# PROJECTE FI DE GRAU
# VIDEOJOCS EDUCATIUS

**ALUMNE:** MAURO MICÓ EBRO
Grau d'Enginyeria en Informàtica

**CONSULTORS:**
HELIODORO TEJEDOR NAVARRO
JORDI DUCH GAVALDÀ

GENER DE 2013

**He de donar les gràcies a SergioHC,
també a Jordi pel disseny de l'ànec,
i sobretot a Andrea i Visi.**

Aquesta obra està subjecta a una llicència de Reconeixement-NoComercial-SenseObraDerivada 3.0 Espanya de Creative

**GNU Free Documentation License (GNU FDL)**

Copyright © 2013 MAURO MICÓ EBRO.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU Free Documentation License".

**Copyright**

© MAURO MICÓ EBRO
Reservats tots els drets. Està prohibit la reproducció total o parcial d'aquesta obra per qualsevol mitjà o procediment, compresos la impressió, la reprografia, el microfilm, el tractament informàtic o qualsevol altre sistema, així com la distribució d'exemplars mitjançant lloguer i préstec, sense l'autorització escrita de l'autor o dels límits que autoritzi la Llei de Propietat Intel•lectual.

**FITXA DEL TREBALL FINAL**

| | |
|---|---|
| **Títol del treball:** | DISSENY VIDEOJOC EDUCATIU |
| **Nom de l'autor:** | MAURO MICÓ EBRO |
| **Nom del consultor:** | HELIODORO TEJEDOR NAVARRO<br>JORDI DUCH GAVALDÀ |
| **Data de lliurament (mm/aaaa):** | 01/2013 |
| **Àrea del Treball Final:** | |
| **Titulació:** | Grau d'Eng. Informàtica. |

**Resum del Treball:**

Videojoc educatiu orientat a nens entre edats de 3 i 4 anys on l'objectiu principal consisteix en associar objectes que apareixen a l'inici de cada nivell fent servir el control d'un personatge de manera tàctil.El joc s'ha realitzat utilitzant la tecnologia AndEngine d'Android per a dispositius mòbils.

A l'inici de cada nivell apareixen cinc objectes (imatges) a la part esquerra de la pantalla posicionats en una pila verticalment. Posteriorment comencen a aparèixer objectes similars des de la part superior de la pantalla que cauen i reboten contra obstacles seguint un patró donat per la gravetat del planeta plutó. El jugador ha d'anar tocant (associant) els objectes que cauen amb els que apareixen a la pila vertical i esquivar els objectes que són diferents. Cada vegada que el jugador toca un objecte que no correspon amb el que apareix a la pila vertical, es perd energia. Si es perd tota l'energia es torna a començar en el nivell en què es trobava el jugador. Tal com es van associant els objectes que són iguals de la pila vertical, aquests desapareixen i es passa al següent nivell quan s'acabi correctament tota l'associació. Cada escena està formada per obstacles diferents i cada dos nivells apareixen objectes d'associació nous. El joc s'acaba quan s'han esgotat els nivells.

**Abstract:**

Educational videogame intended for children aged 3 to 4 whose main objective is to associate objects that appear at the beginning of each level using a player control by touch.

At the beginning of each level there are five objects (images) on the left side of the screen positioned vertically. Then similar objects begin to appear from the top of the screen that fall and bounce off obstacles in a pattern given by the gravity of the planet Pluto. The player must be touching (associating) falling objects with those in the vertical stack and dodge the objects that are different. Each time the player touches an object that does not match the one in the vertical stack, it loses energy. If all the energy is lost, the game starts again at the level in which the player was. As the objects that are the same as those in the vertical stack, these disappear and you pass to the next level when you have completed the whole matching correctly. Each scene consists of different obstacles and objects appear every two levels of new association. The game ends when there are no more levels to play.

**Paraules clau:**

Videojoc, educatiu,AndEngine,Android,mòbil,Box2d,tàctil,gravetat.

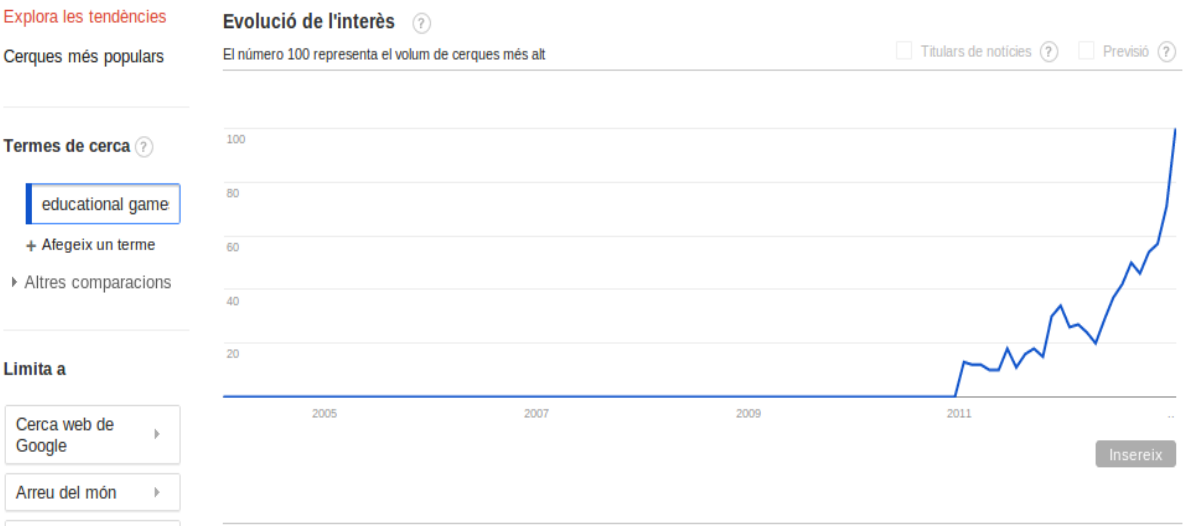
# Índex

1. Introducció....1
1.1 Context i justificació del treball....1
1.2 Objectius del treball....2
1.3 Enfocament i mètode seguit....4
1.4 Planificació del projecte....5
1.5 Breu sumari de productes obtinguts....7
2. Disseny....7
3. Tecnologies utilitzades....11
4. Programació de l'aplicació....12
4.1 Estructura bàsica del projecte....12
4.2 Disseny de classes en el cicle de vida del joc....13
4.3 Aspectes més rellevants....17
4.4 Disseny de sprites. Textures....23
4.5 Disseny de mapes. TiledMap....30
4.6 Les classes....32
5 Conclusions....36
6 Glossari....39
7 Bibliografia....40
8 Annexos....41

# 1. Introducció

## 1.1 Context i justificació del treball

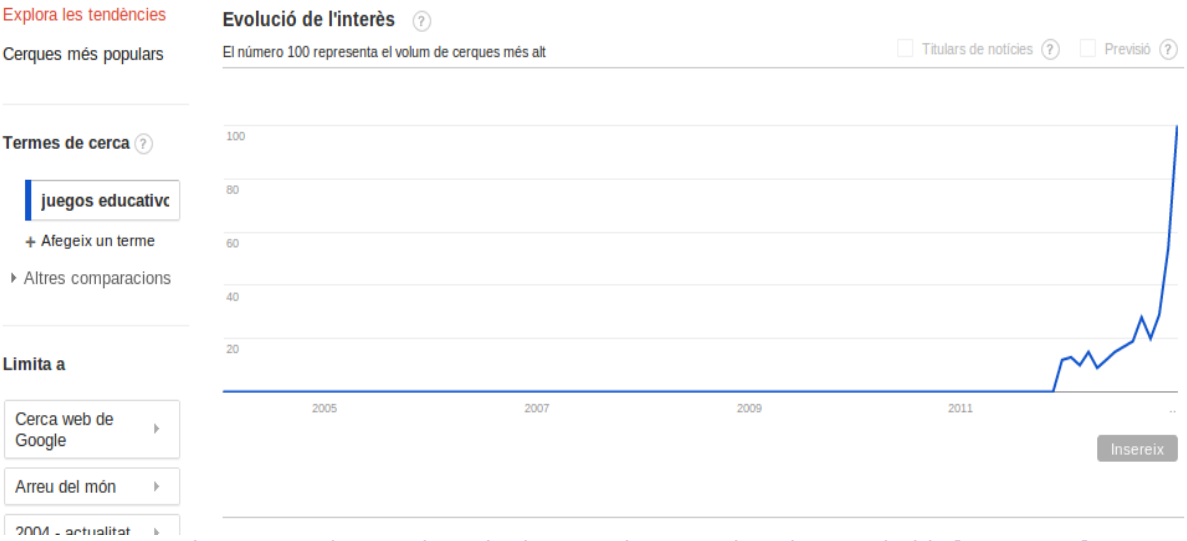
Un videojoc educatiu es presenta com una eina d'aprenentatge, que pot estar enfocada a diferents segments de la societat. Cada tipus de videojoc pot tenir diferents funcions com la transmissió de continguts i valors, adquisició d'habilitats, aprenentatge, potenciar habilitats psicomotrius, exercir la memòria ... podem parlar de jugar per aprendre.
Si a més parlem de dispositius mòbils o de tablets, ens adonarem que es tracta d'un mercat que no para de créixer a tot el món i el seu valor no s'atura. La mobilitat s'ha convertit en la protagonista del món tecnològic en l'actualitat. La unió d'aquests dos conceptes ens pot conduir a un mercat potencial d'explotació com són el de les aplicacions (disseny de videojocs educatius) enfocades a aquest tipus de dispositius mòbils.
Fent un petit anàlisi sobre les tendències dels videojocs per nens utilitzant la tecnologia dels mòbils amb Android, ens podem adonar de l'interès per aquest tipus d'aplicacions. Per tal d'obtenir més informació sobre això, s'han elaborat unes dades utilitzant google trends utilitzant els termes de cerca clau: “educational games android” i "android education apps”.
El resultat per països ens mostra en la primera cerca, que no hi ha un volum prou alt com per mostrar els resultats per pais, encara que la gràfica en decembre de 2012 és de 71 sobre 100. En la segona ens dona un resultat sorprenent: el país amb el volum de consultes més significatiu són els EUA, país capdavanter en consum i ús de les noves tecnologies amb unes dades de 100 sobre 100 segons Google Trends.




Consulta en Google Trends amb el terme ' educational games android'. [4-01-2013]

Una altra dada important, com es pot observar en la gràfica és que el volum de consultes comença a créixer a mitjans de l'any 2011, aconseguint la zona més alta a finals de 2012.

Si fem la mateixa recerca utilitzant el terme “juegos educativos android”, obtenim també que encara no hi ha un volum de dades suficient com per donar dades sobre els països amb aquesta tendència, tot i que el volum que ens mostra la gràfica podem observar que es comencen a utilitzar aquests termes de cerca a principis de l'any 2012 aconseguint major auge entre novembre i desembre d'aquest any.



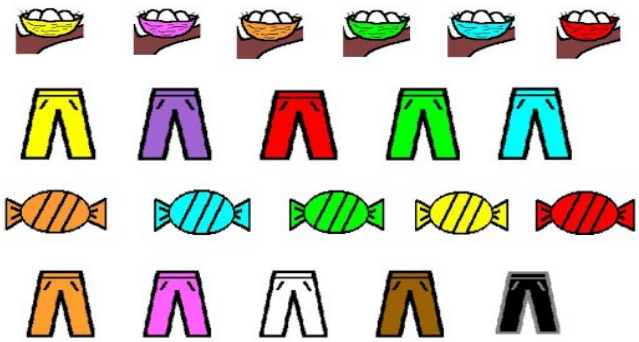

Consulta en Google Trends amb el terme ' juegos educativos android'. [4-01-2013]

## 1.2 Objectius del treball

L'objectiu d'aquest projecte ha estat des de l'inici un aprenentatge continu per assolir les habilitats en la matèria de videojocs educatius que comporten al desenvolupament d'un joc en plataformes compatibles amb versions d'Android igual o superiors a la 2.2.

Donada la baixa experiència inicial en el desenvolupament d'aquest tipus de projectes, es va haver de dedicar un temps extra addicional, abans d'emprendre el projecte, en l'aprenentatge d'un tipus de tecnologia que s'adaptés a l'entorn d'un projecte de vídeo joc educatiu orientat als mòbils i tablets.

La idea base del projecte va ser en un principi, el desenvolupament d'un joc educatiu basat en associacions d'objectes per poder-los discriminar de forma visual i psicomotriu, orientat a nens amb edats compreses entre els 3 i 4 anys (2º cicle d'infantil).



**Exemple d'associació per colors. Font:**http://colorearyaprender.com/juego-de-asociacion-por-colores-para-ninos

Aquest tipus de jocs els podem trobar amb facilitat per ser jugats en un ordinador a través d'Internet, però si es busca a Internet per aquest tipus de jocs, hi ha pocs jocs educatius enfocats a nens d'aquestes edats en el mercat dels jocs per a mòbil.

