola no pot desenvolupar un gran projecte. En el desenvolupament d'un videojoc hi intervenen persones amb diferents rols, com per exemple, el dissenyador gràfic i el compositor.

Treball futur

El moment de finalitzar aquest projecte no significa que finalitzi el desenvolupament d'aquest videojoc. Com a treball futur es podria subratllar que:

- Es pot implementar algun sistema per a que el joc recreï automàticament la pantalla a partir d'alguna cançó que hi afegeixi el jugador. Això ens donaria dues avantatges: per una part, ens oblidaríem de tot el tema de llicències i drets d'autor en l'àmbit de la música perquè nosaltres no seriem responsables de les cançons que l'usuari escull per a jugar, i per altra banda obtindríem un joc amb una rejugabilitat gairebé infinita perquè sempre es podria trobar una nova cançó que originés una nova pantalla completament diferent a l'anterior. Aquest desenvolupament ara per ara és impensable amb el temps de durada del projecte, però qui sap si amb un any o dos de desenvolupament es podria crear alguna eina que permetés aquesta funcionalitat.
- Buscar col·laboradors: A part de desenvolupar l'eina anterior, es pot contacta amb gent que necessiti donar a conèixer la seva obra (ja sigui música o dibuix i animacions) i poder-les incloure en el joc a canvi de la publicitat.
- Analitzar el seguiment del joc mitjançant Google Play, Google Analytics i Google Admob, per veure a quins països es juga més o quant dura una partida de mitjana.
- Millorar l'apartat gràfic. Com que no sóc dissenyador artístic, aquest àmbit és el que ha quedat més coix de l'aplicació, però s'ha realitzat de tal manera que no costaria gens canviar les imatges i animacions per unes altres en qualsevol moment.

Bibliografia

En aquesta secció es mencionaran totes les fonts consultades per a la realització del projecte. Tota la informació consultada ha estat a Internet, per tant, l'únic tipus de fonts són pàgines web i pdfs en línia.

Asociación Española de Videojuegos – www.aevi.org.es, *Balance Económico 2013 Industria Española del Videojuego*, 2014

Wikipedia - Atari, 1 de Setembre de 2014 - http://es.wikipedia.org/wiki/Atari

VidaExtra – El consumo en el sector de los videojuegos en España desciende de nuevo en 2013, 1 de Setembre de 2014 - http://www.vidaextra.com/industria/el-consumo-en-el-sector-del-videojuego-en-espana-desciende-de-nuevo-en-2013

CincoDias – La industria delñ videojuego facturó 762 millones en 2013, un 7'3% menos, 1 de Setembre de 2014 - http://cincodias.com/cincodias/2014/03/24/tecnologia/1395676554_881512.html

NeoTeo – Historia de los juegos de música, 1 de Setembre de 2014 - http://www.neoteo.com/historia-de-los-juegos-de-musica/

Wikipedia – Scrum, 1 de Setembre de 2014 - http://es.wikipedia.org/wiki/Scrum

Wikipedia – Desarrollo àgil de software, 1 de Setembre de 2014 - http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software

Wikipedia – Google Analytics, 1 de Setembre de 2014 - http://es.wikipedia.org/wiki/Google_Analytics

StackOverflow – diversos posts de dubtes - http://stackoverflow.com/

Unity Community – diversos posts de dubtes - http://forum.unity3d.com/

The Sprites Resource, 1 de Setembre de 2014 - http://www.spriters-resource.com/

Free Music Archive, 1 de Setembre de 2014 – http://freemusicarchive.org/

MusOpen, 1 de Setembre de 2014 - https://musopen.org

Annexos

GDD – Game Document Design

El GDD és un document de disseny que es realitza abans del desenvolupament d'un projecte de videojocs. Acostuma a posar-se tot el que es voldria que tingués el videojocs, encara que al final no tot s'hi hagi inclòs. A continuació hi ha el GDD d'aquest projecte, el qual es va escriure durant la fase de disseny del joc.

Introducció

El joc consistirà en un joc de plataformes amb scroll-automàtic a on el jugador haurà d'esquivar i destruir els obstacles que se li presentaran durant la pantalla amb l'objectiu d'acabar-la i aconseguir el màxim de punts possibles.

La particularitat d'aquest joc respectes als altres que ja existeixen en el mercat és que la pantalla estarà completament centrada en la interacció dels elements d'aquesta i la música que sona.

Les rampes, els obstacles, les roques, els objectes destruïbles... tot estarà concordat i sincronitzat amb la música de fons, amb la banda sonora de cada pantalla. Els objectes decoratius, enemics de fons, explosions, llums i efectes de colors també estaran relacionats amb música.