Llavors, fent una cerca en Internet sobre videojocs educatius basats en associacions, podem trobar molts tipus de jocs.Si ens adonem, podrem observar que la majoria d'ells es basen en *associacions estàtiques basats en fitxes o trencaclosques*, on les figures o objectes d'associació queden fixos a la pantalla i en la major part d'ells no existeix moviment.Un exemple molt conegut d'aquest tipus de jocs educatius és el JClic.

Pert tant, un dels objectius bàsics en què es basa aquest projecte de joc és el de l'associació d'objectes utilitzant el moviment i així poder donar-li més acció al joc.

Podríem definir el projecte com un videojoc "arcade" on apareixen gràfics poc complicats, la dificultat augmenta progressivament amb cada pantalla superada, i on no hi ha profunditat quant a història, mitologia, personalitat dels personatges, etc.

A través d'aquest tipus de jocs s'ofereix un altre ventall de possibilitats diferents a les actuals en les quals podem potenciar:

- el desenvolupament motor:
  - coordinació oculo manual (coordinació de la mà i l'ull): la coordinació oculo manual implica l'exercici de moviments controlats i deliberats que requereixen de molta precisió. La posició que ha d'adoptar el dit al ScreeNControl en relació amb la posició visual que ocupa l'objecte en pantalla (personatge).
  - augmentar la precisió en les tasques de motricitat fina: el domini de la precisió amb la que el nen ha d'utilitzar el dit en el ScreeNControl.
- el desenvolupament cognitiu:
  - millorar la capacitat d'anticipar les conseqüències de les pròpies accions.

- augmentar la capacitat d'atenció.
- augmentar la curiositat.
- millorar la capacitat d'interpretació simbólic
- millorar en l'ús general de les noves tecnologies.

La idea o objectiu principal del joc és que el nen desenvolupi la capacitat de memòria a curt termini.

## 1.3 Enfocament i mètode seguit

Desenvolupar un joc no és una tasca fàcil. Durant tot el procés de creació s'han de seguir unes etapes com són la idea del joc, el disseny, la planificació, el desenvolupament i les proves d'aquest. Com a norma general aquest procés sol estar dividit en equips de treball, sobretot per programadors i dissenyadors. Aquest projecte parteix d'una idea pròpia, on tot el procés de creació recau sobre una mateixa persona.
La principal idea del projecte es basa en un desenvolupament que parteix de zero. Estem parlant d'un producte nou. *A l'annex es pot observar una imatge del croquis del joc abans d'iniciar el projecte.*
Per aquest motiu, tot el disseny del mateix ha estat creat únicament com a fase de proves, per poder seguir més ràpidament l'evolució del projecte,i centrar el projecte més detingudament en la fase de la programació. Per tant és important indicar que el disseny de mapes, escenes, sprites, títols o menús, fins al so, s'han confeccionat de forma genèrica, fins i tot en alguns casos de forma molt ràpida, per poder avançar en la part de la programació. Cal dir que a la fase final del projecte, aquestes "plantilles" de disseny s'haurien de modificar per part d'un bon dissenyador per donar més realisme al joc.Serà en la fase de proves, en la qual no s'ha arribat ni de bon tros, on s'haurà d'examinar i on es modificarà o posicionarà tot el disseny del mateix.
La fase inicial del projecte es basava en la creació d'un joc educatiu pensat per desenvolupar en els nens l'aprenentatge de les lletres per poder-les discriminar de manera visual i auditiva, i pel que en un principi la idea base era que el jugador associés les lletres de l'abecedari que anaven apareixent en la pantalla, i tal com es tocaven les lletres, s'escoltés el seu so corresponent.
A mesura que s'ha anat avanzant en el desenvolupament del joc, s'ha anat modificant la idea base de manera que aquests objectes no fossin sols lletres, sinó que es tractés d'objectes diferents perquè fos més agradable visualment i alhora més entretingut segons els diferents nivells que s'anaven superant.

## 1.4 Planificació del projecte

Podríem dividir la fase de planificació del projecte en tres fases molt diferents. La primera fase dedicada a la iniciació i aprenentatge d'andengine juntament amb l'obtenció de la idea del joc. Una segona fase que englobaria la programació del codi del joc , a més dels testejos, correccions i modificacions d'aquest, i una tercera fase dedicada a les modificacions adequades per obtenir una millor jugabilitat a més de la documentació de tot el codi font i de tot el projecte. Encara que també caldria indicar que la fase segona i tercera han estat molt relacionades donades el tipus de projecte emprès.

### 1.4.1 Fase Inicial

Des de l'inici del projecte fins a la seva conclusió, en cadascuna de les fases que es compon el projecte, no ha estat la intenció de tenir un producte totalment acabat. En la primera fase del projecte, fins i tot abans, s'ha invertit molt de temps en provar les llibreries d'andengine creant mini-projectes amb exemples d'escenes, sprites en moviment, ús de menús, ús de tiledMaps,textures, ús de la física , etc ... i testejos en general.
Potser aquesta fase hagi estat la més crítica ja que de tots els testejos realitzats, s'han obtingut uns resultats que han valgut per iniciar el projecte seguint unes pautes i escollint les llibreries que en un principi han semblat més interessants per a l'elaboració del mateix.
Ha estat al final d'aquesta fase quan ha estat interessant emprendre un projecte d'un joc utilitzant la didàctica d'associació d'objectes.
Com es pot observar en el diagrama de Gantt, aquesta fase abastaria gairebé tot el primer mes de setembre.

### 1.4.2 Fase intermèdia

En aquesta fase del projecte es crea una estructura bàsica del joc, implementant les classe principal BaseActivity i la classe 'Juego' on s'inicia el joc. És en la classe 'Juego' on estava definit el nucli inicial del joc, que en els seus orígens contenia la definició del món físic amb objectes gravitant. Més endavant es crearà la resta de classes tal com s'anaven necessitant, renombrant algunes d'elles a més d'intentar independitzar el codi el màxim possible utilitzant només les classes necessàries.
A partir d'aquest moment és quan s'inicia la fase de programació del joc, on es van afegint opcions noves (programació i disseny) i rectificacions tal com es va desenvolupant el projecte. Al diagrama de Gantt es pot observar detalladament com va evolucionant el projecte a mesura que es van incorporant noves característiques al joc, tal com es succeeixen.

Període comprès entre el els inicis del mes d'octubre fins a finals de novembre. En aquesta fase es desenvolupa el gruix del projecte.

### 1.4.3 Fase Final

La fase final està enfocada a les proves i la jugabilitat, intentant establir les condicions òptimes al joc perquè sigui jugable i sobretot entretingut: la movilitat adequada del personatge,la visibilitat dels objectes (mida en pantalla), o la dificultat en els nivells. També es dedica a la documentació del projecte. Aquesta fase abastaria a grans trets de la meitat del mes de desembre fins a inicis del mes de gener.

| Nombre | Fecha de in... | Fecha de fin |
|---|---|---|
| Projecte VideoJocs Educatius | 7/09/12 | 14/01/13 |
| Proves d'introducció general | 7/09/12 | 23/09/12 |
| Configuració de bitbucket e importació del projecte | 23/09/12 | 2/10/12 |
| Programació i disseny | 3/10/12 | 23/12/12 |
| Creació de llistes enllaçades per guardar objectes | 3/10/12 | 11/10/12 |
| Bodies als objectes | 11/10/12 | 15/10/12 |
| Es crea algorisme de col · lisió. | 11/10/12 | 15/10/12 |
| Proves i correcció algoritme de col · lisió | 15/10/12 | 18/10/12 |
| Creació de la classe Niveles per a controlar el nivells del joc | 18/10/12 | 25/10/12 |
| Programació del control de sortida dels objectes d'associació | 25/10/12 | 1/11/12 |
| Missatge provisional nivells | 25/10/12 | 1/11/12 |
| Proves del moviment del personatge | 1/11/12 | 2/11/12 |
| Millora algorisme de sortida de lletres | 1/11/12 | 6/11/12 |
| Tria de lletra a l'atzar de la bateria. | 2/11/12 | 6/11/12 |
| S'integra el personatge principal del joc al món físic | 6/11/12 | 7/11/12 |
| Modificacions sprite, so. | 6/11/12 | 7/11/12 |
| Es modifica mida escenari a 576x384. | 7/11/12 | 14/11/12 |
| S'afegeix un HUD. | 7/11/12 | 14/11/12 |
| funcions de velocitat per al moviment del personatge | 14/11/12 | 15/11/12 |
| Posició del jugador a la sortida. | 15/11/12 | 16/11/12 |
| Rotació del jugador segons sentit del moviment. | 15/11/12 | 16/11/12 |
| Es completa el disseny de les lletres [A-Z]. | 16/11/12 | 17/11/12 |
| S'afegeix un altre tipus d'obstacle amb més elasticitat | 16/11/12 | 17/11/12 |
| S'afegeix menú inicial: start, help -> back, exit. (Provisional). | 17/11/12 | 18/11/12 |
| Disseny: es creen i afegeixen sprites en els menús. | 17/11/12 | 18/11/12 |
| S'afegeix hud el Botó Pause. (Posició i disseny provisional). | 18/11/12 | 23/11/12 |
| Es modifica l'png del jugador: fem servir la columna_4 per la col · lisió. | 23/11/12 | 24/11/12 |
| Nous elements d'associació a la bateria d'associacions. | 23/11/12 | 24/11/12 |
| S'anul · la el touch i s'afegeix un ScreenControl. | 24/11/12 | 26/11/12 |
| S'aplica moviment usant funcions de força. | 26/11/12 | 27/11/12 |
| Es modifica i ajusta la resolució de la pantalla | 26/11/12 | 27/11/12 |
| Modificacions, correccions i millores: | 28/11/12 | 23/12/12 |
| Configuració del moviment. | 28/11/12 | 8/12/12 |
| Millora algorisme sortida de lletres | 8/12/12 | 11/12/12 |
| Evitem duplicitat d'objectes d'associació a la bateria. | 11/12/12 | 13/12/12 |
| Millora algorisme repetició aleatori | 13/12/12 | 15/12/12 |
| Cada 2 nivells apareixen nous objectes. | 15/12/12 | 16/12/12 |
| Jugabilitat | 17/12/12 | 23/12/12 |
| Obtenció d'objectes cada dos nivells | 17/12/12 | 19/12/12 |
| Canvi versió moviment: velocitat per força | 17/12/12 | 19/12/12 |
| Afegim objectes en array i millorem classe Numbers | 17/12/12 | 21/12/12 |
| Nous dissenys i obstacles: molí amb aspes | 20/12/12 | 22/12/12 |
| Augmentem tamany d'escenes. | 22/12/12 | 23/12/12 |
| Documentació i proves. | 8/12/12 | 14/01/13 |

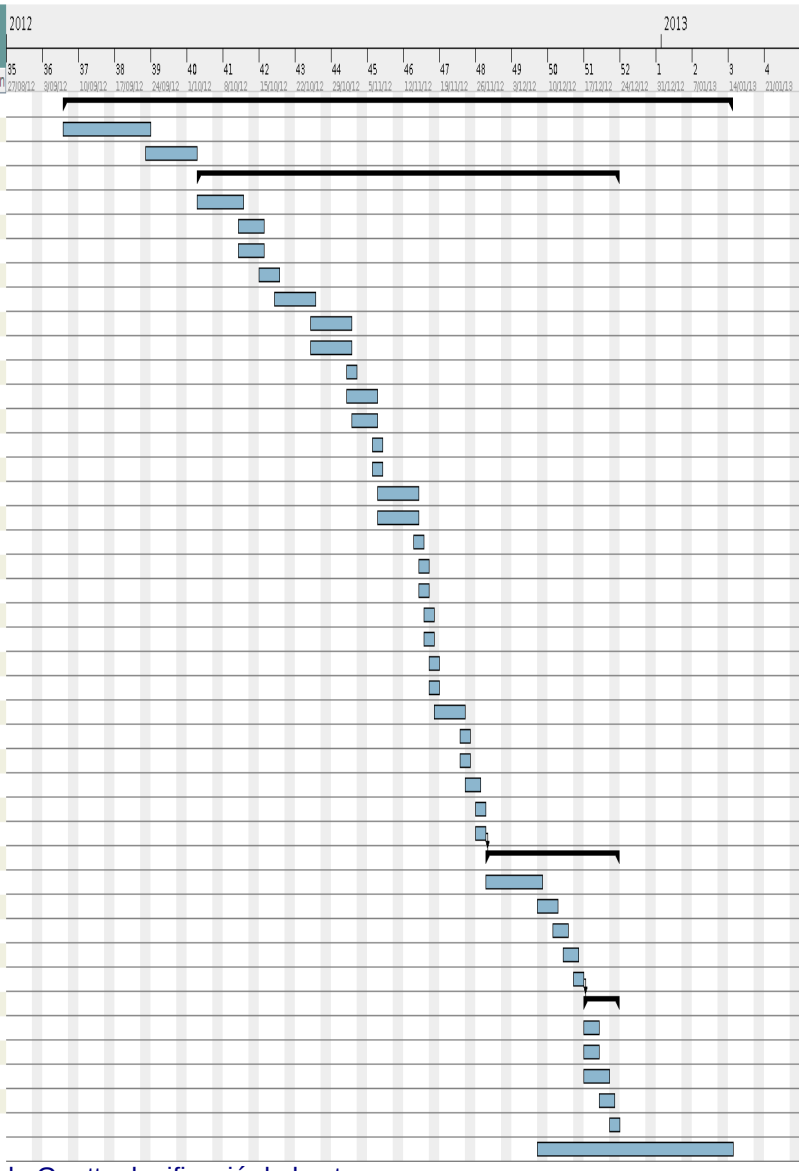



Diagrama de Gantt: planificació de les tasques.

Els dies de la setmana emprats per a la realització del projecte han estat de dilluns a diumenge. Les hores dedicades al projecte han variat depenent del tipus d'implementació realitzada en cada moment passant d'una a quatre hores per dia segons la dificultat (tenint en compte el temps emprat en els dissenys pròpis).
Si tenim en compte que s'han emprat 3 setmanes al setembre, 4 setmanes a l'octubre, 5 setmanes al novembre, 4 setmanes al desembre i dues setmanes al gener, obtenim un total de 18 setmanes. Si la mitjana és de 2.3 hores al dia, podem concloure que la dedicació d'hores al projecte ha estat de 289 hores aproximadament.

### 1.5 Breu sumari de productes obtinguts

Es crea l'estructura principal del joc. Es tracta de codi amb deu classes programades amb andEngine. En arrencar el joc apareix el menú principal, amb la possibilitat d'accedir a una pantalla d'ajuda, sortir del joc o començar a jugar. Es pot jugar a partir del nivell 1, i el joc només està preparat per a cinc tipus d'objectes d'associació i deu nivells, i a més són de caràcter provisional. No es tracta doncs d'un producte finalitzat.

# 2. Disseny

El joc té com a principal objectiu el d'associar els objectes que apareixen a la part esquerra de la pantalla (en una pila vertical) amb els objectes que van apareixent (caient) des de la part superior de la mateixa. Abans de passar a descriure el mecanisme del joc és interessant conèixer alguns dels conceptes bàsics per a interpretar millor el seu funcionament:

- Escena: L'escena és l'entorn del joc que conté tots els objectes que hi ha a la pantalla. L'escena es compon d'una mida on interactuen tots els objectes a més del disseny de fons. Aquest projecte només contempla un disseny d'escena, però està preparat per a poder crear i acoblar totes les escenes que es desitgin de forma dinàmica.
- Nivells: Podem definir un nivell com la dificultat amb què es trobarà el jugador tal com vagi avançant en el transcurs del joc. Es començarà pel primer nivell. Un nivell és un món format per obstacles els quals interferiran entre el jugador i els objectes d'associació que van caient per la pantalla.
- Obstacles: Formen part d'un nivell i per tant s'assignen a cada un dels nivells. Un nivell conté un nombre d'obstacles definits per endavant i poden ser de tipus molt diferents. El joc està preparat

per crear tants tipus d'obstacles com es desitgi, així com de situar-los en zones estratègiques de la pantalla, segons augmenti la dificultat del nivell.
- El món físic: És una representació física del joc. En aquest joc, aquesta representació simula una força de gravetat del planeta Plutó. El món físic s'associa a una escena.
- Perseguint a la càmera (chasing camera): Atès que la mida de l'escena definida és molt més gran que la mida de la pantalla, el personatge es podrà moure lliurement per aquesta escena, fins a arribar als límits de la mateixa. Hi ha una càmera que segueix constantment el moviment del personatge.

El mecanisme o procés que es segueix és el següent:

En iniciar el joc ens apareix un menú a la pantalla amb les següents opcions:
- Start: es comença a jugar.
- Help: es mostra una pantalla d'ajuda explicant el funcionament del joc. Un cop consultada l'ajuda es torna a la pàgina principal seleccionant l'opció: back.
- Quit: se surt del joc.

Menú inicial del joc

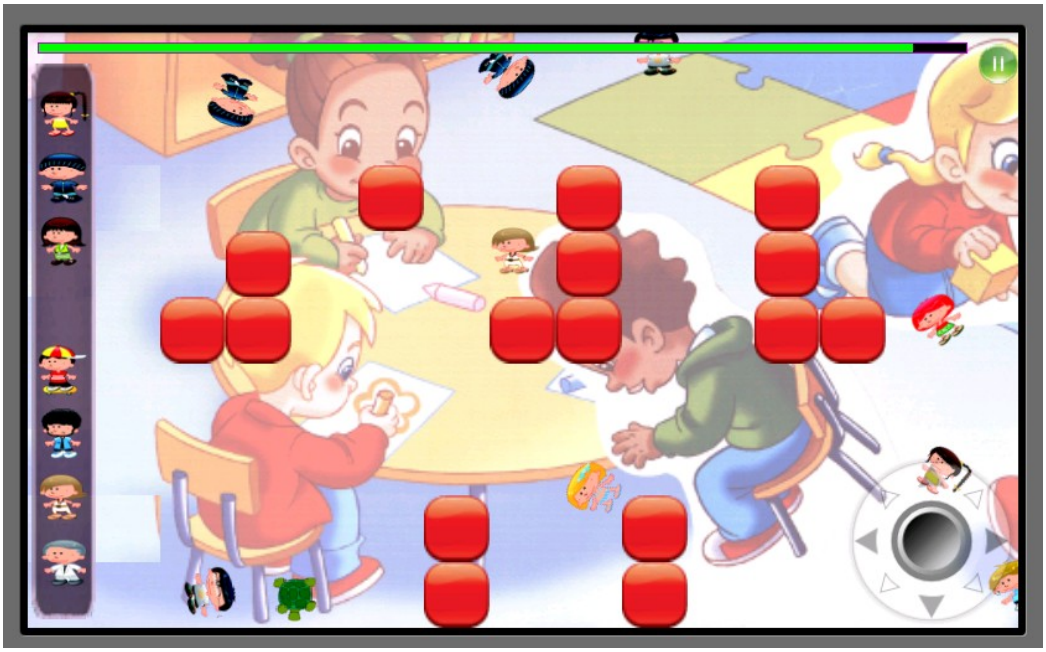
Només prémer l'opció Start del menú comença la partida on es poden destacar les següents funcionalitats:

- El jugador mourà el "personatge" de manera tàctil per tota la pantalla. Per això s'ha implementat un ScreeNControl.
- El joc començarà pel nivell inicial. Nivell 1.
- Apareixerà una bateria d'objectes apilats en vertical a la part esquerra de la pantalla. Es tracta dels objectes que s'han d'associar amb els objectes que apareixeran (cauran) des de la part superior de la pantalla.
- Un cop s'han carregat tots els objectes a la part esquerra, comencen a caure objectes des de la part superior de la

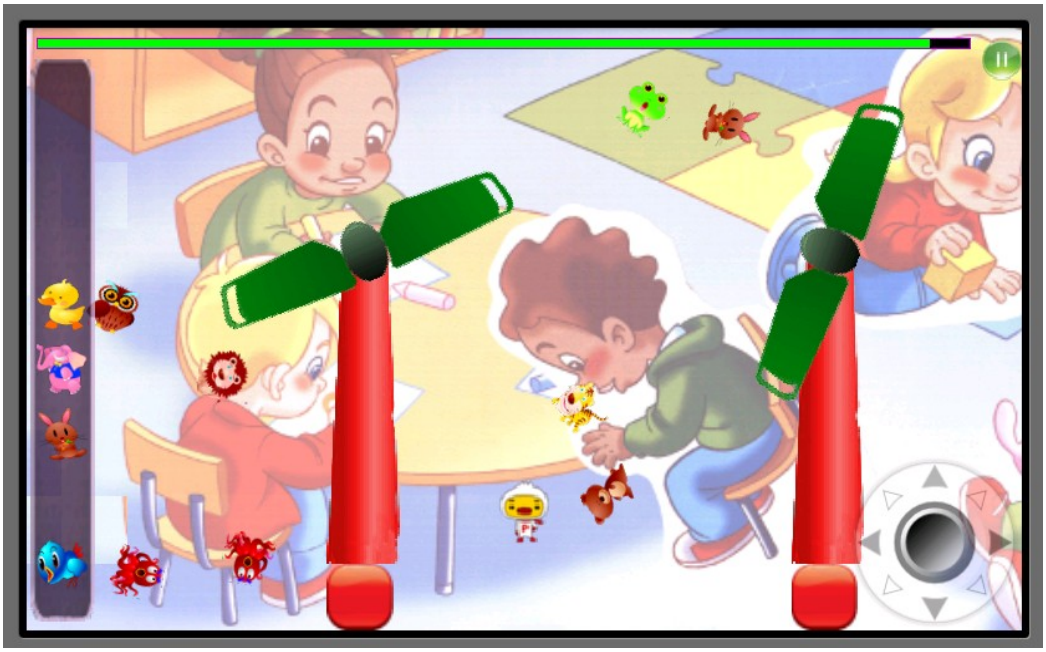
pantalla.El jugador haurà de tocar amb el seu personatge només els objectes mostrats a la bateria d'objectes. Els objectes que cauen de la part superior de la pantalla interactuen entre ells, és a dir, col · lideixen, canviant la seva trajectòria depenent de l'obstacle que han col · lisionat, i graviten en un entorn que "depèn" del món físic que s'ha definit.

- A cada nivell, com ja s'ha descrit, existeixen els obstacles que també interactuen amb els objectes que cauen, xocant i modificant la seva trajectòria en cada moment. Es produeix doncs, un món aleatori, impredictible i diferent en cada partida.
- L'objectiu del joc tracta que el jugador toqui únicament els objectes mostrats a la bateria d'objectes. Quan s'associï un objecte correcte, aquest objecte desapareixerà de la bateria d'objectes, i de la pantalla principal. En cas contrari es decrementarà el nivell d'energia.
- Quan el jugador associï tots els objectes de la bateria es passarà al següent nivell. Els nivells es van incrementant secuencialment.
- La partida acaba si el jugador se li acaba l'energia.
- El projecte està pensat perquè apareguin objectes diferents d'associació cada X nivells de joc.
- El joc ve acompanyat de so que permet apreciar millor l'estat de la partida.
- En començar a jugar disposem d'un botó tàctil de pausa que congela el joc o tornar a reprendre.
- El joc acabaría amb éxit quan s'han superat tots el nivells.

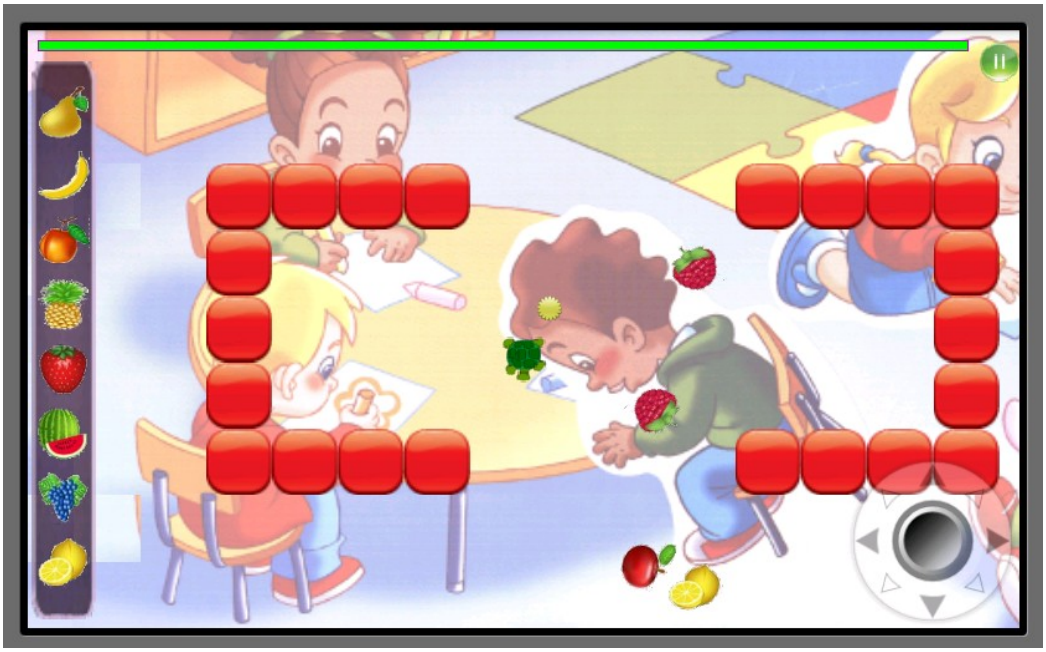
En les següents línies es mostren unes captures de pantalla del joc en funcionament:



Imatge corresponent al nivell 1 de joc. Abans de varies modificacions.