Gameplay

Aquest joc es desenvoluparà en un entorn senzill i fàcil de jugar per part del jugador, ja que només hi haurà dos possibles moviments: saltar i destruir (s'entén per destruir l'efecte de colpejar alguna cosa destruïble).

Les pantalles es desenvoluparan en un entorn 2D amb scroll-automàtic, és a dir, el jugador no tindrà cap tipus de control sobre el moviment del personatge, no podrà controlar la velocitat ni la direcció de moviment, només podrà controlar el salt i l'acció de colpejar.

Sistema

El joc està desenvolupat per a plataformes mòbils Android i iOS, ja que un joc senzill com aquest té molta sortida en el mercat de les apps.

No obstant, també hi ha la possibilitat de desenvolupar-lo cross-plattform amb PC, ja que no hi ha cap complexitat en exportar els moviments tàctils d'aquest joc a control teclat/comandament.

Controls

Per a dispositius mòbils el joc es jugarà de forma que el dispositiu quedi en posició *landscape* (la part més ampla de l'aparell queda en posició horitzontal) i la pantalla estarà "dividida" en dues zones d'acció: la meitat dreta i la meitat esquerra.

Amb controls personalitzables, el joc funcionarà de forma que una meitat de la pantalla servirà per saltar i l'altra meitat servirà per colpejar.

Si el joc s'exportés a PC, els controls serien tan senzills com assignar un botó per saltar (ja sigui de teclat, ratolí o gamepad) i un altre per colpejar.

Càmera

La càmera del joc estarà permanentment centrada en el personatge, donant una sensació de finestra per la qual veiem el que fa el personatge.

Moviments del personatge

Com s'ha comentat abans, el personatge tindrà bàsicament dues accions, saltar i colpejar, però implícitament el personatge tindrà més accions que serien més o menys automàtiques:

- Saltar

  Acció que s'efectuarà quan el jugador premi el control de saltar.

- Colpejar

  Acció que s'efectuarà quan el jugador premi el control de saltar.

- Lliscar

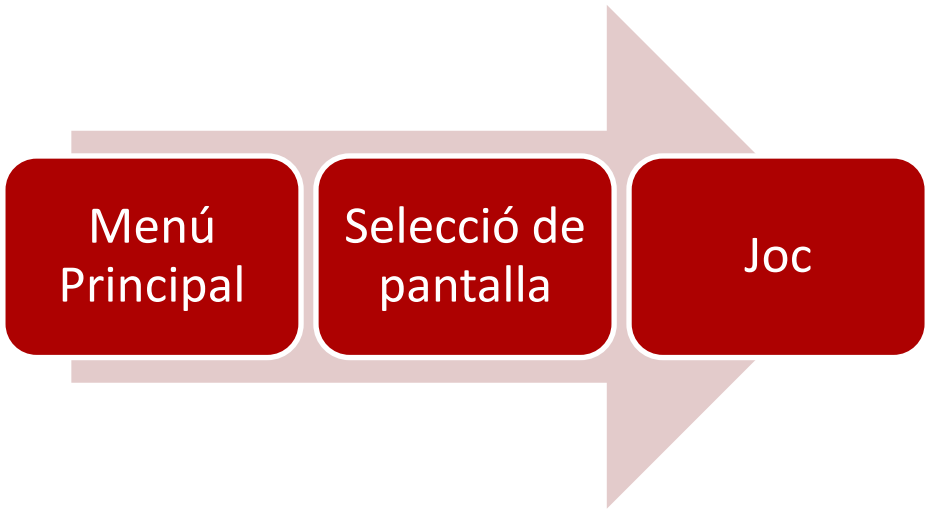
  Acció que s'efectuarà quan el personatge es mogui per una plataforma inclinada cap avall (automàtica).

- Colpejar a l'aire

  Acció que s'efectuarà quan el jugador premi el control de saltar seguit de colpejar.

Estructura del joc

El joc s'estructura de la següent forma:



Manual d'usuari i/o instal·lació

Interfície de joc

A part de totes les pantalles inclòses dins l'apartat Implantació i resultats, a la figura 77 es poden veure els controls tàctils del joc:

Figura 77: Controls tàctils del joc

Instal·lació

La instal·lació del joc (actualment únicament disponible per Google Play (https://play.google.com/store?hl=es) es realitza automàticament mitjançant el cercador amb les paraules **Music Dash** o, en el cas de que no aparegués, amb el nom del desenvolupador **Jose Manrique**. El link de descàrrega és https://play.google.com/store/apps/details?id=com.ManriqueJose.MusicDash&hl=es.