Imatge corresponent al nivell 4 de joc.



Imatge corresponent a un altre nivell de joc.

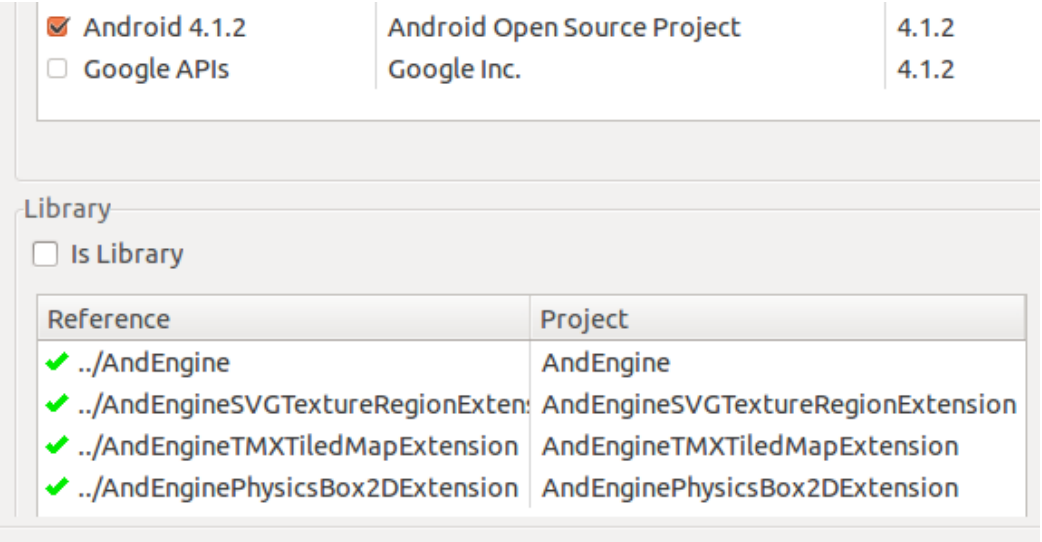
# 3. Tecnologies utilitzades

Eines de treball utilitzades per al desenvolupament del projecte:

- Portàtil Dell Inspiron 9400. Processador Core 2 Duo T7200 (2.0Ghz, 667Mhz FSB, 4MB L2 cache). Pantalla panoràmica 17". Memòria Ram de 2Gb.
- Targeta gràfica de 256 MB Nvidia GeForce Go 7900 GS PCI Express.
- Sistema operatiu Ubuntu 12.04.1 LTS.
- Eclipse Java EE IDE for Web Developers.Versió: Juno Release Build id: 20120614-1722
- eUML2 Studio versió avaluació.
- Tiled Map Editor per a Ubuntu Versió Copyright 2008-2011.
- Programa de manipulació d'imatges Gimp 2.6.12 per Ubuntu.
- Control de versions en bitbucket.org

Desenvolupament del projecte. Llibreries i programari de desenvolupament:

- Llibreries d'Android 4.1.2 Open Source Project 1/4 Api Level 16
- ARM EABI v7a System Image. (Emulador AVD)
- La Llibreria **nicolasgramlich / AndEngine** amb la branca GLES2 (informació de GLES2) que requereix Android versió >= 2.2. Aquesta versió és compatible actualment amb el 93% dels Dispositius mòbils a tot el món - d'acord a les Informacions Oficials d'abril de 2012.
- Les llibreries addicionals:
  - AndEngineSVGTextureRegionExtension
  - AndEngineTMXTiledMapExtension
  - AndEnginePhysicsBox2DExtension

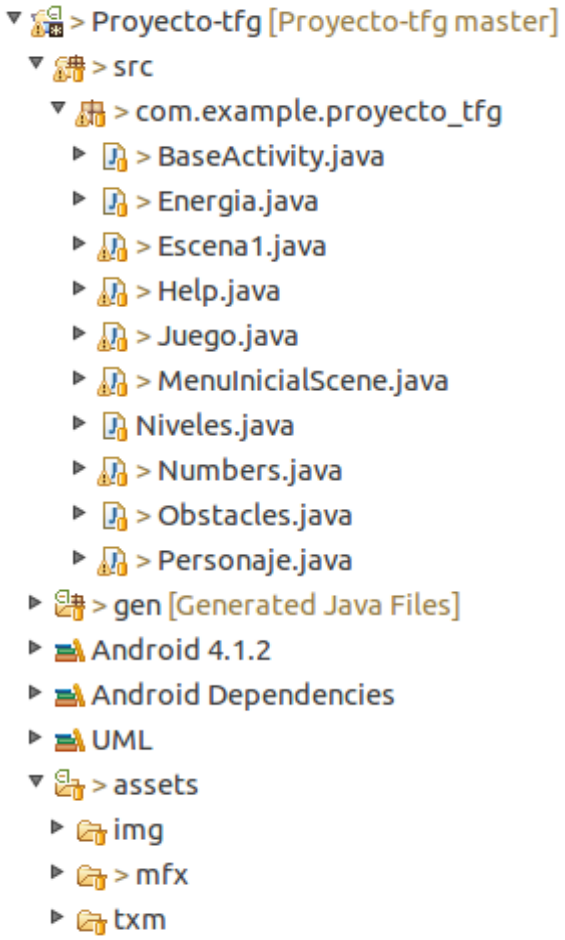



Llibreries del projecte en Eclipse.

# 4. Programació de l'aplicació

## 4.1 Estructura bàsica del projecte

El desenvolupament s'ha dut a terme amb les llibreries d'AndEngine GLES2 i utilitzant l'IDE d'Eclipse Juno. L'estructura del projecte (Projecte-TFG) es basa en una estructura similar als projectes d'Android. En l'estructura del projecte cal destacar el directori **src/** que conté el paquet amb les classes font en .java i el directori **assets/** que conté els recursos gràfics i de so.




Estructura bàsica del projecte.

**assets/**

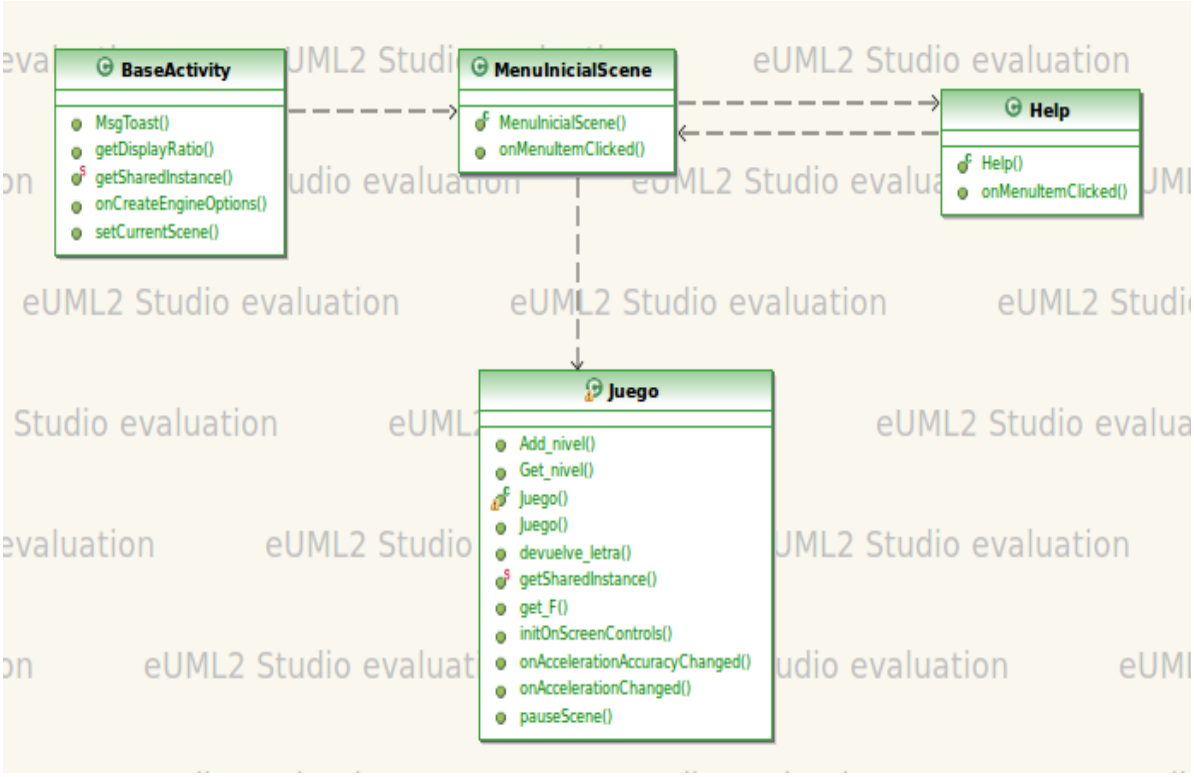
**img/** Conté totes les imatges del joc, dels menús, sprites dels objectes d'associació, el jugador, botons ... en format .png

**mfx/** Conté tots els arxius de so del joc en format .mp3 o .wav

**txm/** Conté un arxiu en format .tmx comprimit en gzip i creat mitjançant el programa Tiled, que dissenya un mapa del joc a partir de textures o 'tileds'.

## 4.2 Disseny de classes en el cicle de vida del joc

El següent diagrama UML ens mostra el cicle de vida seguit pel joc a través de la relació de classes.




Cicle de vida de les activitats.

La clase BaseActivty

La classe *BaseActivity*, a part de generar el motor del joc i de ser la classe encarregada de gestionar tots els recursos, com es veurà més endavant, és la classe d'inici per defecte que crida mitjançant el mètode *onCreateScene()* a l'activitat o classe *MenuInicialScene().*

La clase MenuInicialScene

Aquesta classe de tipus *MenuScene* és l'encarregada de gestionar el menú 'Start', 'Help' i 'Quit', a més dels possibles opcions noves del menú que es optin per habilitar.
Cadascuna d'aquestes opcions es creen mitjançant sprites que apareixen a la pantalla i que s'activen de manera tàctil pel jugador.La opció Start crida a la classe 'Juego', i s'inicia el joc en el primer nivell.
L'opció 'Help' crida a la classe Help que està preparada per mostrar un text d'ajuda sobre el joc. L'opció 'Quit' abandona el joc.

La clase Help

La classe Help, que és una classe de tipus menu escena, està pensada per mostrar tot tipus d'ajuda al jugador. A més conté un botó tàctil de retorn cap a la classe MenuInicialScene ().

La clase Juego

Es pot dir que la classe 'Juego', és la classe nucli del joc. Es tracta d'una classe que s'estén de la classe Scene a Andengine. És l'escena principal del joc, on es basa tota la seva activitat. Aquesta classe Scene és el contenidor arrel de tots els objectes del joc (recursos) que es poden treure a la pantalla. Una escena pot tenir una quantitat específica de capes, objectes i que al seu torn pot contenir una quantitat (fixa o dinàmica) d'entitats. Les entitats són objectes com poden ser tipus de sprites, rectangles, text o línies, etc, que posseeixen una sèrie de característiques específiques que les defineixen mitjançant una sèrie de propietats i mètodes.
A la classe 'Juego' es defineixen les propietats i mètodes bàsics del joc, així com tots els elements principals del joc, com ara objectes instanciats d'altres classes, la definició o porció d'imatge que es veurà en el joc (càmera), la inclusió del "món físic" a l'entorn del joc, el control d'actualitzacions, i en definitiva els mètodes de control dels objectes definits en aquest.

Els següents diagrames UML mostren la relació, els seus associacions, dependències, propietats i mètodes de les classes utilitzades en el projecte.

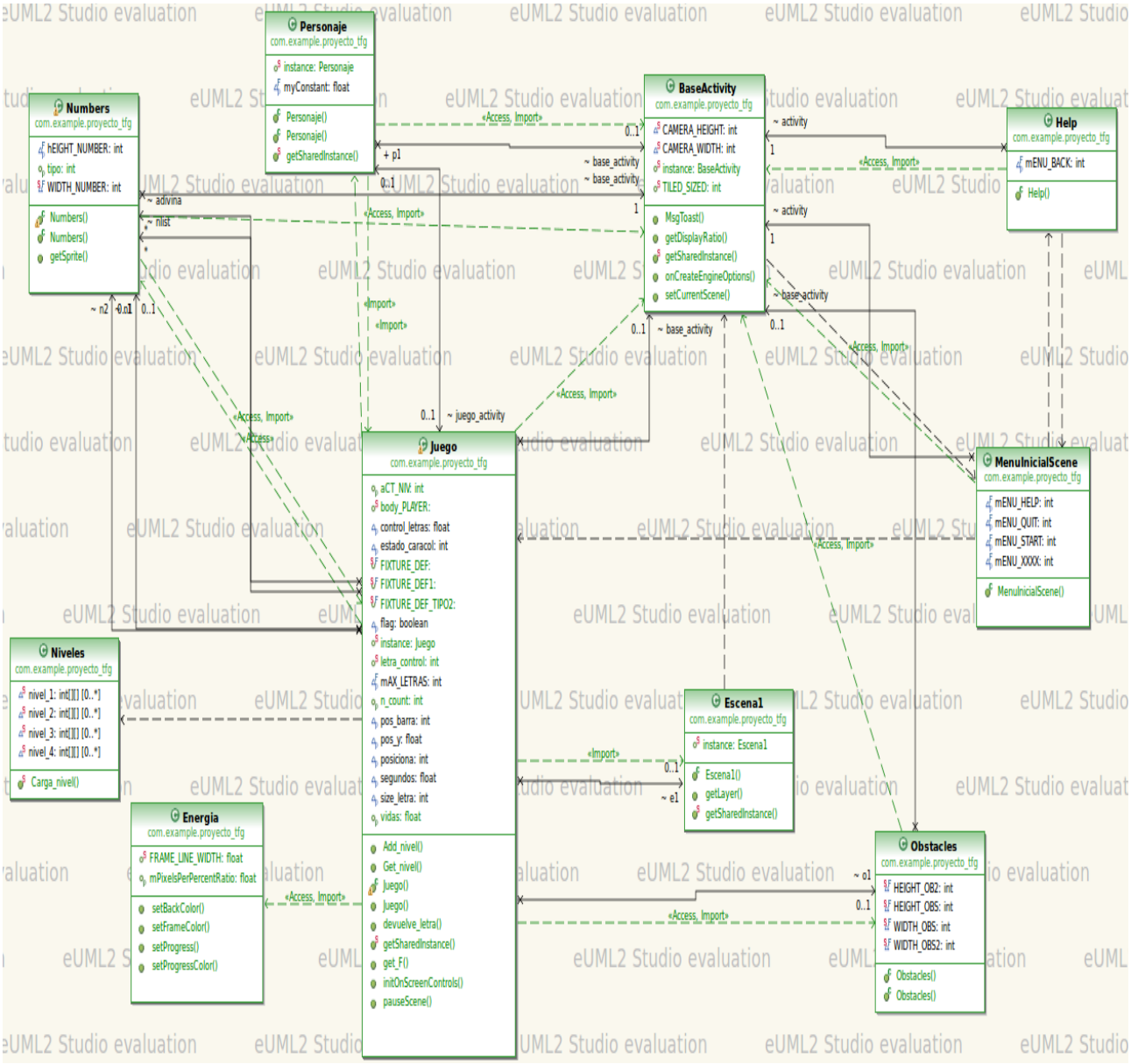
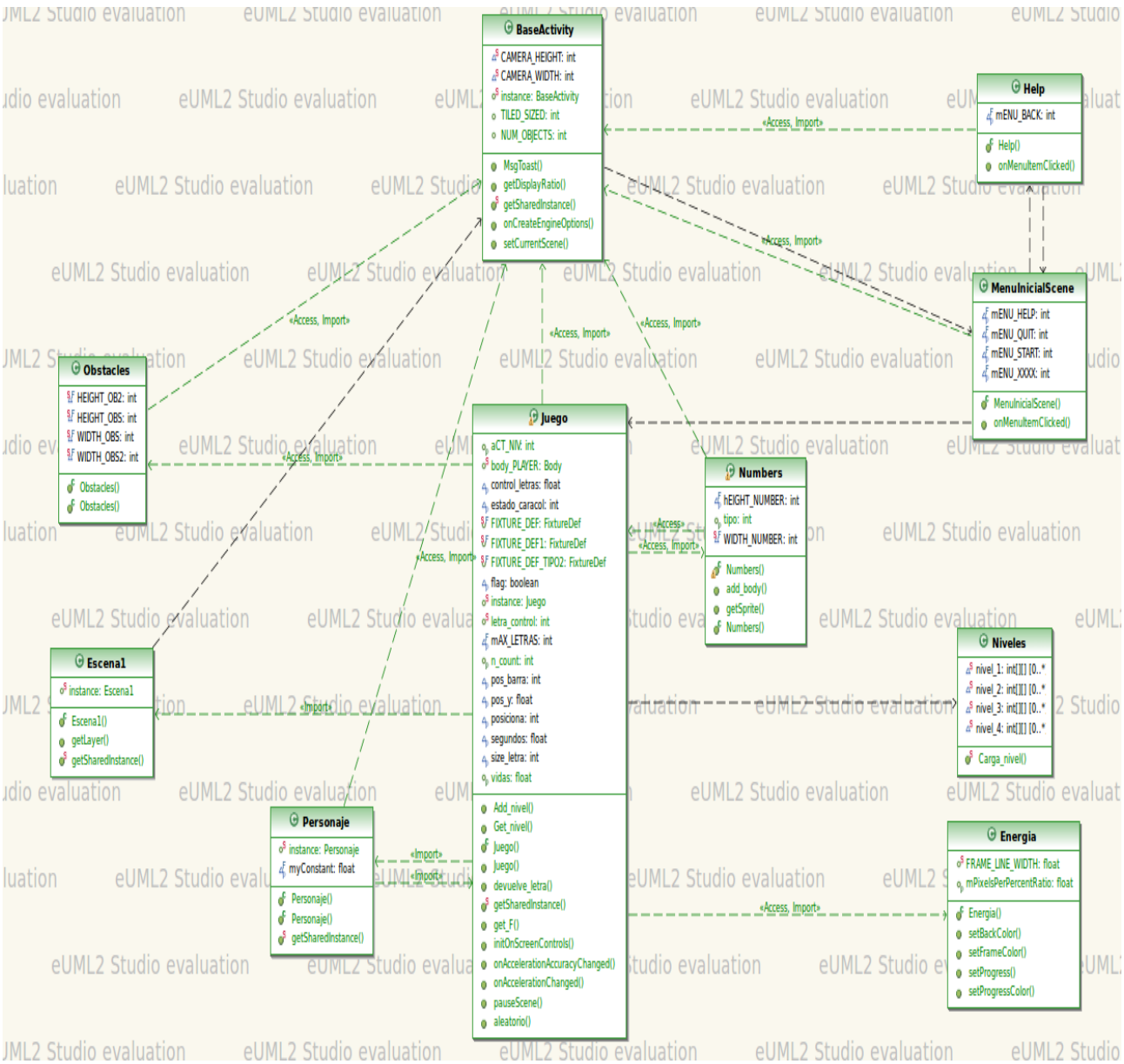



Diagrama complet de vistes.




Vista de les classes per dependències i relacions.

## 4.3 Aspectes més rellevants:

1. El món físic:

   El projecte utilitza la llibreria AndEnginePhysicsBox2DExtension. Box2D conté les llibreries necessàries per a la simulació física de totes les entitats que es defineixin en el joc. El motor de física és un sistema que ens permet crear efectes d'animació sobre el joc on els objectes s'exposen a les forces físiques.
   Totes les entitats del joc estan influenciades per la gravetat. Concretament la gravetat que es simula en el joc és la del planeta Plutó.

```
MundoFisico = new
PhysicsWorld(newVector2(0,SensorManager.GRAVITY_PLUTO), false);
```

   Les entitats (sprites) que es veuran afectades pel món físic són: el player (al qual s'ha anulat la gravetat) , els obstacles, els objectes d'associació i les regions que delimiten el joc. A cadascuna d'aquestes entitats o sprites se'ls associa un "body" o contenidor.

```
body_PLAYER=PhysicsFactory.createCircleBody
      (MundoFisico,Player,BodyType.DynamicBody,FIXTURE_DEF1);
body_PLAYER.setLinearDamping(0.01f);
MundoFisico.registerPhysicsConnector(
      new PhysicsConnector(Player,body_PLAYER, true, true));
```

   A cada un dels contenidors creats s'associa unes característiques especials anomenades "Fixtures", que assignen efectes de densitat, elasticitat i fricció.

```
public static final FixtureDef FIXTURE_DEF =
      PhysicsFactory.createFixtureDef(1, 1f, 0.5f);
```

   Cal comentar que durant la realització d'aquest projecte s'han mantingut dues possibles versions del joc, una versió on el jugador quedava afectat per la gravetat (utilitzant funcions de moviment de força) i una altra versió sense que l'afectés la gravetat (utilitzant funcions de velocitat per al moviment).
   Donada la dificultat en el moviment que comporta la integració del jugador dins un món de gravetat, i atès que no s'ha disposat físicament d'un dispositiu real de proves, per mesurar totes les possibilitats, s'ha

optat per triar la versió del jugador sense afectar-li la gravetat utitzant moviment amb les funcions de velocitat.

2. Control de col · lisions:

Per controlar les col · lisions s'han creat dues llistes enllaçades, una llista per als objectes d'associació de la bateria i una altra per als objectes d'associació que surten per pantalla. La llista dels objectes d'associació de sortida per pantalla es va incrementant a mesura que es van generant els objectes. A cadascuna d'aquestes llistes se'ls assigna les propietats de l'objecte Numbers, que conté el body de l'objecte i el tipus, que correspon a un identificador únic d'objecte. Si el jugador col · lisiona amb un objecte d'associació de sortida, es recorren les llistes per identificar el tipus d'objecte que ha topat i si s'ha produït o no l'encert. Aquesta comprovació es realitza en el registre de col · lisions que realitza el testeig mitjançant el gestor d'esdeveniments IUpdateHandler ().

```
Iterator<Numbers> i = Nlist.listIterator();
while (i.hasNext())
{
      Numbers b = i.next();
      if (Player.collidesWith(b.getSprite())) //Col.lisió
      {
      // Recorrem llista de lletres a endevinar per veure si
      //coincideix amb el que hem topat
      Iterator<Numbers> x1 = Adivina.listIterator();
}
```

Exemple de codi detecció de col.lisions.

3. Objectes d'associació de la bateria.

La bateria d'objectes d'associació són aquells objectes on el jugador ha d'associar i tocar mitjançant el "Player" amb els objectes de sortida per la pantalla. Aquests apareixen en una bateria vertical d'objectes a la part esquerra de la pantalla integrats dins d'un Hud. El joc s'ha configurat per mostrar cinc objectes diferents en la bateria i es pot modificar. La creació d'aquests objectes es realitza mitjançant la creació d'un objecte de la classe Numbers. La classe Numbers conté el mètode específic per crear els objectes de la bateria. Aquests varien o són diferents en funció del nivell.

```
n2 = new Numbers(col_x,posiciona,get_F(),v2[n_count]);
```

Exemple de creació d'objectes a la bateria.

4. Objectes d'associació de sortida per pantalla.

Una vegada que s'han mostrat els cinc objectes d'associació a la bateria, comencen a caure des de la part superior de la pantalla, els objectes que han de ser associats. El temps de sortida entre cada un dels objectes és parametritzable i es pot ajustar segons el nivell de dificultat.

L'identificador de l'objecte de sortida s'obté de forma aleatòria i és per tant que s'ha de controlar que almenys de tant en tant, aparegui un objecte en pantalla que es correspongui amb algun objecte de la bateria d'objectes per disminuir la dificultat i perquè el joc sigui més entretingut.

```
//Crida al constructor. Surtida d'objectes en funció del nivell.
n1 = new Numbers(get_F()); //Obté un id.
Body body_letras = PhysicsFactory.createCircleBody(MundoFisico,
n1.getSprite(),BodyType.DynamicBody, FIXTURE_DEF);
// Assignem el body a l'objecte
n1.add_body(body_letras);
```

Exemple de control de surtida d'objectes. Crida al contructor Number.

Com es pot observar en l'exemple, a la classe Numbers tenim un constructor que és l'encarregat d'obtenir l'identificador de l'objecte, en funció d'un paràmetre que ve donat per un algoritme mitjançant un TimerHandler, que controla diverses funcionalitats:

- El temps en què s'inicia el control d'identificador d'objectes per comprovar si hi ha en aquest moment un objecte en pantalla que està a la bateria d'objectes. Aquest temps és parametritzable. Per defecte, en el joc aquest control es comença a testejar als 40 segons d'inici del joc.
- Una funció que recorre tota la llista de la bateria d'objectes per esbrinar si es troba o no el mateix identificador d'objecte en la llista d'objectes de sortida. Si el troba és perquè hi ha un objecte a la pantalla que també es troba a la bateria i no es fa res, si no el troba, llavors s'obliga a que l'identificador d'objecte que ha de sortir sigui un que es trobi en la bateria de objectes.
- Després del primer control de l'algorisme, el temps en què es torna a realitzar el control dels objectes varia en funció de dos paràmetres que indiquen l'interval aleatori de temps en què es tornarà a iniciar el control.

```
control_letras = 40f;
registerUpdateHandler(l_out_1 = new TimerHandler(control_letras, true,
new ITimerCallback() {
        @Override
        public void onTimePassed(final TimerHandler pTimerHandler) {
```

```
                letra_control = devuelve_letra(); //Retorna id o -1
                //Temps en segons del proper control
                int MaxValue=60;
                int MinValue=40;
                float seg = (int)((MaxValue - MinValue + 1) * Math.random()) +
                                                                    MinValue;
                l_out_1.setTimerSeconds(seg);
            }
        }));
```

Exemple de control de surtida d'objectes.

5. Càrrega d'obstacles en l'escena

El joc està preparat per crear o definir tots els obstacles que es vulguin amb les seves pròpies característiques.
La classe obstacles té implementat un constructor en el qual estan definits tots els tipus diferents d'obstacle per al joc.
A la classe Nivells es defineix per a cada nivell del joc una matriu de 18x12, que correspon a les dimensions de l'escena. En principi, totes les escenes van a tenir el mateix tamany. En aquesta matriu es pot definir un tipus de obstacle diferent, cada un d'ells identificat per un valor numèric.
Per tant en la classe joc es carrega l'escena amb els seus corresponents obstacles depenent del nivell en el qual ens trobem i que hi ha definit a la classe Nivells.

```
for(int x=0;x<18;x++)
        for(int y=0;y<12;y++){
          switch(Niveles.Carga_nivel(Get_nivel())[y][x]){
                 case 1:
                     o1 = new
Obstacles(x*base_activity.TILED_SIZED,y*base_activity.TILED_SIZED,1);
final Body body_Obs = PhysicsFactory.createBoxBody(MundoFisico,o1.Obs,
BodyType.StaticBody, FIXTURE_DEF);
                     attachChild(o1.Obs);
                     MundoFisico.registerPhysicsConnector(new
PhysicsConnector(o1.Obs,body_Obs, true, true));
                 break;
                 case 2:
                     o1 = new
Obstacles(x*base_activity.TILED_SIZED,y*base_activity.TILED_SIZED,2);
                     final Body body_Obs1 =
PhysicsFactory.createCircleBody(MundoFisico,
o1.Obs,BodyType.StaticBody, FIXTURE_DEF_TIPO2);
                     attachChild(o1.Obs);
                     MundoFisico.registerPhysicsConnector(new
PhysicsConnector(o1.Obs,body_Obs1, true, true));
                 break;
            }
```

```
        }
```

Exemple de càrrega d'obstacles en la classe Juego.

```
static int[][] nivel_2 = {{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
              {0,0,0,0,2,0,0,0,0,2,0,0,0,0,2,0,0,0},
              {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
              {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
              {0,0,0,1,1,1,0,0,1,1,1,0,0,1,1,1,0,0},
              {0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0},
              {0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0},
              {0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0},
              {0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0},
              {0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0}};
```

Exemple de configuració del nivell 2 del joc en la classe Niveles.

```
 public static int[][] Carga_nivel(int level){

        int[][] resultado = new int[18][12];

        switch (level){
        case 1:
            resultado = nivel_1;
            break;
        case 2:
            resultado = nivel_2;
            break;
```

Exemple del mètode que retorna el nivell actual del joc en la classe Niveles.

## 6. Moviment del personatge (Player)

A l'inici del projecte el personatge del joc (player) es movia de manera tàctil posicionant el dit sobre el sprite, dirigint el moviment segons la direcció tàctil del dit “touch”. Després de nombroses proves es va comprovar que la posició del dit sobre el dispositiu físic per moure el personatge, dificultava la visió sobre el joc. Així que es va modificar el moviment tàctil sobre el personatge per un ScreeNControl a la pantalla per a generar el moviment. Per tant, el moviment es produeix dins dels esdeveniments “`onControlChange`” d'un ScreenControl.
També és interessant comentar que al principi el jugador no l'afectava la gravetat, però es va creure més convenient aplicar-la per donar més realisme al joc i que estigués influenciat per l'entorn de gravetat. Donada la dificultat de parametrització, s'ha optat per no aplicar la gravetat i utilitzar moviment lineal, encara que en les següents línies es comenta el moviment de la força perquè resulta interessant.

Per al moviment del personatge s'aplicava la funció de força al cos associat “body” perquè interactués amb les forces de la gravetat i amb tots els seus paràmetres relacionats.
També s'aplicava una transformació sobre el cos perquè girés en sentit de la direcció del moviment aplicat.
Com el sprite que defineix el jugador està format per "tiles", i és de tipus animat, s'utilitzen estats diferents per representar les diferents situacions en què es pot trobar el jugador: en moviment, parat o caient influenciat per la gravetat. Cada un d'aquests estats defineixen un disseny concret i un so diferent, que també s'utilitza en el moviment lineal.

```
if (old_estado == estado_player)
    return;

 switch(estado_player){
   case 0://Es mou
       Player.animate(new long[]{200, 200,200,200,200}, 0, 4, true);
       break;
   case 1://Quiet.
       Player.animate(new long[]{200, 200}, 5, 6, true);
       break;
```

Exemple d'estats en el onControlChange per al moviment.

Tres dels paràmetres importants que generen el moviment del jugador són:

- la velocitat: la velocitat del personatge depèn de tres paràmetres: una constant de desplaçament, la posició del personatge i la massa del cos (en cas de força). Gràcies a això s'obté el vector de la velocitat que serà el que s'utilitzarà per calcular el moviment usant la força. Cal dir que cada cos té una massa (escalar) i que normalment les propietats de massa d'un cos s'estableixen automàticament quan s'assignen les propietats del cos (fixtures) encara que es pot ajustar aquest valor.
- la força aplicada: s'aplica utilizant la funció applyForce sobre el cos (body) del personatge en el seu centre de masses.Aquesta força actua gradualment amb el temps per canviar la velocitat del personatge.
- l'amortiment aplicada: l'amortiment s'utilitza per reduir la velocitat del player dins del mon físic.Cal diferenciar l'amortiment a la fricció ja que la fricció es produeix només amb contacte dels cossos.

```
final float vel = 12.0f*body_PLAYER.getMass();
final Vector2 velocity = Vector2Pool.obtain(pValueX *vel, pValueY
                                                          *vel);
```

```
      body_PLAYER.applyForce(velocity,body_PLAYER.getWorldCenter());
      Vector2Pool.recycle(velocity);
```

Exemple d'aplicació de moviment utilizant funció de força.

En el joc, s'ha implementat i per tant s'usa la velocitat lineal.

```
      final Vector2 velocity = Vector2Pool.obtain(pValueX*3.0f,

      pValueY*3.0f);
      Juego.body_PLAYER.setLinearVelocity(velocity);
      final float rotationInRad = (float)(Math.atan2(-pValueX,
                                                     pValueY));
body_PLAYER.setTransform(body_PLAYER.getWorldCenter(),rotationInRad);
      Player.setRotation(MathUtils.radToDeg(rotationInRad));
      Vector2Pool.recycle(velocity);
```

Exemple d'aplicació de moviment utilizant la funció de velocitat lineal.

## 4.4 Disseny de sprites. Textures. So.

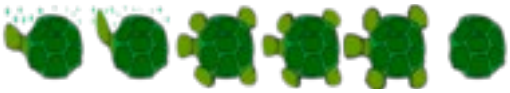
El disseny del jugador

Per a la creació del personatge que es mourà per la pantalla es crea un BitmapTextureAtlas que serà la porció total de la imatge a carregar a la memòria. El BitmapTextureAtlas defineix tot la mida de la imatge. En el joc el personatge té definit un espai de BitmapTextureAtlas de 264x44 píxels. El jugador tindrà un tamany de 44x44 pixels.

Un cop reservat l'espai que ocuparà la imatge, es defineix una regió (TiledTextureRegion) o porció d'imatge que defineix com estarà estructurada aquesta imatge. Això s'aconsegueix gràcies al mètode "*createTiledFromAsset*" que serà el que definirà l'estructura de la imatge. Entre les funcions més importants d'aquest mètode, destaca la de carregar la imatge de tipus png i dividir o definir en regions de files i columnes el BitmapTextureAtlas.
En el joc, la imatge consta d'una fila per vuit columnes, cadascuna destinada a una opció concreta del joc. Aquestes definicions s'assignen a la classe BaseActivity.

```
      mBitmapTextureAtlas = new
      BitmapTextureAtlas(this.getTextureManager(),
            264, 44, TextureOptions.DEFAULT);
      mPlayerTextureRegion =
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(mBitmapTex
                              tureAtlas, this, "pato.png", 0, 0, 8, 1);
                              mBitmapTextureAtlas.load();
```

Exemple de definició del jugador o Player.



Exemple d'una de les imatges del jugador abans del disseny definitu.

Exemple de la imatge del jugador definitiva.

Les columnes de la imatge del jugador corresponen:

- Columna 1 a 5: moviment.
- Columna 6-7: jugador parat.
- Columna 8: a la col · lisió.

Cal comentar que aquest disseny pot ser modificat ràpidament si es desitja per exemple implementar la gravetat o augmentar les columnes de la imatge per inserir més frames per definir el moviment del personatge.

La classe “Personaje” és l'encarregada de crear el sprite associat al personaje.El sprite que defineix el jugador és de tipus AnimatedSprite, que proporciona una sèrie de mètodes apropiats per generar l'animació.Com es pot observar en el següent exemple, a l'AnimatedSprite s'associa la Textura o regió de la textura que s'ha creat amb anterioritat.

```
sprite = new AnimatedSprite(centerX, centerY,
      base_activity.mPlayerTextureRegion,
      base_activity.getVertexBufferObjectManager());
```

Exemple de creació del sprite en la classe “Personaje”.

A la classe 'Juego', depenent de l'estat en què es troba el jugador, s'aplica l'animació al personatge mitjançant el mètode "*animate*”,on rep la durada de cada frame, i la posició o columna inicial a final que indicarà on es generarà el moviment.

```
switch(estado_player){
   case 0://Se mueve
    Player.animate(new long[]{200, 200,200,200,200}, 0, 4, true);
    break;
   case 1://Parado
    Player.animate(new long[]{200, 200}, 5, 6, true);
    break;
 }
```

Exemple d'animació del sprite.

El disseny dels objectes s'associació

Igual que en les definicions del personatge, els objectes d'associació també es defineixen a la classe BaseActivity. Els BitmapTextureAtlas es creen de la mateixa manera. Per a la creació de les regions es crea un vector OB_AS[] on es guarda cadascuna d'aquestes definicions que millorarà el codi del joc.

```
      public final TiledTextureRegion OB_AS[] = new
            TiledTextureRegion[4];
this.OB_AS[1]=BitmapTextureAtlasTextureRegionFactory.createTiledFromAs
set(this.m_AS_1_Texture, this,"AS_1.png", 0, 0, 7, 2);
```

Exemple de creació de regions en els objectes d'associació.

Els objectes d'associació estan formats per 14 imatges dividides per 7 imatges en dues columnes (matriu de 2x7). La mida és de 308x88 píxels, on cada imatge individual té una mida de 44x44 píxels. Cadacuna de les imatges s'accedirà a través d'un índex de 0 a 13.

Exemple de la matriu d'objectes d'associació.

Per a la generació i visualització dels objectes d'associació en la batería s'utilitza el constructor public Numbers (int x, int y, int flotants, int id) on:

- *x e y* indiquen la possició.
- *flotantes* indica l'índex del vector OB_AS[] per obtenir els objectes d'associació diferents a cada nivell.
- *id* indica l'identificador de cada objecte ia més és l'encarregat de seleccionar una imatge de la matriu mitjançant el mètode *setCurrentTileIndex*.

```
      mNum = new TiledSprite(x, y, WIDTH_NUMBER, HEIGHT_NUMBER,
                                   base_activity.OB_AS[flotantes],
                        base_activity.getVertexBufferObjectManager());
      tipo = id;
      mNum.setCurrentTileIndex(id);
```

Exemple de selecció d'una imatge amb *setCurrentTileIndex.*

Per a la generació dels objectes d'associació que surten per la pantalla s'utilitza la mateixa matriu vista amb anterioritat, el constructor public Numbers (int flotants) i s'utilitza la mateixa tècnica que s'ha descrit.

El disseny dels obstacles

Els obstacles també es defineixen a la classe BaseActivity. El joc té definits tres tipus d'obstacle diferents i està preparat per crear tants obstacles com es desitgi. El tamany per defecte és de 32x32 pixels.

```
//Obstacles-tipus-1
 this.mObstaclesTexture = new
     BitmapTextureAtlas(this.getTextureManager(), 32, 32,
                                        TextureOptions.BILINEAR);
 this.mObstaclesTextureRegion =
BitmapTextureAtlasTextureRegionFactory.createTiledFromAsset(this.mObst
                  aclesTexture, this, "bloque-1.png", 0, 0, 1, 1);
 this.mObstaclesTexture.load();
```

Exemple de definició dels obstacles.

Objecte de tipus obstacle_1

Objecte de tipus obstacle_2

L'obstacle de tipus 3: molí amb aspes, utilitzant RevoluteJoint:

La llibreria física, té un nombre 'd'articulacions' (joints) que poden ser utilitzats per connectar dos cossos junts. Aquestes articulacions es poden utilitzar per simular la interacció entre els objectes per formar frontisses, pistons, cordes, politges, rodes, vehicles, cadenes, etc ...
Els obstacles de tipus 3 es defineixen com la unió de dos sprites, i utilitzen aquesta tècnica de "revolute joint" d'andengine. El sprite 1 es refereix a l'aspa giratòria del molí, i el sprite 2 es refereix a la base del mateix. Cal dir que si s'utilitza el moviment sobre l'obstacle, els dos cossos interaccionarien junts.

Sprite_1 [T1] Aspa giratòria del molí.

Sprite_2 [T2] Base del molí.

```
      case 3:
      //Ajustar
      T1 = new Sprite(pX-57,pY-5,150,30,
      base_activity.aspa1_textureRegion,
      base_activity.getVertexBufferObjectManager());
      T2 = new Sprite(pX,pY,36,160, base_activity.aspa2_textureRegion,
                        base_activity.getVertexBufferObjectManager());
      break;
```

Exemple de creació dels sprites a la classe Obstacles.

```
this.MundoFisico.registerPhysicsConnector(new PhysicsConnector(o1.T1,
                                                  body_T1, true, true));
final RevoluteJointDef revoluteJointDef = new RevoluteJointDef();
                        revoluteJointDef.initialize(body_T1, body_T2,
                                              body_T1.getWorldCenter());
revoluteJointDef.enableMotor = true;
revoluteJointDef.motorSpeed = 2;
revoluteJointDef.maxMotorTorque = 200;
MundoFisico.createJoint(revoluteJointDef);
```

Exemple de creació dels obstacles de tipus 3 a la classe 'Juego'

El disseny de la barra lateral en el Hud:

La barra lateral o bateria d'objectes serveix com a imatge de contenidora dels vuit objectes d'associació que s'apilen verticalment a la pantalla.Està definida dins d'un Hud perquè no es desplaci amb la càmera.

```
Sprite barra_fondo = new Sprite(2, pos_barra,
base_activity.mTextureRegion,base_activity.getVerte
                                xBufferObjectManager());

hud.attachChild(barra_fondo);
```

Exemple de creació de la barra en el Hud.

La barra d'energia en el Hud:

La barra d'energia està definida íntegrament dins de la classe Energia. Està formada per línies i rectangles que componen les diferents àrees de disseny com els laterals, la barra de progrés i el fons. S'integra dins un Hud per mantenir la posició fixa.
Inicialment el color està definit com verd  i tal com va decrementant es modifica el color a groc, i vermell quan li queda poca energia.

Exemple de progressió la barra d'energia.

```
final Energia b_e1 = new Energia(pX, pY, pWidth,
                              pHeight,vertexBufferObjectManager);
  b_e1.setBackColor(0, 0, 0, 1); //Color fons Negre
  b_e1.setFrameColor(0.7f, 0, 0.8f, 1); //Vores violeta
  b_e1.setProgressColor(0, 1.0f, 0.f, 1); //Verd a  l'inici
  hud.attachChild(b_e1.mBackgroundRectangle);
  hud.attachChild(b_e1.mProgressRectangle);
  for (int i = 0; i < b_e1.mFrameLines.length; i++)
          hud.attachChild(b_e1.mFrameLines[i]);
```

Exemple d'assignació de la barra d'energia a la classe Juego.

El So.

El so es defineix a la classe BaseActivity i s'activa en les opcions del motor de AndEngine.El joc té cinc sons diferents definits. Aquests sons s'han obtingut del recurs web http://soundbible.com/free-sound-effects-4.html i http://www.freesound.org/.
S'han definit els següents sons:

- objectes que surten en la pantalla.
- col lisió amb acert.
- objectes que surten en la batería (objectes a associar).
- col lisió amb errada.
- gravetat del jugador quan cau (abans dels canvis).

```
try {
  //Objectes de  surtida
  s1 = SoundFactory.createSoundFromAsset(this.getSoundManager(),
                                         this, "out_obj.mp3");
```

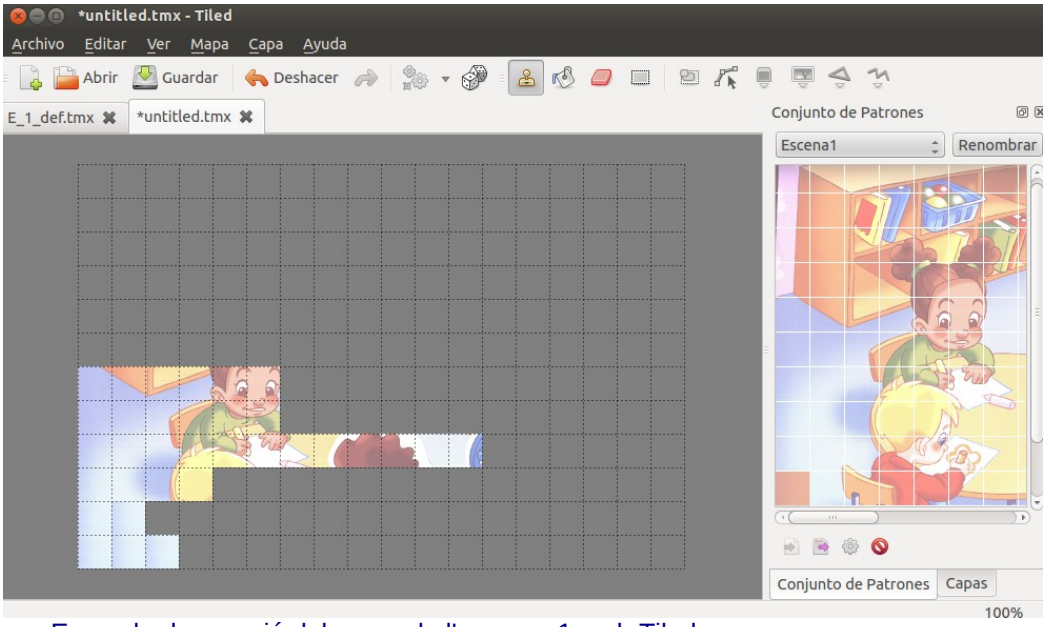
Exemple de recurs de so

Recursos gràfics obtinguts de recursos free en la web
http://www.freedesign4.me/
http://opengameart.org
i elaboració pròpia

## 4.5 Disseny de Mapes. TiledMap.

El disseny de fons de l'escenari s'ha creat amb l'editor Tiled que ens permet crear mapes en la vista clàssica dels jocs 2D del qual es tracta.

Per a la creació del mapa, es crea una textura (e1_textura.png) d'una mida de 96x160 píxels. Aquesta textura es compon de 3 columnes per 5 files. Cada porció de "Tiled" té una mida de 32x32 píxels.
Amb aquesta textura es defineix tot el mapa de l'escena de tipus ortogonal. El mapa té unes dimensions de 18x12, on per cada posició la mida del Tiled és de 32x32 pel que resulta un mapa amb una dimensió total de 576x384 píxels.




Exemple de creació del mapa de l'escena_1 amb Tiled.

Aquesta aplicació transforma el mapa en un fitxer tmx:

```
<?xml version="1.0" encoding="UTF-8"?>
<map version="1.0" orientation="orthogonal" width="18" height="12"
tilewidth="32" tileheight="32">
 <tileset firstgid="1" name="Escena1" tilewidth="32" tileheight="32">
     <image source="img/e1_textura.png" trans="ff00ff" width="576"
height="384"/>
 </tileset>
 <layer name="Capa de Patrones 1" width="18" height="12">
```

```
  <data encoding="base64" compression="gzip">
    H4sIAAAAAAAAAy3MV1cOAAAA0M/eUlZkhGRk75DsUfbIOEdkZa9IsndWHGTLJrL3Fqk
jIzt+jvvwPdzXWy4QCJSnAhWpRGWqUJVqVKcGNalFbUKoQyhh1KUe9WlAQ8JpRGMiaEJTm
tGcSFrQklZE0Zpo2tCWdrQnhg50pBOd6UJXutGdHvSkF73pQyx96Ud/4hhAPAMZxGCGMJR
hDGcEIxlFAomMZgxjGcd4JjCRSUxmCklMZRrTmcGyoJkkM4vZpDCHucxjPgtIZSGLWMwSl
rI8aAUrWcVq0ljDWtJZRwbryWQDG9nEZrawlW1sZwc72cVu9pDFXvaxnwMcJJtDHOYyRzj
KMXI4zglOcorTnOEs58jlPBe4yCWecIWrXOM6edzgJvnc4jZ3uMs97vOAhzzicfB5yjOe8
4KXvOI1b3hLAe94TyEfKKKYj0ElfOIzX/hKKd/4zg9+8ovf/KGMv/zjP5zAs5JgAwAA
  </data>
 </layer>
</map>
```


El resultat se transforma en el fitxer **E_1_def.tmx** que es carrega en la classe Escena1, com es pot observar a continuació en l'exemple:

```
//Recursos Escena1.
      try {
      final TMXLoader tmxLoader = new
      TMXLoader(BaseActivity.getSharedInstance().getAssets(),
      BaseActivity.getSharedInstance().getTextureManager(),
      TextureOptions.BILINEAR_PREMULTIPLYALPHA,
      BaseActivity.getSharedInstance().getVertexBufferObjectManager(),
                        new ITMXTilePropertiesListener() {
                        @Override
                        public void
      onTMXTileWithPropertiesCreated(final TMXTiledMap pTMXTiledMap,
      final TMXLayer pTMXLayer, final TMXTile pTMXTile, final
      TMXProperties<TMXTileProperty> pTMXTileProperties) {
                        }
                  });
                  mTMXTiledMap =
                  tmxLoader.loadFromAsset("txm/E_1_def.tmx");

            }
            catch (final TMXLoadException e)
                  {
                  Debug.e(e);
                  }

            tmxLayer = mTMXTiledMap.getTMXLayers().get(0);

}
```

També és important indicar que el joc està preparat per poder incorporar escenes diferents cada X nivells de joc cadascuna amb una grandària i disseny diferent si cal.

### 4.6 Les classes.

El joc està format íntegrament per deu classes que forme part del paquet *com.example.proyecto_tfg* i que es detallen a continuació:

| Classes | Descripció |
|---|---|
| **BaseActivity** | La classe BaseActivity és l'arrel del joc, que conté els mètodes necessaris per generar les opcions del motor així com la creació i configuració dels recursos del joc. |
| **Energia** | Classe que genera la barra d'energia o barra de progrés. |
| **Escena1** | Aquesta classe carrega un mapa de 'tils' de tipus TMX ubicat al recurs assets dins el recurs txm generat amb el programa Tiled |
| **Help** | Classe per a representar el menú d'ajuda. |
| **Juego** | Classe base del joc. |
| **MenuInicialScene** | Classe que visualitza el menú inicial del joc. |
| **Niveles** | Classe que defineix tots els nivells del joc. |
| **Numbers** | La classe Numbers és la que genera els objectes d'associació. |
| **Obstacles** | La classe obstacles genera els objectes de tipus obstacle. |
| **Personaje** | Classe que crea el Personatge o Player en el joc. |

De totes les classes es realitza una descripció dels mètodes més rellevants:

**La classe BaseActivity:**

`float getDisplayRatio()`
Mètode per a calcular la mida real de pantalla del dispositiu.

`static BaseActivity getSharedInstance()`
Mètode que retorna la instància de la classe.

`void MsgToast(java.lang.String msg)`
Missatge que informa dels nivells.

`org.andengine.engine.options.EngineOptions onCreateEngineOptions()`
Aquest mètode afegeix les opcions de motor del joc

`void setCurrentScene(org.andengine.entity.scene.Scene scene)`
Mètode per a canviar l'escena.
Mètodes de la classe BaseActivity.

**La classe Energia:**

`void setBackColor(float pRed, float pGreen, float pBlue, float pAlpha)`
Color de la barra de progrés

`void setFrameColor(float pRed, float pGreen, float pBlue, float pAlpha)`
Color dels laterals de la barra.

`void setProgress(float pProgress)`
Estableix el progrés actual a la barra de progrés.

`void setProgressColor(float pRed, float pGreen, float pBlue, float pAlpha)`
Color de la barra de progrés.
Mètodes de la classe Energia.

**La classe Juego:**

`void Add_nivel()`
Afegeix nivell

`int[] aleatorio()`
Es genera un vector perquè no es repeteixin els objectes d'associació a la bateria

`int devuelve_letra()`
Inspecciona si existeix un objecte en la batería d'objectes que no estiga en la pantalla coma objecte de surtida.

`int get_F()`
Segons el nivell surten diferents objectes d'associació

`int Get_nivel()`
Retorna el nivell actual

`static Juego getSharedInstance()`
Recupera instància del Joc.

`void initOnScreenControls()`
ScreeNControl i moviment del Player mitjançant funcions de força.

`void onAccelerationAccuracyChanged(org.andengine.input.sensor.acceleration.AccelerationData pAccelerationData)`

`void onAccelerationChanged(org.andengine.input.sensor.acceleration.AccelerationData pAccelerationData)`
Control d'acceleració.

`scene pauseScene()`
Botó del Pause

`void Set_Alpha(int tipo)`
Mètodes de la classe Juego

## La classe Niveles

`static int[][] Carga_nivel(int level)`

Mètode que donat el nivell actual de joc, retorna el nivell d'obstacles corresponent.

Mètode de la classe Niveles.

## La classe Numbers

**Constructors i descripció**

`Numbers(int flotantes)`

Contructor que genera els objectes de surtida en diferents possicions aleatories de la pantalla.

`Numbers(int x, int y, int flotantes, int id)`

Aquest constructor genera els objectes d'associació a la bateria.

Constructors de la classe Numbers

`void add_body(com.badlogic.gdx.physics.box2d.Body b1)`

Assignació del Body a l'objecte en la classe Juego quan es genera l'objecte de surtida.

`TiledSprite getSprite()`

Retorna el sprite de la matriu segons el id.

Mètodes de la classe Numbers

## La classe Obstacles

**Constructor i descripció**

`Obstacles(int pX, int pY, int tipo)`

Contructor que genera els obstacles desde la classe Juego

**Paràmetres:**

pX- Coordinada X de l'objecte obstacle

pY- Coordinada Y de l'objecte obstacle

tipo- tipus d'obstacle que ve definit per el mètode de la classe Niveles: Carga_nivel(Get_nivel())[y][x])

Constructor de la classe Obstacles.

### La classe Personaje

**Constructor i descripció**

`Personaje()`
Posiciona el Player en les coordinades inicials, indicades en la classe Juego.
Constructor de la classe Personaje.

`getSprite()`
public org.andengine.entity.sprite.AnimatedSprite getSprite()
Retorna el sprite Player.

**Retorna:**

el sprite(Player)

Mètode de la classe Personaje.

# 5. Conclusions

Aquest projecte es pot dir que té un abans i un després, com dos pols oposats que convergeixen en una única direcció i amb la satisfacció de veure com es poden aconseguir objectius proposats que difícilment, en els albors del projecte, s'hagués imaginat. Si parlem de l'abans del projecte, ens podem referir a tot un seguit d'eines, utilitats i metodologies apreses que han servit per arribar a un objectiu concret, i si parlem d'un després ens podem referir a l'obtenció del producte final, tot un aprenentatge transformat en una finalitat determinada: utilitzar l'oci com una eina o utilitat que ajudi a formar i desenvolupar als nens a través d'una tecnologia portable.
Aquest projecte ha servit per conèixer a grans trets, algunes de les tècniques bàsiques per a l'aprenentatge en el desenvolupament dels videojocs.Ha servit per aprendre que un videojoc no és un "joc", ja que per a l'elaboració de videojocs, són necessaris uns coneixements i tècniques avançades de programació específica per dur-los a terme, i

sobretot cal especialitzar-se amb el tipus de tecnologia que es va a utilizar.

Coneiximents adquirits:
Cal dir que abans de començar el projecte, l'experiència amb aquest tipus de tecnologies dins de l'àmbit dels videojocs era totalment nul · la. Tant en el desenvolupament com en l'ús dels videojocs.
Per tant, els coneixements adquirits en aquest projecte es poden resumir en els següents punts principals:

- Muntatge d'un entorn de programació de Andengine per Android.
- Coneixement de les diferents biblioteques o llibreries per a la programació amb AndEngine.
- Coneixement del SDK d'Android específiques per al desenvolupament d'aplicacions mòbils.
- Adaptació d'un videojoc a un tipus concret d'usuaris.
- Ús d'un Hud: separació entre la part que ensenya la informació i l'estat del joc.
- Com millorar la 'jugabilitat' dels jocs.
- Cal disposar de testers que donen les seues impresions.

Principals dificultats:
Les principals dificultats trobades durant l'elaboració de tot el projecte han estat:

- Aconseguir una òptima jugabiltat.
- La movilitat del jugador:
  - inicialment el jugador es movía tocant sobre el jugador amb el dit.
  - inicialment s'utilitzen funcions de força dins l'entorn de gravetat.
  - posteriorment es modifica la movilitat utilizant funcions de velocitat.
  - més endavant s'anula la gravetat aplicada al jugador.
  - tots aquestos canvis comporten un canvi total del disseny i de l'estructura de l'índex del jugador.
- Canvis aplicant les opcions de gravetat en el jugador.
- No disposar del material adequat. Encara que sembli absurd no s'ha disposat d'un dispositiu mòbil basat en Android per realitzar les proves i ha estat un error molt gran. Les proves de testing s'han realitzat mitjançant l'AVD (dispositiu virtual d'Android) d'Eclipse. Els resultats utilitzant aquest dispositiu no són els esperats, sobretot usant les llibreries de física.
- Posicionament d'objectes en Hud's.
- Afegir un botó tàctil de Pause.
- Adaptar el joc al tipus d'usuari.
- Que el joc sigui entretingut a cada un dels diferents nivells.

Linies de treball pendents:

Respecte als temes pendents cal dir que el projecte no s'ha pogut conclure.A més, seria recomanable donar fi a una serie de tasques per obtenir un producte amb un índex alt de qualitat:

- Millorar els gràfics.
- Possar-li un nom al joc.
- Poder oferir el joc en diversos idiomes o fer-ho tot en anglès.
- Afegir les opcions d'ajuda en el menú.
- Afegir opcions de so, nivells de dificultat ...
- Oferir al jugador una recompensa o satisfacció cada vegada que superi algun nivell.
- Millorar detalls: barra d'energia, afegir escenes de fons segons nivells, augmentar els nivells i configurar-los de forma correcta, mesurar el temps d'inici entre nivells ...
- Inserir més tipus d'obstacles.
- Crear un nou projecte amb les idees més clares des de l'inici.

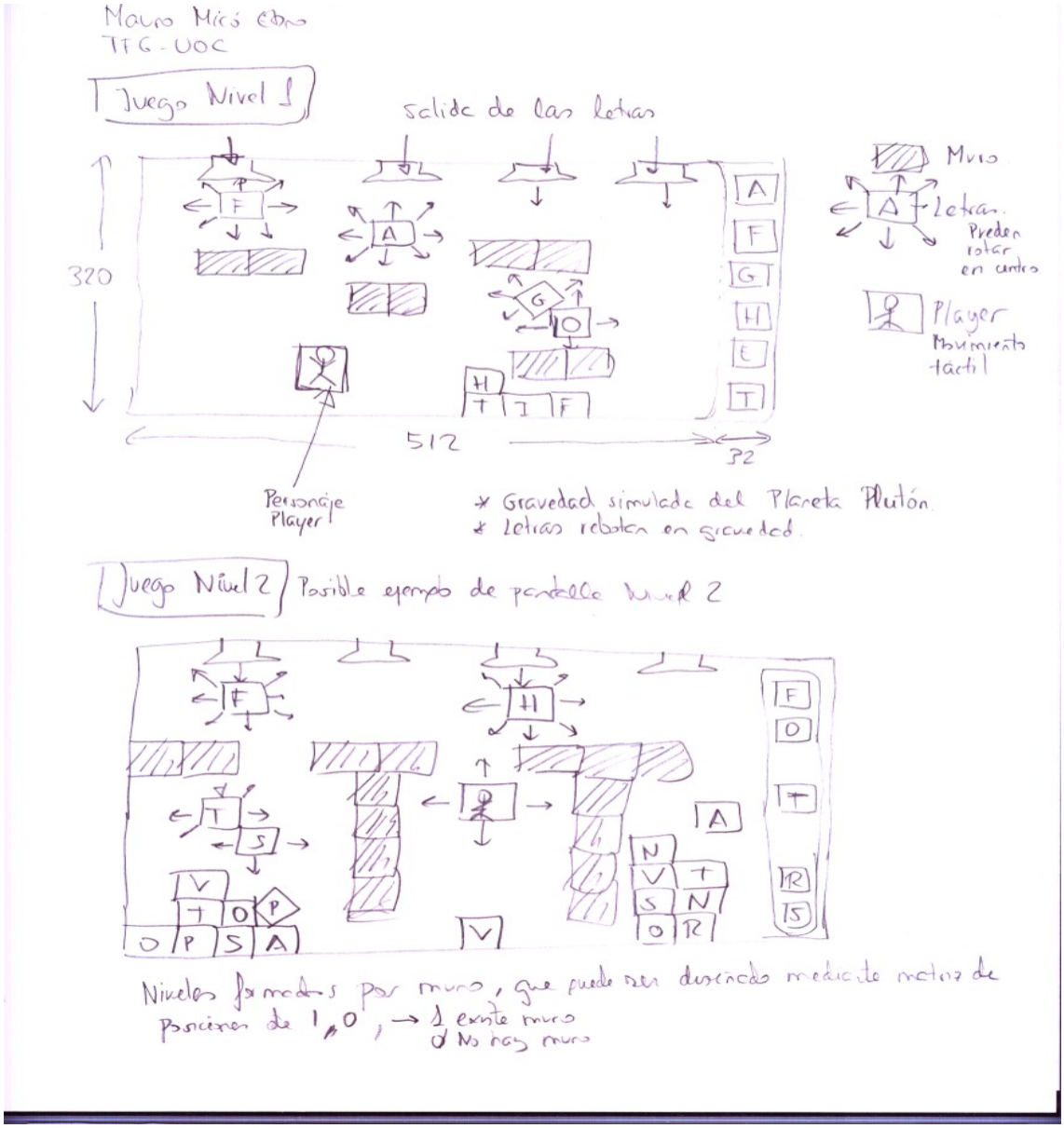
# 6. Glossari

- **Tile:** Una tile és la part gràfica de cada videojoc que pot ser utilitzada per completar parts d'un fons per mitjà d'un tileset (conjunt de tiles). Cada tile és diferent i bàsicament de la mateixa mida, en els videojocs els tiles estan col · locats en un tileset. Cada tileset tindrà la combinació bàsica de certs llocs del joc.
- **ScreeNControl:** Una pantalla tàctil (touchscreen o ScreeNControl en anglès) és una pantalla que mitjançant un contacte directe sobre la seva superfície permet l'entrada de dades i ordres al dispositiu.
- **AVD**: L'AVD (Android Virtual Device) Manager és una utilitat proporcionada per Google com a part del SDK d'Android que et permet crear i emular dispositius Android.
- **HUD:** el HUD (heads-up display) és el mètode pel qual la informació es transmet visualment al jugador com a part de la interfície d'usuari d'un joc.

# 7. Bibliografia


- http://www.andengine.org
- http://www.andengine.org/forums/gles2/
- https://github.com/nicolasgramlich/AndEngine
- http://code.google.com/p/andengineexamples/
- http://developer.android.com/about/dashboards/index.html
- http://developer.android.com/index.html
- http://box2d.org
- www.iforce2d.net
- http://jmonkeyengine.org/wiki/doku.php/jme3:advanced:physics
- http://andengine.wikidot.com/detect-screen-resolution
- http://www.tinhtruong.me/2012/03/how-to-import-git-projects-from.html
- http://insomniacgeek.com/how-to-create-a-new-eclipse-project-and-push-it-to-a-new-git-enabled-bitbucket-repository/
- http://papeltank.blogspot.com.es/2011/11/git-repository-bitbucketorg-ubuntu-1110.html
- http://code.google.com/p/bot-wars/source/browse/src/botwars/main/BotWars.java?r=73febc87c2ab5cba5b3e6a2f180a1e48586ad0f5
- http://paulsonapps.wordpress.com/2010/06/09/tutorial-tiled-mapeditor-part-1-create-a-tmxtiledmap-game/
- http://www.aprendiendodeandroidymas.com/2012/07/utilizando-tile-maps-en-andengine.html
- http://www.mapeditor.org/
- http://es.wikipedia.org/wiki/Arcade
- http://en.wikipedia.org/wiki/HUD_(video_gaming)
- http://opengameart.org
- http://www.freedesign4.me/
- Revista APLICACIÓN EDUCATIVA DE LOS VIDEOJUEGOS. Educar en el 2000 / abril 2004 José Manuel Saz Rubira


Enllaços consultats durant tot el període d'elaboració del projecte.

# 8. Annexos

Imatge del croquis del joc en els seus inicis: