Implementació d'una aplicació multi-jugador a temps real per Android

Memòria del Treball Final de Màster

Màster Universitari en Aplicacions Multimèdia
Itinerari professional
Universitat Oberta de Catalunya

Autor: Gabriel Reinés March

Consultor: Sergio Schvarstein Liuboschetz
Professor: David García Solórzano

22 de juny de 2014

Crèdits/Copyright

Als meus pares Sebastià i Catalina,

per la seva inestimable ajuda al llarg de tot el projecte,

i a na Maria, pel seu suport incondicional i la seva paciència infinita.

Abstract

Nowadays, mobile devices are in everybody's life, and thanks to the new communication technologies (ICT), the phenomena of ubiquity has been achieved, so that a user can be uninterruptedly connected to the internet, regardless of its location. One of the main niches of the mobile entertainment market is the applications' one, specially the games sector. For this reason, this Master's Thesis has focused on the design and development of a real-time multiplayer game in Java, for contributing to the mobile media content market as well as for improving the knowledge in Android development. The game engine AndEngine has been used in this project. It consists in a set of open-source libraries which provides the different functions and interfaces needed for conceiving a game. The application follows a client-server scheme, communicated through a TCP-IP interface. The final product consists of a completely functional Air-Hockey-type application with a real-time communication interface, which allows the player to control his or her own paddle and see the movement of the opponent's one on the same device.

Resum

Actualment els dispositius mòbils estan presents en la vida quotidiana de qualsevol persona, i gràcies a les noves tecnologies de telecomunicació, s'ha aconseguit el fenomen de la ubiqüitat, de forma que l'usuari pot estar ininterrompudament connectat a la xarxa independentment de la seva ubicació. Un dels nínxols importants del mercat d'oci mòbil és el de les aplicacions multimèdia, concretament el sector dels jocs. Per aquest motiu, el Treball Final de Màster s'ha centrat en el disseny i desenvolupament d'un joc multi-jugador a temps real en Java, amb l'objectiu de contribuir en el mercat de continguts per a dispositius mòbils i aprofundir en els coneixements de desenvolupament Android. Per a portar-ho a terme s'ha fet ús del motor de joc AndEngine, un conjunt de llibreries de codi obert que proporciona les funcions i interfícies necessàries per a facilitar el desenvolupament d'un joc. L'aplicació s'ha desenvolupat seguint un esquema client-servidor, comunicats entre si mitjançant una interfície de comunicació TCP-IP. El producte final consisteix en una aplicació tipus "Air Hockey" completament funcional amb una interfície de comunicació a temps real que permet controlar la pròpia paleta i observar els moviments de l'altre jugador a sobre d'un mateix dispositiu.

Paraules clau

Android, aplicació, joc, AndEngine, multi-jugador, temps real, comunicació

Notacions i Convencions

Per al contingut textual de la memòria es fan ús de les següents tipografies:

- Títol de capítol: Arial negreta, mida 20

1. Introducció

- Secció de primer nivell: Arial negreta, mida 13

2. Objectius generals

- Subsecció de primer nivell: Arial negreta cursiva, mida 10

3.1 Estudi de mercat

- Text: Arial, mida 10

Descripció general del tema des d'un punt de vista personal

- Codi font: Consolas, mida 10, amb colors corresponents a la funció de cada element

```
public boolean myFunction(int p1){
	this.mValue = p1;
	if(p1 > 0){
		return true;
	}
	return false;
}
```

Índex

Capítol 1: Introducció 9

1.Introducció i motivació del projecte 9

2. Descripció de la proposta de treball 10

3. Objectius generals i abast del projecte 13

3.1 Objectius principals 13

3.2 Objectius secundaris 13

3.3 Abast del projecte: 14

4. Metodologia i procés de treball 15

4.1 Estratègia utilitzada i metodologia 15

4.2 Recursos utilitzats 16

5. Planificació 18

5.1 Taula de fites 18

5.2 Diagrama de Gannt 18

5.3 Canvis en la planificació inicial 19

6. Pressupost 20

7. Estructura de la resta del document 21

Capítol 2: Anàlisi 22

1. Estat de l'art 22

2. Anàlisi del mercat 28

2.1 Estudi de mercat 28

2.2 Comparativa de la competència 28

2.3 Estratègia de màrqueting 29

3. Públic objectiu i perfils d'usuari 34

3.1 Perfils d'usuari 34

Capítol 3: Disseny 37

1. Arquitectura general de l'aplicació 37

1.1 Model de distribució i comunicació 37

1.2 Arbre de navegació 37

1.3 Interfície gràfica 39

1.4 Llibreries externes 41

2. Descripció detallada dels components de l'aplicació 42

2.1 MainActivity 42
2.2 MainMenuScene 43
2.3 GameActivity 46
3.3 GameServer 49
3.4 GameClient 52
3.5 Altres classes implementades a l'aplicació 54

Capítol 4: Implementació 56
1. Requisits d'instal·lació 56
2. Instruccions d'instal·lació 57

Capítol 5: Demostració 58
1. Vídeo demostratiu 58

Capítol 6: Conclusions i línies de futur 59
1. Conclusions 59
1.1 Assoliment d'objectius 59
1.2 Treball realitzat 59
1.3 Seguiment de la planificació 59
2. Línies de futur 61

Bibliografia 62
Annexos 63
Annex A: Lliurables del projecte 63

Figures i taules

Índex de figures

Figura 1. Diagrama de Gannt inicial....18
Figura 2. Diagrama de Gannt actualitzat....19
Figura 3. Diagrama de navegació de l'aplicació....38
Figura 4. Interfície primera activitat....39
Figura 5. Paràmetres de connexió: servidor (esq.) i client (dreta)....40
Figura 6. Pantalla del joc en acció (esq.) i menú de pausa (dreta) 41
Figura 7. Diagrama de classe de MainActivity....43
Figura 8. Diagrama de classe de MainMenuScene 46
Figura 9. Diagrama de classe de GameActivity....47
Figura 10. Diagrama de classe de GameServer....52
Figura 11. Diagrama de classe de GameClient 52
Figura 12. Diagrama de classes de GameClientMessages....54

Índex de taules

Taula 1. Taula de fites del projecte....18
Taula 2. Pressupost del projecte 20
Taula 3. Comparativa de diferents aplicacions existents en el mercat....29
Taula 4. Missatges de client....54
Taula 5. Missatges de servidor....55

Capítol 1: Introducció

1.Introducció i motivació del projecte

Actualment els dispositius mòbils han esdevingut objectes d'ús quotidià per la gran majoria de la població. Gràcies a les noves tecnologies de telecomunicació, l'usuari pot estar ininterrompudament connectat a la xarxa independentment de la seva ubicació i sense necessitat de cap infraestructura física. Per aquest motiu, i d'acord amb les dades proporcionades per (SuperMonitoring, 2013), els usuaris actualment consumeixen més temps davant del dispositiu mòbil que davant de l'ordinador, i més d'un 30% del temps dedicat al mòbil és per jugar. Prova d'aquesta demanda d'oci és el creixement exponencial que han experimentat les diferents tendes d'aplicacions o App Stores dels principals proveïdors. En el cas d'Android, en els últims 12 mesos s'han duplicat el nombre d'aplicacions disponibles a Google Play. Segons (AppBrain, 2014), al febrer de 2014 hi havia més de 1.100.000 aplicacions disponibles a la tenda, respecte a les 600.000 publicades el mateix mes de 2013.
Amb aquesta introducció es vol emfasitzar la importància del desenvolupament d'aplicacions per a dispositius mòbils actualment, i per aquest motiu s'ha triat aquest camp com a temàtica del Treball Final de Màster.

La motivació personal per aquest tema és doble: per una part, ampliar els meus coneixements de programació en una nova plataforma i explorar les diferents interfícies de comunicació que presenta; i per altra, poder ocupar una petita part dins el món dels desenvolupadors mòbils, contribuint amb diferents aplicacions a la varietat i qualitat del mercat Google Play.

Ampliar coneixements sempre és un repte motivador, ja que permet augmentar les teves fronteres laborals i personals. A més de ser una contribució nova al currículum actual, la possibilitat de tenir i dominar una eina tan poderosa actualment, com és la programació d'aplicacions natives per Android, és un estímul personal cap a la millora contínua i cap a l'ampliació de la formació.
Per altra banda, el mercat de les aplicacions és una possibilitat de negoci molt temptadora, ja que no presenta cap risc. Amb les plataformes actuals de distribució d'aplicacions, qualsevol persona amb un ordinador pot concebre una aplicació i pujar-la a aquests portals, que actuen com a mostradors a escala mundial de la teva obra. L'èxit de la mateixa és el que determinarà directament els ingressos del negoci: si es crea una app mestra i aconsegueix la fama i distribució necessàries per a l'èxit, ens trobem amb els casos com *Candy Crush Saga*, amb la desorbitant xifra de 567 milions de dòlars de benefici l'any 2013 per part de l'empresa King Digital (Forbes, 2014). Si per contra l'aplicació queda relegada a unes quantes desenes de descàrregues, l'únic que s'ha "perdut" en l'empresa individual són les hores que el programador hi ha dedicat. Per tant, sembla un mercat fàcil d'entrar-hi, fet motivador de cara a experimentar i a poder publicar els treballs realitzats.

2. Descripció de la proposta de treball

La temàtica del treball final de grau és el desenvolupament d'una aplicació per a dispositius mòbils Android. Una aplicació (també coneguda com a app, de l'anglès *application*) és una peça de software pensada i dissenyada per a ser executada sobre un telèfon intel·ligent, tauleta, Smart TV, etc. El desenvolupament d'aplicacions mòbils ha tingut un creixement exponencial durant els darrers anys, propulsat principalment per dos factors: per una part, l'augment de la capacitat de processament dels dispositius mòbils; per altra part, l'elevada disponibilitat de les diferents eines de desenvolupament d'aplicacions. Aquests dos fets han motivat que la comunitat global de desenvolupadors de software hagi pogut accedir al mercat de creació d'aplicacions mòbils, generant d'aquesta manera una oferta diversa i extensa.

Les tipologies d'aplicacions presents avui en dia al mercat són molt variades:

- **Aplicacions basades en la transferència d'informació**: el fonament d'aquest tipus d'aplicació és mostrar dades actualitzades (imatges, text, àudio...), com per exemple una aplicació de resultats esportius, receptes de cuina, previsió del temps, agendes personals... Dins d'aquest grup s'hi situen majoritàriament les *web apps*, que són aquelles programades amb els llenguatges estàndards del web (HTML5, CSS, Javascript). L'avantatge d'aquestes aplicacions és la facilitat d'actualització de les dades, ja que cada cop que l'usuari arranca l'aplicació aquesta es connecta al servei web que proveeix la informació, i per tant es carreguen les dades actualitzades.

- **Aplicacions basades en el hardware del dispositiu**: degut al gran nombre de sensors que incorporen els dispositius mòbils actuals (GPS, giroscopi, càmera fotogràfica...), és possible crear aplicacions que en facin ús i proporcionin informació a l'usuari basant-se en les dades recollides pel conjunt de sensors. Exemples d'aquesta categoria són els navegadors GPS, aplicacions llanterna (fent ús del flaix de la càmera fotogràfica), apps de realitat augmentada, brúixoles...

- **Jocs**: és el tipus d'aplicació més nombrós, i el fonament d'aquesta és entretenir l'usuari. Dins d'aquesta categoria podem fer moltes classificacions: en funció de la temàtica (plataformes, puzzles, rol...), gràfics (2D/3D), nombre de jugadors (un jugador, multi-jugador), etc.

Actualment, d'acord amb les dades presentades per (SuperMonitoring, 2013), Android és la plataforma més estesa, instal·lat a més de 1000 milions de dispositius arreu del món. Aquest és un factor a tenir en compte a l'hora de determinar el sistema operatiu sobre el qual correrà l'aplicació, ja que en determina el nombre d'usuaris potencials.

L'aplicació que es vol desenvolupar durant el treball de final de màster és un joc, per tant s'engloba dins de la tercera categoria esmentada anteriorment. Les característiques del mateix es mostren a continuació:

- **Tipologia**: esports, hoquei taula.

- **Objectiu del joc**: aconseguir que la pilota entri a la porteria del rival.

- **Interacció**: l'usuari ha de moure la paleta amb el dit mitjançant la interfície tàctil del dispositiu.

El tret diferencial d'aquesta aplicació respecte a la majoria de les ja existents a la plataforma Google Play és la capacitat de poder jugar en mode multi-jugador des de dos dispositius diferents. D'acord amb les característiques de les aplicacions semblants, aquestes permeten la interacció entre dos jugadors, però actuant sobre un sol dispositiu. Per tant aquest nou enfocament suposa una innovació en aquest àmbit.

L'aplicació es desenvoluparà en Java, ja que es tracta d'una aplicació nativa per a dispositius Android. Aquesta decisió s'ha pres degut a les característiques de la mateixa, ja que l'aplicació fa ús d'algunes funcionalitats hardware del dispositiu (per exemple la interfície WiFi), i Android proveeix les APIs Java necessàries per a poder gestionar-ho eficientment.

Inicialment, en instal·lar l'aplicació, s'hauran de concedir els permisos d'ús de funcionalitats d'internet i de la interfície WiFi del dispositiu (entre d'altres permisos), ja que la comunicació a temps real entre els dos jugadors es farà mitjançant aquest canal.
Un cop instal·lada, l'usuari tindrà la possibilitat de crear una partida o bé d'unir-se a una ja existent. Tant per a anunciar la creació d'una partida com per connectar-se a una partida ja creada, s'haurà d'activar la interfície WiFi del dispositiu.

Per al desenvolupament de la lògica, la part gràfica i la comunicació de l'aplicació s'ha fet ús d'un motor de joc. Concretament s'ha emprat AndEngine (Gramlich, AndEngine: Free Android 2D OpenGL Game Engine, 2010), un motor de joc de codi obert creat per N. Gramlich. L'avantatge d'usar aquest component és l'abstracció que porta a terme de les funcionalitats bàsiques que empra un joc, com pot ser la física (moviments, simulació de la gravetat...), el rendering de la interfície, elaboració de menús interactius, gestió de la comunicació, etc.

A mode de resum, el que es vol aconseguir en haver finalitzat el treball és:

- **Creació del joc**: implementació de la lògica del joc. Bàsicament es tracta d'un element gràfic (la pilota del joc) que inverteix el vector de velocitat "y" si interfereix amb l'element de la paleta.

Si per contra la pilota arriba a l'extrem inferior/superior de la pantalla, es comptabilitza un punt a favor del jugador que ha marcat el gol.

- **Generació de la interfície**: creació d'una interfície gràfica sensible als esdeveniments tàctils que ocorrin sobre la pantalla, des de la selecció d'opcions en el menú per mitjà de botons, fins al desplaçament de la paleta sobre el tauler de joc.

- **Implementació de la interfície de comunicació a temps real**: implementació d'un protocol de comunicació a temps real entre dos dispositius, per tal de possibilitar l'execució del joc simultàniament en dos terminals. Aquesta connexió s'implementarà mitjançant la interfície WiFi que proveeix el motor de joc AndEngine, la qual està basada en l'API WiFi que proveeix Android.

El producte final consisteix en una aplicació instal·lable per Android tipus "Air Hockey" completament funcional, amb una interfície de comunicació a temps real que permet controlar la pròpia paleta i observar els moviments de l'altre jugador a sobre d'un mateix dispositiu.

3. Objectius generals i abast del projecte

3.1 Objectius principals

3.1.1 Objectius de l'aplicació:

- Implementar una aplicació multi-jugador tipus “ping-pong” per Android, en la qual els diferents jugadors puguin connectar-se entre si mitjançant una connexió TCP-IP sense necessitat d’un servidor extern.

- Permetre que un jugador pugui escoltar el medi mitjançant la interfície WiFi del dispositiu per trobar altres jugadors potencials, així com crear una partida nova o unir-se a una ja existent.

- Comprendre els protocols de comunicació involucrats en aquest tipus de xarxes, i implementar-los mitjançant les APIs que proveeix la plataforma.

- Minimitzar els requeriments del dispositiu de destí, tant a nivell de hardware (CPU, memòria...) com a nivell de software (versió del SO), per tal d’arribar al màxim d’audiència possible.

3.1.2 Objectius personals de l'autor del TFM:

- Aprofundir en el coneixement de la plataforma Android i les interfícies de comunicació de la mateixa, així com investigar les possibilitats que ofereixen els motors de joc disponibles al mercat.

- Aplicar els coneixements de gestió de projecte adquirits al llarg del màster per a la planificació i seguiment del mateix.

3.2 Objectius secundaris

- Donar a conèixer i promocionar una aplicació pròpia a través dels diferents canals disponibles, com ara Google Play, xarxes socials... per completar així el cicle de creació d’una aplicació multimèdia.

3.3 Abast del projecte:

- L'aplicació en qüestió consistirà bàsicament en dues pantalles (activitats, segons la terminologia de la plataforma): un menú inicial on l'usuari podrà crear una partida o unir-se a una ja existent; i la superfície de joc, a la qual s'hi accedeix un cop establerta la connexió.

- La paleta del jugador es podrà controlar mitjançant la interacció tàctil amb la pantalla del dispositiu.

- La connexió s'establirà i es mantindrà mitjançant un socket TCP implementat amb l'API WiFi corresponent de la plataforma Android, mitjançant les llibreries proporcionades pel motor de joc AndEngine.

Els següents apartats inclouen (de manera no exclusiva) aspectes que no s'implementaran a l'aplicació al llarg del projecte:

- Publicació de les puntuacions en altres plataformes (Google Play, Facebook...).

- Control de la paleta amb elements no descrits específicament més amunt (acceleròmetre, control de veu...).

- Joc multi-jugador en línia (a través d'un servidor extern amb connexió a internet).

4. Metodologia i procés de treball

4.1 Estratègia utilitzada i metodologia

Per a portar a terme el desenvolupament del projecte s'ha adoptat una estratègia de nova creació, és a dir, concebre l'aplicació, dissenyar-la, implementar-la amb les diferents eines software i hardware disponibles, i usar-la com a aplicació funcional, tancant així el cicle de vida de creació d'una aplicació multimèdia. Aquesta estratègia es cenyeix al marc de treball CDIO (CDIO, 2014), desenvolupat per al context educatiu de l'àmbit de l'enginyeria. Aquest es basa en treballar les quatre fases principals del cicle de vida d'un producte –Conceive, Design, Implement, Operate– des de la idea inicial fins al producte final funcional.
Aquesta aproximació ha estat encertada pel tipus de producte creat: una aplicació multimèdia. Degut als objectius proposats més amunt per l'autor, l'estratègia òptima de desenvolupament és crear un producte nou, ja que el que es pretén és contribuir en el mercat d'aplicacions amb un producte relativament innovador i poc freqüent, i per altra part aprofundir en els coneixements de desenvolupament d'aplicacions natives per Android.

D'acord amb l'estratègia descrita, la metodologia emprada pel desenvolupament del projecte respon a un procés iteratiu i incremental, en el qual podem diferenciar quatre fases:

- **Definició del projecte**: elaboració de la proposta de projecte, així com la seva justificació i motivació. Aquesta fase coincideix amb el lliurament de la PAC 1 de l'assignatura.

- **Recerca d'informació i disseny**: durant aquesta etapa es fa una recerca d'informació exhaustiva sobre la matèria de desenvolupament del projecte, i es proposa una definició d'alt nivell de les prestacions del producte, així com també l'abast del projecte. En aquest cas, la part més desconeguda del projecte (i per tant amb més necessitat d'informació) ha estat la relativa al motor de joc emprat. Per saber com dissenyar el joc en funció del motor s'han de conèixer amb profunditat tots els mòduls que implementa, així com les extensions de suport de característiques multi-jugador i de simulacions físiques.

- **Implementació**: etapa caracteritzada per la programació de l'aplicació, de manera iterativa i incremental. Podem diferenciar tres grans blocs en el transcurs de la programació del joc: interfície, lògica i comunicació. Tanmateix, degut a la naturalesa iterativa de la majoria de projectes de software, el contingut dels diferents blocs s'entremescla durant el procés de programació. Per exemple, el renderitzat de la interfície gràfica del terreny de joc va estretament lligat a la interfície de comunicació de l'aplicació, ja que la posició de la paleta de l'adversari es transmet a través dels paquets de missatges que gestiona el servidor, i un cop rebuda es dibuixa sobre la pantalla del dispositiu.

- **Test i lliurament**: l'última etapa del projecte correspon a realitzar diferents proves de funcionalitat sobre diversos dispositius mòbils de diferents característiques, i finalment lliurar tota la documentació explicativa del procés de desenvolupament del projecte.

4.2 Recursos utilitzats

A continuació es llisten els principals recursos utilitzats per a l'elaboració del projecte. A l'apartat de Bibliografia es pot trobar una llista exhaustiva de totes les fonts bibliogràfiques tant físiques com virtuals emprades al llarg de la realització del mateix.

Recursos bibliogràfics:

- **AndEngine for Android Game Development Cookbook** (Schroeder & Broyles, 2013): publicació que recull i desenvolupa les principals característiques del motor de joc AndEngine, i ofereix petits fragments de codi per exemplificar-les.

- **Mòduls de l'assignatura**, emprats per a la correcta elaboració dels esquemes de gestió i planificació del projecte final de Màster.

Recursos digitals:

- **Programming Mobile Applications for Android Handheld Systems** (Porter, 2014): curs a distància realitzat a través de la plataforma d'eLearning Coursera.

Maquinari:

- **PC** amb sistema operatiu Windows 7 32-bit i amb la distribució Java 6 instal·lada.

- **Telèfon intel·ligent** Samsung Galaxy SII amb sistema operatiu Android 4.1.2, emprat com a plataforma d'execució i depuració del codi font.

- **Telèfon intel·ligent** Samsung Galaxy Fame amb sistema operatiu Android 4.1.2, emprat com a segon dispositiu per a provar la interfície de comunicació de l'aplicació.

Programari:

- **Android Development Tools (ADT)**: entorn de desenvolupament integrat basat en Eclipse que conté tots els *plugins* necessaris per al desenvolupament d'aplicacions Android.

- **Microsoft Word**: contingut textual de les PACs.

- **Microsoft Project**: elaboració de la planificació del projecte.

5. Planificació

5.1 Taula de fites

En aquest apartat es mostra la taula de fites del projecte actualitzada, així com les dates clau associades a cada fita:

Nom de la fita	Data assoliment
Proposta de projecte (PAC 1)	17/03/2014
Mandat del projecte (PAC 2)	31/03/2014
Interfície de comunicació (PAC 3)	01/05/2014
Prototip interfície gràfica (PAC 4)	26/05/2014
Aplicació finalitzada	15/06/2014
Entrega projecte (PAC 5)	22/06/2012

Taula 1. Taula de fites del projecte

5.2 Diagrama de Gannt

En aquest apartat es mostren els diagrames de Gannt inicial i actualitzat del projecte. Aquest últim ha estat modificat d'acord amb les desviacions temporals sorgides i amb les noves tasques que han sorgit al llarg del projecte. Les fites del projecte estan marcades en vermell i representades per un rombe a sobre de l'eix temporal:

Diagrama de Gannt inicial:

	Nombre de tarea	Duración	Comienzo	Fin
1	**PAC3 - Interfície de comunicació**	**21 días**	**mar 01/04/14**	**lun 28/04/14**
2	Estudi APIs Bluetooth	2 días	mar 01/04/14	mié 02/04/14
3	Disseny app echo	5 días	jue 03/04/14	mar 08/04/14
4	Implementació app echo	14 días	mié 09/04/14	lun 28/04/14
5	**Lliurament PAC 3**	**0 días**	**lun 28/04/14**	**lun 28/04/14**
6	**PAC4 - Interfície gràfica**	**20 días**	**mar 29/04/14**	**lun 26/05/14**
7	Estudi motor de joc	5 días	mar 29/04/14	sáb 03/05/14
8	Disseny lògica	4 días	dom 04/05/14	mié 07/05/14
9	Disseny UI	3 días	jue 08/05/14	lun 12/05/14
10	Integració motor de joc	10 días	mar 13/05/14	lun 26/05/14
11	**Lliurament PAC 4**	**0 días**	**lun 26/05/14**	**lun 26/05/14**
12	**PAC5 - Integració i test**	**15 días**	**mar 27/05/14**	**lun 16/06/14**
13	Integr motor joc (cont)	11 días	mar 27/05/14	mar 10/06/14
14	Test	4 días	mié 11/06/14	sáb 14/06/14
15	Empaquetament	2 días	dom 15/06/14	lun 16/06/14
16	**Documentació**	**56 días**	**mar 01/04/14**	**lun 16/06/14**
17	Redacció memòria	56 días	mar 01/04/14	lun 16/06/14
18	Presentació	5 días	mar 10/06/14	lun 16/06/14
19	**Lliurament Projecte (PAC 5)**	**0 días**	**lun 16/06/14**	**lun 16/06/14**

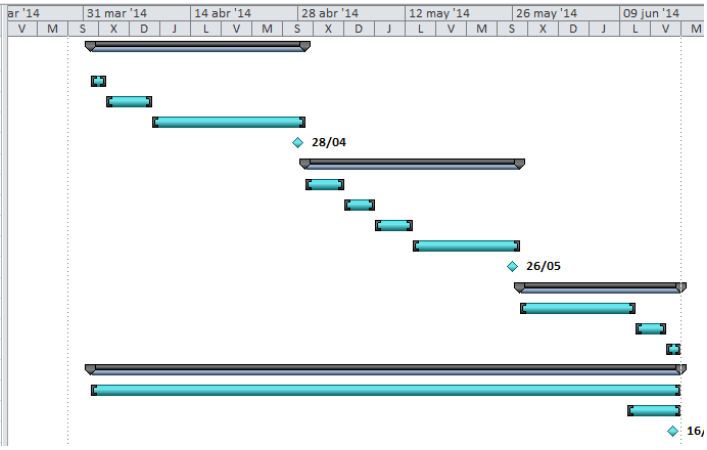
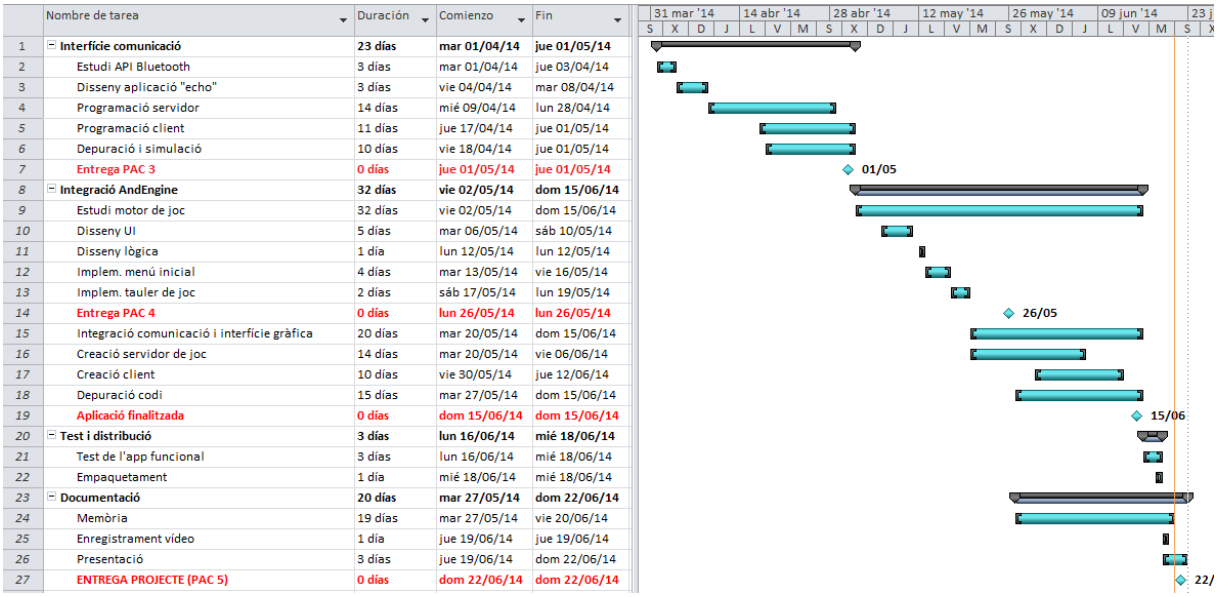


Figura 1. Diagrama de Gannt inicial

Diagrama de Gannt actualitzat:



	Nombre de tarea	Duración	Comienzo	Fin
1	**Interfície comunicació**	**23 días**	**mar 01/04/14**	**jue 01/05/14**
2	Estudi API Bluetooth	3 días	mar 01/04/14	jue 03/04/14
3	Disseny aplicació "echo"	3 días	vie 04/04/14	mar 08/04/14
4	Programació servidor	14 días	mié 09/04/14	lun 28/04/14
5	Programació client	11 días	jue 17/04/14	jue 01/05/14
6	Depuració i simulació	10 días	vie 18/04/14	jue 01/05/14
7	Entrega PAC 3	0 días	jue 01/05/14	jue 01/05/14
8	**Integració AndEngine**	**32 días**	**vie 02/05/14**	**dom 15/06/14**
9	Estudi motor de joc	32 días	vie 02/05/14	dom 15/06/14
10	Disseny UI	5 días	mar 06/05/14	sáb 10/05/14
11	Disseny lògica	1 día	lun 12/05/14	lun 12/05/14
12	Implem. menú inicial	4 días	mar 13/05/14	vie 16/05/14
13	Implem. tauler de joc	2 días	sáb 17/05/14	lun 19/05/14
14	Entrega PAC 4	0 días	lun 26/05/14	lun 26/05/14
15	Integració comunicació i interfície gràfica	20 días	mar 20/05/14	dom 15/06/14
16	Creació servidor de joc	14 días	mar 20/05/14	vie 06/06/14
17	Creació client	10 días	vie 30/05/14	jue 12/06/14
18	Depuració codi	15 días	mar 27/05/14	dom 15/06/14
19	Aplicació finalitzada	0 días	dom 15/06/14	dom 15/06/14
20	**Test i distribució**	**3 días**	**lun 16/06/14**	**mié 18/06/14**
21	Test de l'app funcional	3 días	lun 16/06/14	mié 18/06/14
22	Empaquetament	1 día	mié 18/06/14	mié 18/06/14
23	**Documentació**	**20 días**	**mar 27/05/14**	**dom 22/06/14**
24	Memòria	19 días	mar 27/05/14	vie 20/06/14
25	Enregistrament vídeo	1 día	jue 19/06/14	jue 19/06/14
26	Presentació	3 días	jue 19/06/14	dom 22/06/14
27	ENTREGA PROJECTE (PAC 5)	0 días	dom 22/06/14	dom 22/06/14

Figura 2. Diagrama de Gannt actualitzat

5.3 Canvis en la planificació inicial

La modificació més important de la planificació inicial ha estat el temps dedicat al desenvolupament i integració dels components de l'aplicació amb el motor de joc. Inicialment estava prevista una dedicació de 21 dies per aquesta tasca, mentre que la dedicació real final ha estat de 32 dies (gairebé un 50% més del previst). Aquest allargament ha provocat que les tasques de test i empaquetament no hagin pogut ser tan exhaustives com s'havia previst, passant d'una dedicació inicial de 6 dies a només 3. A més, s'ha hagut de sol·licitar una pròrroga de 6 dies respecte al termini previst d'entrega. Tot i així, s'han acomplert els objectius del projecte satisfactòriament.

6. Pressupost

El total del pressupost del projecte resideix en el cost de l'equip humà de desenvolupament. A continuació es desglossa el cost total del projecte en funció dels principals blocs de desenvolupament.

Concepte	Quantitat	Preu unitat	Total
Hores desenvolupament			
Disseny interfície gràfica	10	10€	100€
Disseny lògica del joc	2	10€	20€
Programació activitats del joc	18	20€	360€
Programació client - servidor	50	20€	1.000€
Integració UI - comunicació	40	20€	800€
Test	6	10€	60€
Empaquetament i distribució	1	5€	5€
TOTAL			**2.345€**

Taula 2. Pressupost del projecte

7. Estructura de la resta del document

El Capítol 2 descriu la fase d'anàlisi del projecte, i inclou l'estudi de l'estat de l'art del producte a desenvolupar, una anàlisi del mercat actual i un estudi de la tipologia d'usuaris que faran ús de l'aplicació.

El Capítol 3 es centra en l'etapa de disseny del projecte, explicant l'arquitectura de dades de l'aplicació i els detalls d'implementació.

El Capítol 4 explica els recursos necessaris per a executar l'aplicació, així com el procés d'instal·lació de la mateixa.

El Capítol 5 inclou una demostració en vídeo del funcionament de l'aplicació.

El Capítol 6 recull les conclusions i les línies de futur del projecte.

Capítol 2: Anàlisi

1. Estat de l'art

En aquest apartat es fa un repàs de les diferents solucions adoptades per diferents institucions (individuals, col·lectives, empresarials...) als múltiples problemes i reptes que presenta el projecte, com ara la creació d'un joc, la implementació de característiques multi-jugador, o la connexió que s'estableix entre els diferents usuaris per tal de poder intercanviar dades a temps real. Per aquest motiu, l'apartat es divideix en quatre seccions: en les tres primeres s'expliquen els avenços i situació actual dels tres problemes descrits. En la quarta secció s'exposen els tres problemes anteriors de manera conjunta, i s'estudien les aplicacions ja existents que compleixen els tres requisits.

Creació d'un joc:

En el desenvolupament d'aplicacions mòbils, en el sentit ampli del terme, hi intervenen una sèrie de requisits tècnics i humans necessaris per a implementar el producte en si, com poden ser el coneixement del llenguatge natiu de la plataforma per part del desenvolupador, o la disponibilitat de simuladors físics per a provar la usabilitat de l'aplicació. Generalment, les aplicacions basades en transferència d'informació o aquelles basades en el hardware del dispositiu (tal i com es van descriure a la PAC anterior) presenten una interfície d'usuari relativament senzilla (llistes, botons, mapes...) que es pot implementar fàcilment amb l'ajuda de les múltiples APIs que proveeixen les diferents plataformes. A continuació es llisten les interfícies específiques d'Android per a la implementació dels tres elements esmentats a tall d'exemple:

- Llista: la classe ListView (Google - Android Developer, 2014) proveeix una estructura (layout) per a crear llistes navegables.

- Botó: la classe Button (Google - Android Developer, 2014) serveix per a instanciar botons. Mitjançant un receptor d'esdeveniments, el programador pot associar accions al fet de prémer el botó.

- Mapa: mitjançant la classe MapView (Google - Android Developer, 2014) es poden afegir mapes a l'aplicació, basats en les dades de Google Maps.

D'igual manera, per accedir als components hardware de l'aplicació, Android proveeix múltiples APIs (localització, acceleròmetre, càmera, etc.).

Quan es tracta de desenvolupar un joc, però, els requeriments seran més complexos, ja que generalment s'impliquen les següents qüestions:

- Lògica: la lògica d'un joc acostuma a ser més complexa que la d'una aplicació genèrica, ja que s'han de contemplar totes les regles del joc i possibles accions del jugador.

- Física: la simulació d'aspectes físics com ara la gravetat o el rebot és un apartat fonamental en la majoria de jocs, sobretot en aquells basats en plataformes.

- Gràfics: la interfície d'usuari d'un joc sol ser molt complexa i difícil d'implementar amb els elements estàndard definits abans, per la qual cosa requereix una programació de baix nivell.

- Animació: en la majoria dels jocs, els gràfics no són estàtics, per la qual cosa s'han d'implementar uns algoritmes d'animació que sovint inclouen una component matemàtica important.

- Àudio i banda sonora: normalment els jocs van acompanyats d'una banda sonora que els fa més amens i recognoscibles.

Per aquests motius, per a la implementació d'un joc normalment s'empra un motor de joc (*game engine* en anglès), component que facilita el desenvolupament de tasques rutinàries d'una aplicació d'aquesta tipologia mitjançant l'abstracció de les mateixes dins d'una API.
Els motors de joc van començar a implementar-se com a components independents a mitjans dels anys 90, ja que el cost de fabricació "in-house" per part de cada empresa de desenvolupament havia crescut considerablement. A partir d'aquest moment van sortir al mercat nombroses implementacions, tant de codi obert com amb llicència. D'acord amb Game Career Guide (UBM LLC, 2014), podem diferenciar entre tres tipus de motor segons el seu nivell d'abstracció:

- **Baix nivell**: emprats per les companyies que prefereixen construir-se un motor de joc "a mida". Exemples d'aquesta tipologia serien les llibreries OpenGL, DirectX (implementació gràfica), Havok (simulació física)...

- **Nivell intermedi**: són motors que ja integren les múltiples funcions esmentades anteriorment, però que requereixen part de programació per a poder-les integrar correctament dins del joc. Alguns exemples són Genesis3D, Torque o AndEngine.

- **Alt nivell**: també anomenats "point-and-click", aquests motors inclouen un entorn de desenvolupament complet mitjançant una interfície d'usuari intuïtiva, de manera que es pot crear el joc sense gairebé escriure una línia de codi. Com a part negativa, és el tipus de motor que presenta més limitacions a l'hora de desenvolupar. Exemples: GameMaker, Unity3D.

D'acord amb la naturalesa del joc i de la plataforma de desenvolupament, un dels motors que més s'adapta a les característiques necessàries és AndEngine (Schroeder & Broyles, 2013) (Brownlee, 2013). Es tracta d'un motor de joc de codi obert 2D per Android basat en OpenGL, escrit en Java. Actualment és un dels motors més populars dins de la categoria, i consta d'una bibliografia completa i actualitzada.

Multi-jugador:

Les característiques d'un joc multi-jugador difereixen lleugerament de les aplicacions concebudes per l'ús d'un sol usuari simultàniament. En termes generals, es pot diferenciar entre dues arquitectures multi-jugador:

- Multi-jugador sobre un sol dispositiu: el joc està implementat perquè dos o més jugadors interactuïn a sobre del mateix dispositiu físic. Aquesta arquitectura implica dos requisits: suport d'una interfície multi-tàctil per part del dispositiu, i una superfície d'interacció (pantalla) prou gran per a poder interactuar amb comoditat dos jugadors com a mínim. Referent a la primera qüestió, el sistema operatiu Android suporta els gestos multi-tàctils (Google - Android Developer, 2014), que són aquelles interaccions amb la pantalla on hi estan involucrats més d'un dit (o punters, d'acord amb la terminologia de la plataforma). Respecte al segon requisit, actualment existeixen dispositius amb pantalles prou grans per a permetre la interacció còmoda de dos o més usuaris a la vegada.

- Multi-jugador en múltiples dispositius: amb aquesta arquitectura, els diferents jugadors empren dispositius diferents, però tots comparteixen la mateixa sessió del joc.

Aquesta última arquitectura és la que presenta un repte més gran a l'hora d'implementar-la, ja que requereix crear la sessió multi-jugador, buscar i gestionar els múltiples candidats a jugar, connectar els jugadors i intercanviar les dades de joc.
En funció de l'interval de refresc de les dades, es pot diferenciar entre dues subcategories (Google - Android Developer, 2014):

- Multi-jugador basat en torns: les restriccions temporals són més laxes, ja que la velocitat en la transmissió de la informació no és un factor crític.
- Multi-jugador a temps real: els missatges d'intercanvi d'informació han d'arribar al receptor en el menor temps possible per assegurar la fluïdesa i el bon funcionament del joc.

Google ofereix una API anomenada GoogleApiClient (Google - Android Developer, 2014) que proveeix una interfície per a facilitar les tasques de connexió als serveis de Google Play i poder d'aquesta manera iniciar una sessió de joc i convidar o connectar-se amb altres usuaris de la plataforma.

Per al projecte a desenvolupar, però, la connexió s'establirà mitjançant una xarxa ad-hoc creada pels mateixos jugadors, sense necessitat de connectar-se a un servei extern com el proveït per Google. Al següent apartat es discuteix el tema de la connexió i intercanvi de dades.

Connexió:

Com ja s'ha pogut veure al llarg de l'apartat anterior, la connexió entre els diferents dispositius implicats en el joc és un punt fonamental en qualsevol aplicació multi-jugador. El present projecte es basa en una connexió P2P (Buford, Yu, & Lua, 2009) (*Peer-to-Peer* en anglès) entre els diferents jugadors, una tipologia de xarxa on tots els elements participants tenen el mateix rol, i consumeixen dades i/o recursos alhora que els proveeixen als altres participants. Aquest fet permet simplificar l'arquitectura d'intercanvi de dades emprada, ja que es pot prescindir d'un servidor que gestioni les diferents connexions i informació enviada.

Amb aquest propòsit, Android ofereix dues interfícies de comunicació P2P per facilitar l'intercanvi d'informació entre dispositius:

- Bluetooth (Google - Android Developer, 2014): aquesta interfície permet la connexió i intercanvi de dades entre múltiples dispositius a través de la funcionalitat Bluetooth del dispositiu. Entre altres funcionalitats, la API suporta la cerca i enllaç amb altres dispositius Bluetooth, l'establiment de canals de comunicació RFCOMM i la transmissió i recepció de dades.
- WiFi P2P (Google - Android Developer, 2014): aquesta API, a través de la interfície WiFi del dispositiu, permet la cerca, connexió i transmissió de dades entre dispositius mitjançant el protocol de comunicació IEEE 802.11.

Les xarxes de comunicació P2P entre dispositius mòbils i dispositius electrònics en general és un tema molt actiu actualment, ja que és un camp amb moltes aplicacions, i a més implica una estructura d'implementació senzilla per part de l'usuari final. Algunes d'aquestes aplicacions són:

- Dispositius de salut: actualment Android proveeix una API (Google - Android Developer, 2014) per a interaccionar amb dispositius mèdics (monitors cardíacs, monitors de glucosa en sang, termòmetres...).

- Auriculars: moltes marques del sector electroacústic ofereixen auriculars sense fils que implementen connexió Bluetooth.
- Dispositius de navegació: algunes marques comercials ofereixen dispositius GPS sincronitzables amb els dispositius mòbils mitjançant Bluetooth.
- Jocs multi-jugador.

El desenvolupament de jocs és un camp molt actiu també, i la implementació de característiques multi-jugador no es queda endarrere. Per exemple, l'empresa DB-Best Technologies explica en un post de novembre de 2013 (DB-Best Tech, 2013) les dificultats per a poder connectar simultàniament en una xarxa P2P dispositius Android i iOS. Davant de la impossibilitat de fer-ho amb les interfícies estàndard que proveeixen els SO, van haver d'implementar un protocol propi per a poder permetre la connexió Bluetooth entre dispositius d'aquests dos sistemes operatius.
Per altra part, en el repositori de codi Google Code podem trobar diferents APIs creades per la comunitat per a facilitar la comunicació P2P entre dispositius Android. Concretament, "P2P Communication Framework for Android" (Google, 2014) ofereix funcionalitats d'auto-connexió amb xarxes Bluetooth properes, control de flux de la comunicació, etc.

Per tant, es pot veure com l'àmbit de les comunicacions Peer-to-Peer és un camp actiu actualment, amb constants aportacions i innovacions tant de la comunitat de desenvolupadors com per part d'empreses.

Mercat actual d'aplicacions multi-jugador en xarxa P2P:

Actualment es poden trobar en el mercat d'aplicacions de Google molts jocs que implementen funcionalitats de multi-jugador mitjançant una xarxa P2P. A continuació se'n llisten una selecció dels més populars amb les característiques bàsiques de cada un.

- UNO (Google, 2014): es tracta del famós joc de cartes de Mattel. Implementa una modalitat multi-jugador online (a través d'una connexió de dades) i una modalitat local, permetent emprar les interfícies WiFi o Bluetooth per a l'establiment de la xarxa P2P. Aquesta aplicació es classifica dins de la tipologia de multi-jugador per torns.
- Belote (Google, 2014): joc de cartes que suporta la característica multi-jugador en línia o en local, mitjançant la interfície Bluetooth. També, igual que el cas anterior, es tracta d'una aplicació per torns.
- Minecraft (Google, 2014): joc de plataformes que permet implementar la característica multi-jugador mitjançant una xarxa WiFi local.
- MiniSquadron! (Google, 2014): arcade multi-jugador local (WiFi P2P) i en línia.

En resum, es pot observar que en les aplicacions multi-jugador local s'usen indistintament les interfícies Bluetooth o WiFi per establir una comunicació de xarxa. Les característiques d'ambdues plataformes són molt similars, així com les interfícies d'implementació que ofereix Android.

2. Anàlisi del mercat

Actualment si hi ha una cosa que estigui present en la vida quotidiana de qualsevol persona, independentment de l'edat, la cultura i fins i tot del poder adquisitiu, són els dispositius mòbils.
Gràcies a les noves tecnologies de telecomunicació, s'ha aconseguit el fenomen de la ubiqüitat, de forma que l'usuari pot estar ininterrompudament connectat a la xarxa independentment de la seva ubicació i sense necessitat de cap infraestructura física. Aquest segurament és el motiu pel qual els usuaris actualment passin més temps davant dels dispositius mòbils que davant l'ordinador, i més d'un 30% del temps dedicat al mòbil és per jugar (SuperMonitoring, 2013). Prova d'aquesta demanda d'oci és el creixement exponencial que han experimentat les diferents tendes d'aplicacions o App Stores dels principals proveïdors. En el cas d'Android, en els últims 18 mesos s'han duplicat el nombre d'aplicacions disponibles a Google Play havent-n'hi actualment més de 1.250.000. Segons un estudi encarregat per la Comissió Europea (Mulligan & Card, 2014), l'any 2013 compradors i anunciants de la Unió Europea es gastaren 6.100 milions d'euros en aplicacions, el 30% de la despesa mundial en apps.

2.1 Estudi de mercat

El joc implementat emula l'esport del Hoquei Taula en el qual competeixen dues persones que intenten anotar punts a la porteria contrària.
El tret diferencial d'aquesta aplicació respecte a la majoria de les ja existents a la plataforma Google Play (Google, 2014) és la capacitat de poder jugar en mode multi-jugador des de dos dispositius diferents. D'acord amb les característiques de les aplicacions semblants, aquestes permeten la interacció entre dos jugadors, però actuant sobre un sol dispositiu. Per tant aquest nou enfocament suposa una innovació en aquest àmbit.
És un joc que pot interessar majoritàriament al segment de població jove, primer perquè són els principals clients de les boleres i sales de màquines recreatives i per tant coneixen i possiblement han practicat alguna vegada aquest esport a la vida real, i segon perquè és un segment de població que compleix dos requisits necessaris: la gran majoria disposa de *smartphone*, i a més habitualment es passen molt temps reunits amb grups d'amics, amb qui poden disputar una partida.

2.2 Comparativa de la competència

Actualment podem trobar en el mercat d'aplicacions de Google jocs que emulen l'esport del hoquei taula, i per altra part, jocs que implementen funcionalitats de multi-jugador mitjançant una xarxa P2P. El tret diferencial de l'aplicació fruit d'aquest projecte envers la majoria de les ja existents a la plataforma Google Play (Google, 2014), és la capacitat de poder jugar a hoquei taula en mode multi-jugador des de dos dispositius diferents, compartint la mateixa sessió del joc.
La Taula 3 mostra una comparació d'algunes de les aplicacions disponibles a Google Play més populars (algunes citades a l'apartat 1. Estat de l'Art), amb les característiques bàsiques de cada una.

Nom	Característiques	Multi-jug.	Connexió	Tipologia	Logotip
UNO	Joc de cartes de Mattel	Sí	WiFi Bluetooth	Per torns	
Belote	Joc de cartes	Sí	Bluetooth	Per torns	
Minecraft	Joc de plataformes. Creació d'un mon virtual.	Sí	WiFi	Més d'un jugador alhora	
MiniSquadron	Arcade. Joc on pilotes un avió i has d'anar abatent els avions enemics.	Sí	WiFi	Més d'un jugador alhora	
Air Hockey Deluxe	Joc emulador de l'esport de Hoquei Taula	Sí	No	Dos jugadors en el mateix dispositiu	
Air Hockey Ultimate	Joc emulador de l'esport de Hoquei Taula	No	No	-	

Taula 3. Comparativa de diferents aplicacions existents en el mercat

2.3 Estratègia de màrqueting

El que ens caldrà primer és crear una marca, tant a nivell de desenvolupador com a nivell d'aplicació. Una de les primeres passes per aconseguir-ho és dissenyar una imatge corporativa i una imatge que identifiqui el producte que volem promocionar, en aquest cas el joc.

Un cop que tinguem definit el nom comunicatiu del producte caldrà iniciar la promoció de l'app, que tindrà com a objectiu maximitzar-ne la visibilitat, per tal de donar a conèixer la marca i atreure potencials usuaris. Fent referència novament al tipus d'aplicació, al tractar-se d'un joc, és de cabdal importància arribar completament (i per diverses vies) al públic potencial, ja que cada descàrrega aporta el seu granet d'arena en l'èxit del projecte. Diferent seria el cas d'una aplicació creada a mida per una empresa, on els usuaris finals serien el personal, amb un perfil molt concret i una predisposició molt diferent a la d'un usuari desconegut que es descarrega una aplicació.

Amb aquest exemple es vol fer èmfasi en la importància de la projecció de l'aplicació a sobre del nostre *target*, ja que necessitem maximitzar la "necessitat" de l'usuari per descarregar-se-la i la "fidelitat" del mateix un cop instal·lada.

L'estratègia promocional consistirà en potenciar la presència de la marca a la xarxa, ja que és un dels aparadors més efectius actualment i de més baix cost.

Pàgina web:

Actualment, la presència a internet és una obligació. Qualsevol usuari que busqui el nom de l'aplicació o de la marca a la xarxa ha de rebre, entre els primers resultats, la nostra pàgina web. Aquesta ha d'incloure les següents característiques (WIP Connector Ltd, 2013):

- Ha d'explicar clarament quin o quins productes oferim i quina funcionalitat tenen. En el nostre cas es tractaria d'explicar el funcionament i les instruccions del joc, mitjançant elements entenedors (exemples, gràfiques de flux, captures de pantalla, gràfiques d'interacció...).

- La inclusió d'una pàgina de contacte amb l'equip és fonamental per a gestionar possibles consultes sobre el funcionament o sol•licituds d'altres empreses per a publicitar-se o promocionar la nostra app.

- S'ha d'incloure un petit text concís que descrigui el producte i "enganxi" als potencials usuaris.

- Optimització dels motors de cerca (SEO), per a aparèixer en les primeres posicions quan un usuari busqui el nostre nom o una referència a la nostra aplicació.

- Cal incloure l'enllaç de l'aplicació a l'App Store, ja que d'aquesta manera es facilita als usuaris l'accés al portal de l'app, podent descarregar-la, revisar els comentaris fets per altres usuaris...

Xarxes socials:

L'aparició de referències a l'aplicació a les xarxes socials és una potencial eina de màrqueting basada en el boca a boca electrònic (eWOM), considerat una de les estratègies de màrqueting que genera més confiança al receptor, ja que generalment la font és una persona coneguda. Accions a desenvolupar:

- Creació d'una pàgina de l'aplicació en les principals xarxes socials (Facebook, Twitter...). La compartició de la pàgina entre els contactes de l'empresa és el primer pas per a difondre el coneixement de la marca.

- Promoure el "networking" amb altres empreses del sector, per tal de donar-nos a conèixer i actuar mútuament com a element promocional.

- Recompensar els usuaris de l'aplicació amb béns virtuals a canvi de promocionar l'aplicació al seu perfil social.

- Incloure (com en el cas del portal web) enllaços que dirigeixin l'usuari a l'App Store per a poder-se descarregar l'aplicació.

- Actualitzar freqüentment el mur amb les notícies i novetats relacionades amb l'aplicació i l'empresa. Un perfil social sense activitat, amb entrades molt antigues, mina la credibilitat de l'usuari en la marca (WIP Connector Ltd, 2013).

- Respondre els comentaris que puguin fer els usuaris. D'aquesta manera es crea una atmosfera propera entre l'equip de desenvolupament i l'usuari, fet que aquest últim valora molt positivament.

Pel que fa a anuncis publicitaris ho centrarem en la col·laboració amb altres empreses del sector ja que és una manera d'obtenir publicitat "gratuïta", acordant intercanvis d'anuncis. Aquesta estratègia de màrqueting és important, ja que ens permet ampliar el nostre *target* amb altres usuaris del mateix sector: consumidors d'aplicacions mòbils. Aquest factor implica que gairebé la totalitat d'ells també seran potencials consumidors de la nostra aplicació. Accions a realitzar:

- Menció del soci col•laborador a la nostra pàgina web i mur social, i viceversa. D'aquesta manera ampliem la projecció del nostre producte a través dels usuaris de l'altre.

- "Intercanvi d'usuaris": s'aprofiten les llistes de mailing de cada empresa per a enviar informació o publicitat relacionada amb l'aplicació de l'altre soci.

- Intercanvi d'espais publicitaris: oferim de forma gratuïta una franja publicitària de la nostra aplicació a canvi d'aparèixer als anuncis de la seva app.

L'element aglutinador de tots aquestes accions és la definició del producte: en tot moment s'ha de tenir clar (i s'ha d'expressar clarament) quina és la finalitat de l'aplicació i què oferim als usuaris, ja que d'aquesta manera obtenim una visió única del producte, fet que permet focalitzar i optimitzar els esforços de promoció de l'aplicació.

Recuperació de la inversió:

El model de monetització que farem servir per l'aplicació és l'anomenat "Free to Play". El fonament d'aquesta estratègia és la gratuïtat de l'aplicació, és a dir, l'usuari se la pot descarregar, instal·lar i jugar de manera gratuïta. Cal destacar el fet de no carregar cap import per la descàrrega de l'app, ja que d'aquesta manera l'usuari pot provar-la fàcilment (sense la necessitat d'efectuar un pagament) i decidir si li agrada o no. El fet d'imposar un preu de descàrrega a la nostra aplicació provocarà molt probablement que l'usuari no assumeixi el risc d'efectuar el pagament per un producte desconegut, fet que repercuteix directament en el nombre de descàrregues.
En aquest cas la monetització es basa en el fet de que l'usuari pot desbloquejar opcions avançades del joc, com pot ser per exemple la personalització de les paletes i terreny de joc (inclusió de fotos personals, dissenys avançats...), noves modalitats de joc (afegir obstacles, noves regles...), etcètera.

L'altre pilar sobre el qual es fonamenta el retorn de la inversió de l'aplicació són els anuncis. Els beneficis que reporta aquesta tècnica són proporcionals al nombre de descàrregues de l'aplicació. Més descàrregues significa més tràfic, fet que augmenta el nombre de visualitzacions i clics a sobre dels anuncis. A més, el mercat espanyol és el segon classificat en el rànquing de CTR o *Click-Through Rate* (WIP Connector Ltd, 2013). Aaquest índex indica el nombre de clics que es fan sobre un anunci entre el total de visualitzacions del mateix. A Espanya aquest valor és del 3.4%, fet que significa que per cada 100 visualitzacions d'un anunci, una mica més de tres usuaris cliquen a sobre per accedir al contingut de l'anunci. Aquesta situació és positiva, ja que les companyies poden arribar a pagar un preu més elevat per cada clic.
A més, el fenomen de la ubiqüitat dels *smartphones* es pot utilitzar com un valor afegit en relació als anuncis. Actualment els usuaris porten el seu telèfon intel·ligent a tot arreu. Per tant, si els anuncis que ofereix el nostre proveïdor estan basats en la localització i en els gustos i costums de l'usuari, la nostra aplicació té un altre valor afegit per a l'usuari.

D'acord amb les guies que es suggereixen al document citat (WIP Connector Ltd, 2013), la gestió dels anuncis ha de seguir aquestes directrius:

- S'ha de trobar l'equilibri perfecte entre un anunci prou visible però no intrusiu. L'excés d'anuncis interrompent el ritme del joc molesta al jugador i per tant empitjora l'experiència d'usuari, fet que pot provocar l'abandó de l'aplicació per part de l'usuari. Per altra part, un anunci poc visible generarà poques visualitzacions i menys clics, fet que repercuteix directament en els ingressos de l'aplicació. Per tant s'ha de trobar l'equilibri òptim entre les variables de localització i visibilitat, per tal de maximitzar el nombre de clics.

- Un altre punt relacionat amb l'anterior és el del típic botó "Omet l'anunci". Tal i com es recomana a la bibliografia (WIP Connector Ltd, 2013), aquest botó s'ha de retardar uns segons, amb l'objectiu que l'usuari tingui temps de jutjar si li interessa l'anunci o no abans de

prémer el botó d'omissió. Segons s'explica, si el botó "Skip" és present des del principi, l'usuari té tendència a prémer-lo sense esperar a visualitzar-ne el contingut. Retardant-ho, el jugador pot valorar si l'anunci és del seu interès. En cas afirmatiu, repercuteix en el nombre de clics, i per tant en el benefici de l'aplicació.

- L'ús d'anuncis visualment rics (anomenats Rich Media) són molt més atractius per als usuaris que no pas les imatges estàtiques. Per tant es valora la innovació, amb vídeos atractius i ús de les noves tecnologies audiovisuals (3D...).

3. Públic objectiu i perfils d'usuari

Els dispositius mòbils, especialment els telèfons intel·ligents o *smartphones*, s'han popularitzat tant que ja han esdevingut un objecte quotidià a les mans de tot tipus d'usuaris, independentment del sexe, edat, cultura o nivell econòmic. L'usen els infants i els avis, els alts executius i els que cerquen feina, els que caminen per Wall Street i els que ho fan per un poblat indígena de Kenia, i qualsevol d'aquestes persones pot quedar enganxat a un joc tan senzill com és aquest. Però malgrat tots ells en siguin usuaris potencials, crec que el públic objectiu a qui hem de dirigir el producte és aquell format per la població de menor edat, ja que són els que més ganes tenen de provar productes i experiències noves, i són els que conviuen i juguen més amb amistats del mateix rang d'edat.

3.1 Perfils d'usuari

Nom	Miquel	**Tipus**	Focal	**Persona 1**
Sexe	Masculí			
Edat	15			
Professió	Estudiant			
Descripció de la persona	En Miquel és un jove estudiant centrat primordialment en la tasca d'estudiar. Durant el temps de pati li agrada compartir experiències amb els seus companys de classe i els caps de setmana i durant l'estiu també ho fa amb els companys de colla. Tots disposen d'*smartphone* i comparteixen grups de Whatsapp.			
Descripció de l'escenari	És l'hora del pati i en Miquel està amb els amics. Coneix el joc de hoquei taula que li va mostrar un d'ells que ara li proposa fer una partida. Saben que tenen temps, una partida i prou, no els cal disposar de WiFi, es connecten per Bluetooth i es posen a jugar. Sona el timbre i ja han de tornar a classe. En Miquel i el seu company ja han acabat la partida i parlen de fer una lliga amb els altres companys. Sembla que n'hi ha d'interessats.			

Nom	Biel	**Tipus**	Focal	**Persona 2**
Sexe	Masculí			
Edat	11			
Professió	Estudiant			
Descripció de la persona	En Biel és un jovenet que disposa d'una tauleta que empra únicament per jugar. Vol provar tots els jocs que tenen els seus cosins, amb els que juga tots els caps de setmana. Els jocs els hi descarrega son pare mentre en Biel es compromet a jugar-hi només els caps de setmana en haver acabat els deures. La realitat és que entre setmana també hi juga qualque estona amb la seva germana, que és un parell d'anys més gran, i ja té un telèfon intel·ligent.			
Descripció de l'escenari	En Biel va conèixer el joc de hoquei a través dels cosins i ho explica a son pare a qui li sembla adequat i li descarrega a ell i a la seva germana. Avui els ha trobat fent una partida abans d'anar a dormir i després d'advertir-los que han incomplert els pactes consent passar-ho per alt si fan una partida amb ell. Sap que és una partida i prou, un temps limitat i assumible, no els provocarà cap frustració de no poder-se passar un nivell o una pantalla, no és un joc d'aquest tipus. El pare perd i els proposa fer la revenja el cap de setmana.			

Nom	Rafel	**Tipus**	Focal	**Persona 3**
Sexe	Masculí			
Edat	25			
Professió	Esportista professional			
Descripció de la persona	En Rafel és un esportista que es veu obligat a viatjar juntament amb els seus companys d'equip tots els caps de setmana. Es passen moltes estones esperant als aeroports o viatjant amb autobús. Li agrada escoltar música i ho fa emprant el seu *smartphone* de darrera generació. Viu amb els seus pares i encara no té despeses d'hipoteca. Li agrada gastar els doblers que guanya amb l'esport amb roba i tecnologia. No entra a valorar si una app es gratuïta o de pagament a l'hora de descarregar-se-la al seu dispositiu.			
Descripció de l'escenari	En Rafel viatja amb autobús amb l'equip, encara queda una bona estona per arribar al lloc on han de disputar el partit. Al seu costat hi seu un dels companys amb qui sol compartir les novetats tecnològiques. Li proposa fer una partida de hoquei taula i connecten els seus dispositius per Bluetooth. Tindran temps de fer unes quantes partides i inicien un torneig a un nombre de partides limitat. És una de les opcions avançades del joc. Qui perdi el torneig pagarà el berenar i les begudes pel camí de tornada.			

Nom	Imma	**Tipus**	Secundari	**Persona 4**
Sexe	Femení			
Edat	20			
Professió	No definida			
Descripció de la persona	L'Imma en acabar l'ESO va optar per la formació professional, que no li va acabar d'agradar. Va aconseguir una feina de temporada en el camp de l'hostaleria i el fet de disposar d'aquells ingressos amb les despeses bàsiques cobertes (encara viu amb els pares), la va satisfer i va pensar que deixar els estudis havia estat un encert. Per ella l'*smartphone* és una eina primordial. Per les tardes al bar juga amb els amics que es troben com ella, ara que ha acabat la temporada turística.			
Descripció de l'escenari	L'Imma ha dedicat el matí a trescar per internet, primer cercant feina, però després ha desviat l'atenció cap als aparadors virtuals de jaquetes, ja que arribarà el fred i es trobarà només amb la de la temporada passada. A les xarxes socials l'han convidada a descarregar-se algunes app i no és la primera vegada que ha caigut amb la temptació. Ara al bar mostra als amics la darrera que ha descobert i això els indueix a pensar amb la lliga de hoquei taula que tenen en marxa, així que decideixen fer-ne unes partides.			

Nom	Toni	**Tipus**	Secundari	**Persona 5**
Sexe	Masculí			
Edat	44			
Professió	Administratiu			
Descripció de la persona	En Toni procura estar al corrent de les novetats que sorgeixen en noves tecnologies, primer perquè és un tema recurrent al despatx on treballa, i després perquè té dos fills amb edat de deixar-lo amb evidència en converses sobre aquest camp. No dubta en interessar-se per les aplicacions que li mostren els companys de feina i també intenta compartir les que es descarreguen els seus fills, ara que encara estan disposats a fer-ho.			
Descripció de l'escenari	És un dia feiner i en Toni està cansat en arribar el vespre. Les notes d'ambdós fills han estat bones i els vol demostrar la satisfacció fent una proposta que sap que els agradarà. Els proposa fer unes partides de hoquei taula abans de sopar i el que perdi arraconarà la taula. Els fills accepten encantats perquè confien guanyar i alliberar-se d'aquesta feina. Son pare també n'és conscient, però acceptarà el resultat de bon grat.			

Capítol 3: Disseny

1. Arquitectura general de l'aplicació

1.1 Model de distribució i comunicació

L'aplicació dissenyada segueix un model client-servidor, una estructura d'aplicació distribuïda que diferencia els nodes de procés de la informació (servidors) i els nodes sol·licitants d'informació. En el cas concret d'aquest projecte, el servidor és l'encarregat d'iniciar la connexió amb els clients, monitoritzar la posició i estat dels elements gràfics de l'aplicació (pilota, paletes, marcadors...), i de computar la lògica del joc (col·lisions, recompte de gols...). Els clients simplement s'encarreguen d'enviar la posició de la seva paleta al servidor, perquè aquest pugui processar-la i enviar mitjançant un missatge *broadcast* la posició de la mateixa perquè la resta de clients la pugui renderitzar al lloc corresponent a temps real.

Tot i tractar-se d'un model distribuït, la mateixa aplicació serveix per implementar tant la part de client com de servidor. D'aquesta manera dos dispositius són suficients per a permetre jugar a dos usuaris, ja que un dels dos fa alhora de client i servidor. En aquest cas, el client es connecta al servidor (corrent en un altre fil d'execució o *thread* del mateix dispositiu) mitjançant l'adreça IP de *loopback* (127.0.0.1), i totes les comunicacions es fan a través d'aquesta interfície. Contràriament, el dispositiu que actua solament com a client es connecta al dispositiu servidor a través del canal de radiocomunicació establert amb la interfície WiFi del dispositiu mòbil, mitjançant l'adreça IP introduïda per l'usuari a través de la interfície gràfica (veure Subsecció 1.3).

D'acord amb aquesta estructura, un requisit de la interfície de comunicació és la presència d'un canal de connexió amb el mateix dispositiu (adreça de *loopback*) per a la transferència de dades del client local amb el servidor. La interfície Bluetooth no disposa d'aquesta característica, i per tant per al projecte s'ha usat la interfície WiFi del dispositiu per a la implementació de les comunicacions, que sí ho suporta.

1.2 Arbre de navegació

L'arbre de navegació de la Figura 3 mostra la distribució de les pantalles de l'aplicació i el flux de navegació entre elles. Cada color correspon a una activitat diferent de l'aplicació. Una activitat (*Activity* en anglès) és aquella classe que mostra i/o permet a l'usuari realitzar una acció determinada, i generalment coincideix amb les diferents pantalles de l'aplicació. En aquest cas, com es pot veure a l'arbre, només s'han definit dues activitats, tot i que l'aplicació consta de fins a cinc pantalles diferents: s'han agrupat aquelles que tenen un contingut similar, i s'han definit com a escenes de la mateixa activitat. Una escena és un conjunt d'elements gràfics amb un propòsit concret: mostrar un menú, mostrar la pantalla d'inici del joc... Una activitat pot contenir diverses escenes. La justificació d'aquesta elecció és la següent: una activitat per sí sola té un cicle de vida que s'ha de complir: creació, càrrega de recursos, renderització, pausa, represa, alliberament dels recursos i destrucció. Aquests passos comporten una certa càrrega computacional, amb el conseqüent consum de temps (Gramlich,

AndEngine Forum, 2013). Per aquest motiu, les pantalles que fan servir recursos similars i que tenen unes funcionalitats semblants s'han agrupat dins de la mateixa activitat. D'aquesta manera, les funcions de creació, càrrega, etc. de la mateixa només s'han d'executar una vegada, i per a mostrar els diferents continguts s'empren diferents escenes.

AndEngine disposa del paquet "scene", que conté les classes SplashScene (per a crear una pantalla inicial animada), MenuScene (creació de menús) i Scene (classe genèrica per a la creació d'escenes).

Com es pot veure a l'arbre, la primera activitat (en verd) conté la pantalla inicial, el menú principal de l'aplicació i el menú d'opcions de l'aplicació. La pantalla inicial correspon a una escena de tipus SplashScene, on es mostra el títol de l'aplicació i el nom de l'autor de forma animada. Les dues escenes restants, menú principal i menú opcions, simplement estenen la classe MenuScene, que serveix per a fer escenes de menú amb elements clicables.

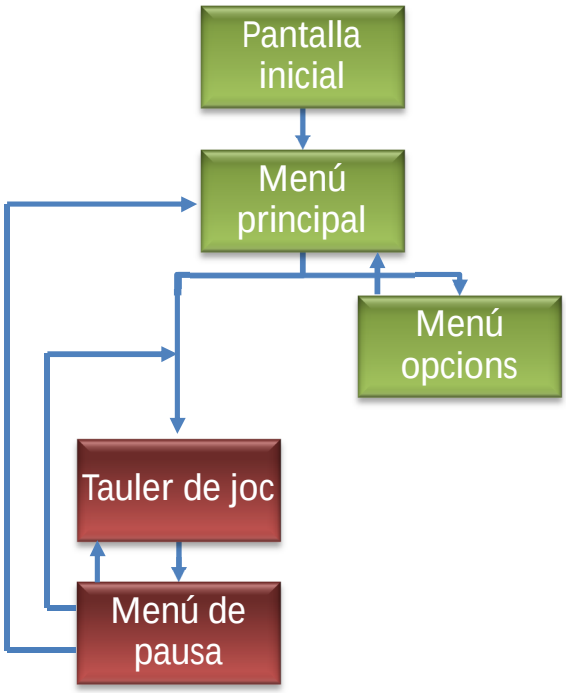


Figura 3. Diagrama de navegació de l'aplicació

La segona activitat (en vermell) conté l'escenari del joc: l'escena principal conté el propi tauler de joc on els jugadors interactuaran amb el dispositiu i els elements de joc, i una pantalla de pausa, on l'usuari pot triar entre reprendre el joc, iniciar una nova partida, o tornar al menú principal de l'aplicació.

1.3 Interfície gràfica

A continuació es mostren les diferents pantalles de l'aplicació. Degut a la complexitat que suposa el conjunt del projecte (implementació de la interfície de comunicació, programació de la lògica del joc en un entorn nou...), la interfície gràfica proposada és extremadament senzilla, ja que d'acord amb la planificació temporal del projecte no s'ha previst el disseny gràfic dels elements del joc (paleta, pilota, imatge de fons...). Per tant, la línia base de disseny de l'aplicació és la que es mostra a continuació. De cara a una possible ampliació futura del projecte es contempla el disseny específic dels elements gràfics del joc.

Primera activitat: MainActivity

Com es pot observar a la Figura 4, la interfície consta d'un fons uniforme obscur amb text de color vermell i font informal, per ressaltar els elements textuals. La imatge de l'esquerra representa la "Pantalla inicial", d'acord amb la denominació emprada a l'arbre de navegació. Simplement conté una escena de tipus SplashScene, on es mostra el títol del joc i el nom de l'autor. Aquesta escena disposa d'un temportitzador automàtic per passar a l'escena del menú principal després d'uns cinc segons.

Figura 4. Interfície primera activitat

En la imatge de la dreta es pot observar el menú principal. Conté quatre opcions disponibles: crear una nova partida, unir-se a una partida existent, mode jugador sol i opcions. Si es tria la primera opció, el dispositiu de l'usuari instancia un objecte servidor i un objecte client, i s'inicia la segona activitat (descrita en el següent punt). Si l'usuari en canvi prem el botó "unir-se a una partida existent", l'aplicació passa a actuar com a client. En el mode jugador sol, el funcionament és similar al de la

primera opció descrita, només que l'oponent sempre retorna la pilota. L'escena del menú d'opcions és molt similar a la del menú principal, amb els paràmetres editables també disposats en forma de llista.

Segona activitat: GameActivity

Aquesta activitat conté les diferents escenes del tauler de joc. És la que conté tota la lògica del joc, i la que fa un ús més intensiu de les característiques que proporciona el motor AndEngine. A més, ha d'interactuar amb la interfície de comunicació per tal de mostrar a temps real la posició de l'altre jugador i de la pilota, així com la gestió dels missatges de pausa dels clients. Per tant, tot i que la interfície gràfica és senzilla, la lògica subjacent és relativament complexa.

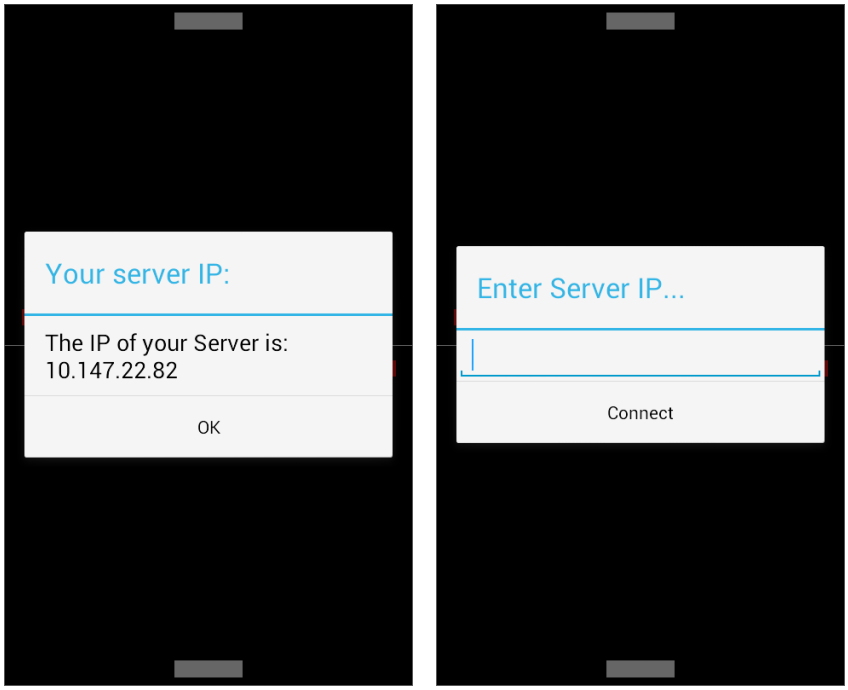


Figura 5. Paràmetres de connexió: servidor (esq.) i client (dreta)

La Figura 5 mostra l'aspecte de la pantalla un cop s'ha iniciat GameActivity (resultat de prémer una de les opcions del menú principal mostrat al punt anterior). Depenent de si l'aplicació actua com a servidor o com a client, aquesta canvia. Si el dispositiu allotja el servidor, apareix una finestra de diàleg informant de l'adreça IP del servidor, per tal que els demés jugadors la puguin conèixer per a posteriorment connectar-s'hi. Prement la tecla de confirmació simplement es tanca la finestra.
Si el dispositiu en canvi actua com a client, apareix una finestra amb un camp de text editable on l'usuari ha d'introduir l'adreça IP del servidor al qual es vol connectar. Un cop confirmada l'acció, el programa inicia un socket i intenta connectar-se a la IP introduïda. Si es correspon amb la d'un servidor del joc, aleshores s'estableix la connexió i la partida pot començar.

La Figura 6 es correspon amb les pantalles que es poden veure un cop la partida s'ha iniciat. A la imatge de l'esquerra es pot veure el terreny de joc amb les paletes. Al voltant de la línia divisòria del terreny de joc es poden veure els marcadors de cada jugador.
La captura de la dreta es correspon amb el menú de pausa, que apareix quan l'usuari prem el botó corresponent del tauler de joc. Durant aquest estat ni la pilota ni la paleta es mouen. El menú ofereix tres opcions: reiniciar la partida, navegar fins al menú principal de l'aplicació, o bé continuar amb la partida actual. A l'arbre de la Figura 1 es pot veure el flux de navegació de l'aplicació en funció de l'opció triada en el menú.

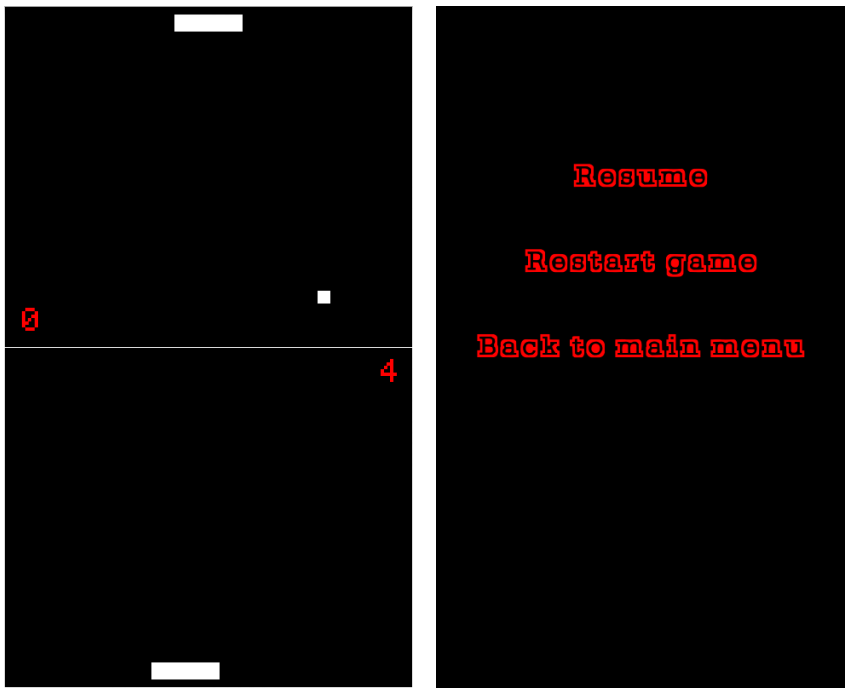


Figura 6. Pantalla del joc en acció (esq.) i menú de pausa (dreta)

1.4 Llibreries externes

Per al desenvolupament de l'aplicació s'ha fet servir el motor de joc AndEngine, com ja s'ha citat. La distribució de codi obert d'aquesta eina es presenta en forma de projecte Android, és a dir, un conjunt de classes Java ordenades en funció dels diferents paquets de funcionalitats inclosos, a més dels documents XML addicionals, com per exemple el Manifest Android del projecte.
A part del nucli de funcions que proveeix el motor de joc, també s'ha fet ús de les extensions Multiplayer Extension i PhysicsBox 2D Extension, necessàries per al desenvolupament de l'aplicació.
Per a integrar les funcionalitats del motor de joc a l'aplicació, el codi font del mateix (així com de les seves extensions) ha de ser inclòs com a llibreria del projecte, mitjançant les opcions que proveeix l'IDE Android Development Tools. D'aquesta manera es poden importar les diferents classes implementades pel motor de joc que s'hagin d'usar per al desenvolupament del projecte.

2. Descripció detallada dels components de l'aplicació

2.1 MainActivity

És l'activitat principal de l'aplicació. Està definida com a *launcher* en el Manifest d'Android, és a dir, és la classe que s'instancia quan arrenca l'aplicació, per la qual cosa és l'encarregada d'instanciar i configurar el motor de joc que s'utilitzarà al llarg del cicle de vida de l'aplicació.
Aquesta classe estén SimpleBaseGameActivity, una classe pròpia d'AndEngine que ofereix tots els mètodes necessaris per a configurar i obtenir una instància del motor de joc:

```
public class MainActivity extends SimpleBaseGameActivity
```

A continuació es comenten alguns dels mètodes abstractes de SimpleBaseGameActivity que s'implementen a MainActivity per tal de portar a terme la configuració del motor de joc:

- **onCreateEngineOptions**: en aquest mètode es defineix i s'instancia l'objecte Camera, que representa bàsicament el camp visual de l'usuari a sobre de l'espai infinit de joc. El constructor de la classe Camera té com a paràmetres les mides d'amplada i alçada que tindrà el camp visual.
  Un altre paràmetre important a definir dins d'aquesta funció és la política de resolució que ha de renderitzar el motor. Depenent de les mides de la pantalla del dispositiu on s'executi l'aplicació, el renderitzat s'adaptarà a la mida del display per tal de mantenir la proporció original de la càmera del joc.
  Finalment, dins d'aquesta funció també es defineixen aspectes com la posició de la pantalla (*portrait* o *landscape*), si la pantalla s'ha d'apagar o no després d'un període d'inactivitat...

- **onCreateEngine**: aquest mètode simplement instancia l'objecte Engine amb els paràmetres definits a la funció anterior. També es defineix la taxa de refresc del renderitzat (imatges per segon, *frames per second* en anglès).

- **onCreateResources**: dins d'aquesta funció es carreguen tots els recursos que s'empraran dins l'activitat, tals com textures, fonts, sons, imatges...

- **onCreateScene**: dins d'aquest mètode s'instancia l'escena que ha d'aparèixer en primer pla. En casos d'aplicacions on només hi ha una escena per activitat, normalment s'aprofita aquesta funció per introduir directament el codi de renderització de l'escena. Tot i així, la millor pràctica consisteix en crear una classe apart que estengui els mètodes de la classe Scene, i dins d'aquesta funció simplement crear-ne una instància. En aquest cas es crea un nou objecte TitleScene que conté tot el conjunt de textos i animacions que apareixen a la pantalla inicial del joc. onCreateScene també serveix per registrar els diferents escoltadors d'esdeveniments necessaris per a l'activitat (esdeveniments tàctils, de refresc, de sensors del dispositiu...).

La classe MainActivity s'implementa com a classe *singleton*, és a dir, si una altra classe vol instanciar-la, n'obté una instància compartida de l'original. D'aquesta manera es poden compartir els recursos propis d'aquella activitat (motor de joc, configuració, atributs...) arreu de l'aplicació.

La Figura 7 mostra un diagrama de classe de MainActivity, amb tots els atributs, constants i mètodes declarats.

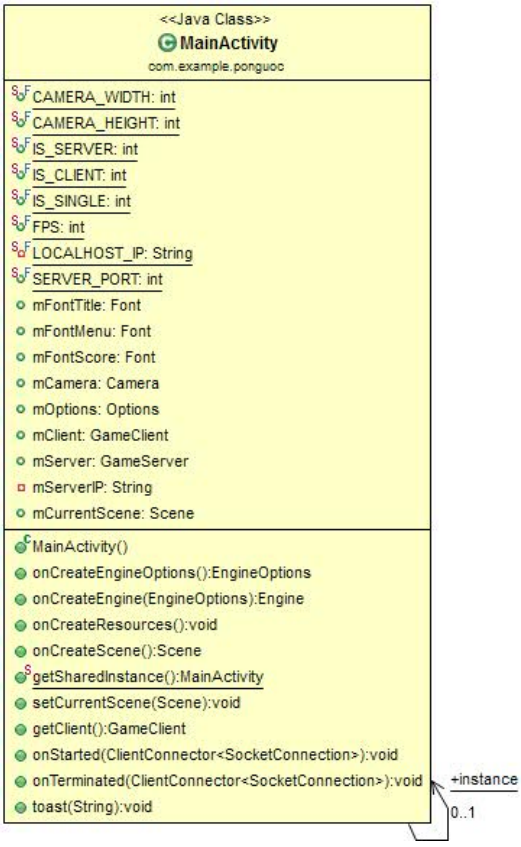


Figura 7. Diagrama de classe de MainActivity

Alguns dels elements destacables de la classe són:

- **CAMERA_WIDTH** i **CAMERA_HEIGHT**: constants que indiquen la mida de la càmera del motor de joc.
- **FPS**: constant que indica el nombre de refrescs de pantalla per segon. Aquest paràmetre és fonamental per a la crida de funcions d'actualització de dades i gràfics, ja que determina el temps que ha de transcórrer entre dues crides.
- A la part de mètodes es poden veure les quatre funcions bàsiques del cicle de vida del motor de joc, definides més amunt.
- **getSharedInstance()**: mètode que retorna una instància compartida de la mateixa classe MainActivity.

2.2 MainMenuScene

Aquesta és l'escena encarregada de mostrar el menú principal del joc, on l'usuari pot decidir el tipus de partida que vol jugar (iniciar nova partida com a servidor, unir-se a una partida ja existent com a client, o jugar sol). La classe estén MenuScene, una classe pròpia d'AndEngine que presenta funcions específiques per a la creació de conjunts d'elements clicables (menús d'opcions). A més, també implementa la interfície IOnMenuItemClickListener, que proveeix les funcions necessàries per a escoltar els esdeveniments tàctils de cada un dels elements del menú i respondre-hi adequadament, en funció del comportament que es desitja per a cada opció concreta. Per aquesta classe val la pena comentar-ne el constructor, per a veure com es defineix una escena tipus menú (recurrent al llarg de l'aplicació).

```
public MainMenuScene() {
	super(MainActivity.getSharedInstance().mCamera);
	activity = MainActivity.getSharedInstance();

	// Define “create new game” option
	IMenuItem serverButton = new TextMenuItem(SERVER_MODE, activity.mFontMenu,
		activity.getString(R.string.server_option),
		activity.getVertexBufferObjectManager());
	serverButton.setPosition(activity.mCamera.getWidth()/2,
		activity.mCamera.getHeight()-200);

	// Define “join existing game” option
	IMenuItem clientButton = new TextMenuItem(CLIENT_MODE, activity.mFontMenu,
		activity.getString(R.string.client_option),
		activity.getVertexBufferObjectManager());
	clientButton.setPosition(activity.mCamera.getWidth()/2,
		activity.mCamera.getHeight()-300);

	// Define “single game” option
	IMenuItem singleButton = new TextMenuItem(SINGLE_MODE, activity.mFontMenu,
		activity.getString(R.string.single_option),
		activity.getVertexBufferObjectManager());
	singleButton.setPosition(activity.mCamera.getWidth()/2,
		activity.mCamera.getHeight()-400);

	// Define “settings” option
	IMenuItem settingsButton = new TextMenuItem(SETTINGS_MODE,
		activity.mFontMenu, activity.getString(R.string.settings_option),
		activity.getVertexBufferObjectManager());
	settingsButton.setPosition(activity.mCamera.getWidth()/2,
		activity.mCamera.getHeight()-500);


		addMenuItem(serverButton);
		addMenuItem(clientButton);
		addMenuItem(singleButton);
		addMenuItem(settingsButton);

		setOnMenuItemClickListener(this);

}
```

El primer pas que es fa és invocar el superconstructor de la classe estesa MenuScene, requisit de la mateixa classe. Seguidament s’obté una instància compartida de la classe MainActivity, ja que per a la definició dels elements del menú s’empraran característiques del motor i recursos definits en aquella classe.

A continuació es procedeix a la creació dels elements de menú. AndEngine proveeix la classe IMenuItem per a la definició d’elements d’aquest tipus. En aquest cas, cada element queda caracteritzat per un identificador únic, una font, un contingut textual i un VBOM:

- **L’identificador**, assignat mitjançant les constants definides a la classe, serveixen per a la identificació de l’element premut.

- La **font** indica la tipologia de lletra, color, mida... de l'element del menú. Aquest recurs es crea durant la inicialització de MainActivity. Per tant en aquest punt es pot veure la importància de crear una instància compartida de l'activitat, amb l'objectiu de poder-ne accedir als recursos des de qualsevol altra classe.
- El **contingut textual** indica el contingut del text de l'element. La definició de les diferents cadenes de text es troben centralitzades al document strings.xml, present al directori de recursos de l'aplicació. Aquesta és una bona pràctica en Android, ja que permet gestionar el contingut textual de l'aplicació des de fora del codi font. D'aquesta manera, per exemple, es poden definir múltiples idiomes per al contingut sense haver de modificar el codi font, ja que el SO s'encarrega de carregar els recursos corresponents a l'idioma definit a les preferències del dispositiu concret.
- **VBOM**: el VertexBufferObjectManager és un objecte propi d'AndEngine que s'encarrega de gestionar els recursos gràfics de l'aplicació.

Una vegada definit l'element del menú, es procedeix a editar-ne la posició (X,Y) per mostrar-lo a una regió concreta de la pantalla. En aquest cas es pot veure que els elements es col·loquen centrats horitzontalment, i separats 100 píxels entre ells verticalment.
Seguidament es procedeix a afegir els elements de menú creats a l'escena. Mitjançant la funció addMenuItem(), es defineixen els elements de menú com a fills de la classe MainMenuScene, permetent així ser visualitzats.
Finalment es registra l'escoltador d'esdeveniments tàctils dels diferents elements del menú, per tal de respondre adequadament als gestos de l'usuari sobre la pantalla.

La resposta de l'aplicació en funció de l'element clicat es defineix al mètode de *callback* de l'escoltador d'esdeveniments tàctils onMenuItemClicked(). Aquest mètode és cridat pel propi SO quan es detecta un gest sobre un element del menú. En funció de l'identificador de l'element clicat s'executa un codi o un altre, depenent del comportament desitjat per part del programador.
En aquest cas, les tres primeres opcions simplement inicien l'activitat GameActivity per mitjà de la funció startActivity() i passant-li com a paràmetre un Intent que conté una constant que indica el tipus de distribució que es vol adoptar (client o servidor) i les opcions del joc.
L'última opció (Settings) instancia un objecte OptionScene (que conté la interfície i la lògica del menú d'opcions) i el declara com a escena de primer pla mitjançant la funció setCurrentScene() definida a MainActivity.
La Figura 8 mostra el diagrama de classe de MainMenuScene. Els atributs són una instància de MainActivity (per poder accedir als recursos i opcions del motor de joc, tal com s'ha definit més amunt) i un objecte Options, que conté les opcions de joc. Les constants definides s'empren per a identificar els diferents elements del menú. També es poden veure els mètodes de la classe.

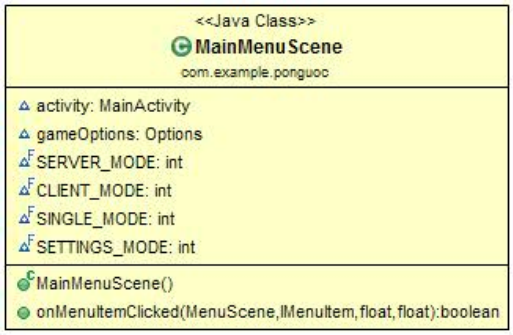


Figura 8. Diagrama de classe de MainMenuScene

2.3 GameActivity

Aquesta classe és la que defineix, entre altres coses, la interfície del terreny de joc i la gestió d'esdeveniments tàctils sobre la mateixa. Igualment, en funció de la naturalesa de la partida creada, instancia un objecte GameClient (si l'aplicació actua com a client), o bé un objecte GameServer i un objecte GameClient (si actua com a servidor). Per tant, es tracta d'un element important de l'aplicació. La Figura 9 mostra el diagrama de classe de GameActivity. A continuació es fa una revisió dels atributs més importants de la classe:

- A la secció de definició de **constants** es determinen les mides de la càmera (CAMERA_WIDTH i CAMERA_HEIGHT), del tauler de joc (TABLE_WIDTH i TABLE_HEIGHT), de les paletes (PADDLE_WIDTH i PADDLE_HEIGHT) i de la pilota (BALL_SIDE), així com el *frame rate* del motor (FPS), l'adreça de loopback del servidor i el port on s'inicialitzarà el socket de servidor, entre d'altres.

- Atributs **mClient** i **mServer**: instàncies de GameClient i GameServer, respectivament. El servidor s'inicialitza dins del constructor de la classe, mentre que el client s'inicialitza quan l'usuari introdueix l'adreça IP del servidor. En el cas de que l'aplicació actuï solament com a client, mServer no s'instancia.

- Atribut **mMessagePool**: s'instancia un Pool de missatges per tal de reciclar els missatges de client emprats per a informar al servidor de la posició de la paleta. Aquest fet és important per millorar el rendiment de l'aplicació, ja que si no s'implementés d'aquesta manera s'estaria generant una instància de missatge 30 vegades per segon (d'acord amb el *frame rate* emprat pel projecte), generant així una quantitat d'objectes inactius molt gran, amb el conseqüent consum de memòria i recursos del sistema. Amb aquesta aproximació, quan un objecte missatge ha estat usat, es retorna al Pool per a tornar a ser emprat quan sigui necessari. Aquesta estratègia s'ha usat en totes aquelles classes que generen missatges de comunicació amb freqüència d'acord amb el *frame rate* de l'aplicació, per tal d'optimitzar-ne el rendiment.

- Atributs **mBall**, **mPaddleList**, **mScoreList**: instàncies de la pilota, les paletes i puntuacions dels dos jugadors, respectivament.

Figura 9. Diagrama de classe de GameActivity

A continuació es descriuen els mètodes més importants de la classe:

Constructor:

Durant la inicialització de la classe es creen els elements gràfics del terreny de joc. Seguidament es mostra la instanciació de la pilota i de les línies de la pista:

```
mBall = new Rectangle(GameActivity.CAMERA_WIDTH/2,
       GameActivity.CAMERA_HEIGHT/2, BALL_SIDE,
       BALL_SIDE, vertexBufferObjectManager);
mBall.setPosition(TABLE_WIDTH/2, TABLE_HEIGHT/2);
mScene.attachChild(mBall);


mScene.attachChild(new Line(0, 0, 0, TABLE_HEIGHT,
       vertexBufferObjectManager)); //left line
mScene.attachChild(new Line(0, TABLE_HEIGHT, TABLE_WIDTH, TABLE_HEIGHT,
       vertexBufferObjectManager)); //top line
mScene.attachChild(new Line(TABLE_WIDTH-1, TABLE_HEIGHT, TABLE_WIDTH-1, 0,
       vertexBufferObjectManager)); //right line
mScene.attachChild(new Line(TABLE_WIDTH, 1, 0, 1, vertexBufferObjectManager));
//bottom line
mScene.attachChild(new Line(0, TABLE_HEIGHT/2, TABLE_WIDTH, TABLE_HEIGHT/2,
       vertexBufferObjectManager)); //central line
```

Com es pot veure, el procediment és molt similar al vist per al menú principal del joc (MainMenuScene): els elements es creen d'acord amb els paràmetres que defineix l'usuari, es configura la posició dels mateixos i a continuació s'inclouen com a elements fill de l'escena en qüestió. El procediment és el mateix per a la creació de les paletes i de la puntuació dels jugadors.

Un cop instanciats tots els elements gràfics, es crida la funció showDialog(), que mostra una finestra de diàleg amb la IP del servidor (si l'aplicació actua com a servidor) o amb un camp de text per a introduir l'adreça del servidor (si actua com a client). Per a més informació, consultar la definició del mètode onCreateDialog(), explicada més avall.
A continuació es registra l'escoltador d'esdeveniments tàctils de l'escena, que s'encarrega de llegir la posició del gest tàctil de l'usuari per determinar la posició de la paleta.
Finalment, es registra l'escoltador d'esdeveniments de refresc amb el mètode registerUpdateHandler(). Aquesta funció implementa el mètode onUpdate(), i cada cop que s'actualitza la interfície gràfica (d'acord amb el *frame rate* del motor) s'executa el codi del seu interior:

```
mScene.registerUpdateHandler(new IUpdateHandler() {
       @Override
       public void onUpdate(final float pSecondsElapsed) {
              if(mClient != null){
                     ClientMsgMovePaddle message = (ClientMsgMovePaddle)
                            mMessagePool.obtainMessage(GameClientMessages.CLIENT_ME
                            SSAGE_MOVE_PADDLE);
                     message.setPaddleID(mPaddleID, mMyPaddleYPos);
                     mClient.sendMessage(message);
                     mMessagePool.recycleMessage(message);
              }
}
```

La funció del codi intern de la funció onUpdate() és simplement enviar un missatge de client al servidor amb la posició actual de la paleta.

Funcions d'update:

La família de funcions updatePaddle(), updateBall() i updateScore() simplement actuen editant la posició (X,Y) de l'element en qüestió, o en el cas de updateScore() modificant el valor de l'string que representa el marcador.

onCreateDialog:

Aquesta funció s'encarrega d'inicialitzar i crear el diàleg a mostrar quan s'executa la funció showDialog(), present en el constructor de la classe. D'acord amb la funcionalitat de l'aplicació (client o servidor) es mostra un diàleg o un altre:

- Si l'aplicació actua com a servidor, es crea una finestra de diàleg informativa anunciant l'adreça IP del servidor, que és allà on s'hauran de connectar els clients. Acte seguit es crida la funció initServerAndClient(), que senzillament inicialitza les variables mClient i mServer.

- Si l'aplicació actua com a client, el diàleg consisteix en un camp de text editable on l'usuari ha d'introduir l'adreça del servidor. Un cop confirma l'acció, es recull el valor del camp de text i s'inicialitza el client amb aquella adreça IP.

Mètodes de gestió de pausa:

La família de mètodes resumePausedGame(), sendFinishMessage(), sendRestartMessage() i onKeyDown() simplement gestionen els missatges que s'envien al servidor quan l'usuari prem el botó de pausa o navega pel menú de pausa de l'aplicació. D'aquesta manera comunica al client l'acció, i aquest pot actuar en conseqüència (reprenent el joc, tornant al menú principal, etc.).

3.3 GameServer

El servidor de joc és un altre dels elements clau de l'aplicació, ja que s'encarrega de gestionar la interfície de comunicació i la computació de la lògica del joc, dos dels pilars fonamentals del disseny. A continuació es descriuen les principals etapes del servidor, així com la relació entre elles:

Inicialització del món físic:

El constructor de la classe conté totes les comandes per inicialitzar el món físic del joc. Com a món físic s'entén tot aquell conjunt d'elements que presenten unes propietats físiques determinades (elasticitat, coeficient de fregament, gravetat...) i que poden interaccionar entre ells. Es creen les parets del terreny

de joc, les paletes i la pilota com a cossos físics, i seguidament es registra un escoltador de contacte entre elements físics. D'aquesta manera el motor de joc gestiona autònomament el contacte entre cossos, i actua en conseqüència en funció del contingut del mètode beginContact() (descrit més avall). Cal remarcar que el cos físic és invisible, i és subjacent a l'element gràfic que representa. Per exemple, el disseny gràfic de la paleta es defineix a GameActivity, com s'ha vist anteriorment, i el cos físic de la paleta es situa per sota de la capa gràfica, per tal de dotar l'element d'unes propietats físiques determinades.

Inicialització de la connexió:

Mitjançant la funció initServer() es crea i s'inicialitza un socket TCP de servidor i un connector de client que escolta les possibles connexions entrants. Dins d'aquest mètode també es registren els missatges de client que el servidor rebrà al llarg del desenvolupament de la partida, i es defineixen les accions que s'han de portar a terme si es rep aquell missatge. Per exemple, a continuació es mostra el procés de registre del missatge de client "pausar joc":

```
clientConnector.registerClientMessage((short)GameClientMessages.CLIENT_MESSAGE_PAUSE_GAME, ClientMsgPauseGame.class, new IClientMessageHandler<SocketConnection>() {
	@Override
	public void onHandleMessage(final ClientConnector<SocketConnection>
			pClientConnector, final IClientMessage pClientMessage) throws
			IOException {
		final ClientMsgPauseGame pauseGameClientMessage =
				(ClientMsgPauseGame)pClientMessage;
		final boolean paused = pauseGameClientMessage.mPaused;
		mStartGame = !paused;
	}
});
```

Com es pot veure, quan el servidor rep un missatge etiquetat amb el flag *CLIENT MESSAGE PAUSE GAME*, aleshores ha de llegir el valor booleà inclòs en el cos del missatge i seguidament modificar el valor de l'atribut mStartGame, la funció del qual és permetre o no que s'executi la funció de refrescar la lògica del joc.
Una vegada registrats tots els missatges, s'inicia el socket de servidor mitjançant la crida al mètode start(), el qual inicia el procés d'escoltament de connexions i missatges entrants.

Execució periòdica de la lògica del joc:

El mètode onUpdate() s'executa cada vegada que es refresca l'escenari, d'acord amb el *frame rate* del motor de joc. En primer lloc es comprova si el flag mResetBall està actiu. Aquest s'activa quan el joc s'inicia o bé quan hi ha un gol per part d'algun jugador. Si la condició és certa, aleshores es reseteja la posició de la pilota, tornant al seu origen inicial i es restableix el vector de velocitat de la mateixa.

Un cop s'ha comprovat el flag mResetBall, es procedeix a actualitzar la posició de la pilota. Un cop es tenen les noves coordenades, es genera un missatge de servidor per informar als clients de la posició actual de la pilota. A continuació, es llegeix la posició de les paletes dels dos jugadors, i també es genera un missatge de servidor per informar als clients de la posició de les dues paletes sobre el terreny de joc. El servidor coneix la posició de les paletes gràcies als missatges que s'envien contínuament des de la classe GameActivity amb la informació de la posició de l'esdeveniment tàctil de l'usuari, com s'ha explicat abans.
Si durant el procés de refresc de la lògica es detecta algun contacte entre dos cossos físics, es fa una crida a la funció beginContact(), explicada a continuació.

Gestió de contactes entre cossos físics:

La principal funció del mètode beginContact() és la de detectar si s'ha produït un gol durant el transcurs d'una partida, és a dir, detectar el contacte de la pilota amb alguna de les parets de fons del terreny de joc. Si això ocorre, aleshores s'incrementa en un punt la puntuació del jugador que ha marcat i es genera un missatge de servidor per informar als clients del canvi en la puntuació. A més, el flag mResetBall esmentat abans s'activa, ja que s'ha de reiniciar la posició de la pilota després d'un gol.
Si en canvi el contacte es produeix entre qualssevol altres cossos (per exemple pilota-paleta o pilota-paret lateral), aleshores només s'inverteix la component del vector de velocitat de la pilota, per tal de simular un rebot.

La Figura 10 mostra el diagrama de classe de GameServer. La funció dels principals mètodes de la classe s'han explicat en els subapartats anteriors. A continuació se'n descriuen els principals atributs:

- Les constants **PADDLE_FIXTUREDEF**, **BALL_FIXTUREDEF** i **WALL_FIXTUREDEF** s'usen per configurar les propietats físiques de les paletes, la pilota i les parets.

- **mPhysicsWorld** és una instància de PhysicsWorld, classe encarregada de gestionar les lleis físiques que regeixen els cossos instanciats com a elements físics.

- **mBallBody** i **mPaddleBodies** contenen instàncies dels cossos físics de la pilota i de les dues paletes, respectivament.

- **mMessagePool** és una instància de Pool, emprada per reciclar els missatges en desús ja emprats pel servidor (tal com s'ha explicat en la secció 3.2 GameActivity)

- **mSocketServer** conté la instància del socket de servidor emprat per a la comunicació amb els clients.

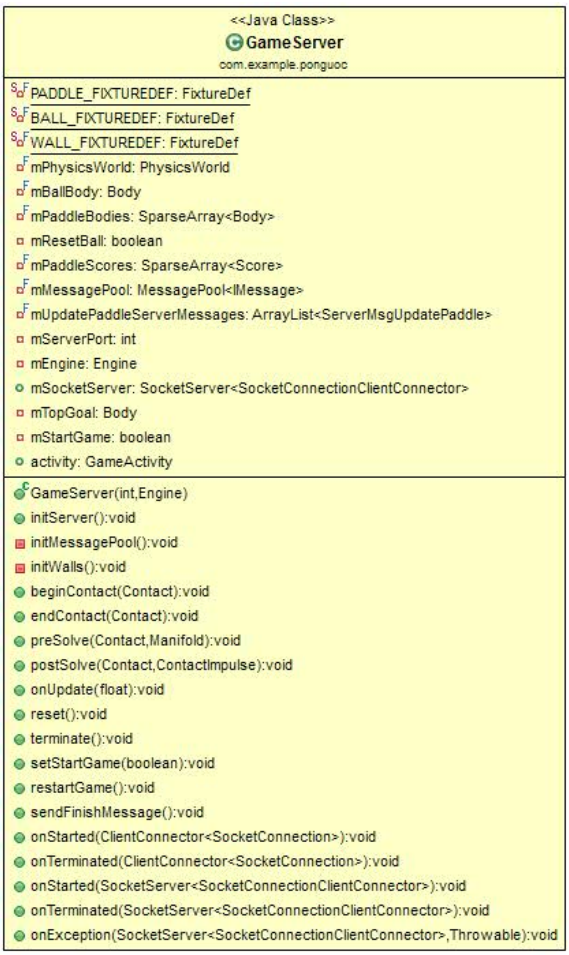


Figura 10. Diagrama de classe de GameServer

3.4 GameClient

La principal funcionalitat del client de joc és establir la connexió amb el servidor, i un cop connectat interpretar els diferents missatges que va rebent per part d'aquest. La Figura 11 mostra el diagrama de classe del client.

Figura 11. Diagrama de classe de GameClient

Els atributs principals d'aquesta classe són:

- **activity**: instància compartida de GameActivity. Necessària per invocar els mètodes d'actualització dels elements gràfics que conté. En funció del missatge que rep del servidor (actualitzar pilota, actualitzar paletes, actualitzar puntuació...) fa la crida al mètode corresponent de GameActivity per a portar-ho a terme.
- **mServerConnector**: instància de la classe ServerConnector, que implementa un socket de client per establir la connexió amb el servidor.

El mètode fonamental de la classe GameClient és initClient(). A continuació es fa una descripció de les parts principals d'aquesta funció:

- **Inicialització connexió**: s'inicia un socket amb l'adreça IP i el port del servidor, i seguidament aquest s'usa per a inicialitzar l'objecte mServerConnector, classe implementada d'AndEngine emprada per a gestionar la connexió.

- **Registre de missatges**: a continuació el client registra tots els tipus de missatges que pot rebre del servidor, i es defineix el codi a executar per cada missatge que arribi. A continuació es mostra un exemple de registre de missatge "actualitzar pilota":

```
mServerConnector.registerServerMessage(GameServerMessages.SERVER_MESSAGE_UPDATE_BALL, ServerMsgUpdateBall.class, new IServerMessageHandler<SocketConnection>(){
	// If a client receives the SERVER_MESSAGE_UPDATE_BALL server message,
	// Fire the following code...
	@Override
	public void onHandleMessage(ServerConnector<SocketConnection>
		pServerConnector, IServerMessage pServerMessage) throws IOException {
		// obtain the class casted server message
		ServerMsgUpdateBall message = (ServerMsgUpdateBall) pServerMessage;
		// Update ball, based on values obtained via the server message received
		activity.updateBall(message.getX(), message.getY());
	}
});
```

Si el client rep un missatge amb el flag *SERVER MESSAGE UPDATE BALL*, aleshores llegeix el missatge per extreure'n les variables corresponents a les coordenades (X,Y) de la pilota, i seguidament invoca la funció updateBall() de GameActivity per a què s'actualitzi la posició de l'element gràfic.

- **Inici de la connexió**: mitjançant el mètode start() de mServerConnector s'inicia la connexió en un fil d'execució dedicat. És important destacar la importància d'executar les connexions en un *thread* diferent del principal, ja que aquestes solen contenir crides bloquejants (funcions que esperen el resultat d'un altre procés per a poder seguir amb l'execució). Per això, si s'executessin en el *thread* principal de l'aplicació (anomenat UIThread), encarregat del renderitzat de la interfície gràfica, aquesta es quedaria congelada en entrar a un procés

bloquejant, resultant en un mal funcionament de l'aplicació i en una nefasta experiència d'usuari. El mateix criteri aplica per les connexions i la computació de la lògica del servidor.

3.5 Altres classes implementades a l'aplicació

- **GameClientMessages**: classe que conté els diferents missatges que usen els clients per comunicar-se amb el servidor. Per cada tipologia de missatge s'ha implementat una classe privada interna. La Figura 12 mostra el diagrama de classes de GameClientMessages. A la Taula 4 es detallen els missatges de client implementats.

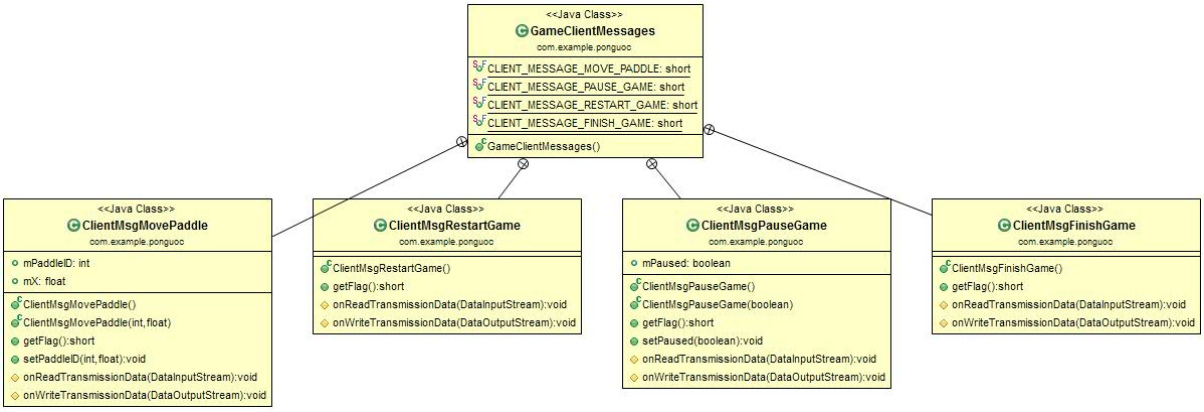


Figura 12. Diagrama de classes de GameClientMessages

Missatge	**Paràmetres**	**Significat**
ClientMsgMovePaddle	- mX: coordenada X	Indica la posició X de la paleta del jugador.
ClientMsgRestartGame	(no)	Indica que el jugador ha restablert la partida.
ClientMsgPauseGame	- mPaused: indicador de l'estat del joc (pausat o no)	Indica que el client ha pausat el joc.
ClientMsgFinishGame	(no)	Indica que el jugador ha abandonat la partida.

Taula 4. Missatges de client

- **GameServerMessages**: igual que el cas anterior, aquesta classe conté els missatges que el servidor usa per a comunicar-se amb els clients. L'estructura és idèntica a la de GameClientMessages: s'usen classes privades internes per a definir cada tipus de missatge. La Taula 5 mostra els detalls de cada missatge de servidor.

Missatge	**Paràmetres**	**Significat**
ServerMsgUpdateScore	- mPaddleID: ID del jugador - mScore: valor de la puntuació	Indica que s'ha d'actualitzar la puntuació del jugador amb ID mPaddleID.
ServerMsgUpdateBall	- mX: coordenada X - mY: coordenada Y	Indica la nova posició de la pilota.
ServerMsgUpdatePaddle	- mPaddleID: ID del jugador - mX: coordenada X - mY: coordenada Y	Indica la nova posició de la paleta del jugador amb ID mPaddleID.
ServerMsgFinishGame	(no)	Indica als clients que el joc s'ha finalitzat (degut a l'abandonament de la partida per part d'un jugador).

Taula 5. Missatges de servidor

- **TitleScene**: classe que conté els elements i animacions de l'escena inicial de l'aplicació.

- **PauseScene**: classe que conté els elements del menú de pausa i en gestiona els esdeveniments tàctils.

- **OptionsScene**: classe que conté els elements del menú d'opcions i en gestiona els esdeveniments tàctils.

- **Score**: objecte que representa la puntuació de cada usuari. Té un mètode per incrementar la puntuació i un altre per posar-la a zero.

- **Options**: objecte que representa les opcions de joc disponibles.

Capítol 4: Implementació

1. Requisits d'instal·lació

Per instal·lar l'aplicació desenvolupada al llarg del projecte es requereix:

- Dispositiu mòbil amb SO Android (mínim API 8) i suport de la interfície WiFi.
- Paquet instal·lable de l'aplicació.

L'aplicació no està verificada per Google ni està penjada a cap plataforma de distribució de continguts, per la qual cosa s'ha d'instal·lar manualment com a "Aplicació de tercers". Tot i aquest fet, no es requereixen coneixements d'informàtica per a portar a terme la instal·lació. En el següent apartat es detalla pas a pas com s'ha d'instal·lar l'aplicació.

2. Instruccions d'instal·lació

A continuació es detalla el procés d'instal·lació de l'aplicació:

1. L'aplicació dissenyada no ha estat verificada per Google ni per cap entitat de confiança. Per minimitzar el risc d'instal·lació de programari maliciós en els dispositius, Android per defecte no deixa instal·lar aplicacions d'origen desconegut. Per a permetre-ho, hem de dirigir-nos al menú "Ajustaments > Seguretat > Orígens desconeguts" i activar l'opció.
2. Carregar l'executable de l'aplicació en format .apk a la targeta de memòria del dispositiu.
3. Accedir des del telèfon al directori on hem copiat l'arxiu, a través del sistema de navegació.
4. Seleccionar l'aplicació perquè s'iniciï la instal·lació.

Capítol 5: Demostració

1. Vídeo demostratiu

El següent enllaç es correspon amb un vídeo on es pot veure l'aplicació en funcionament:

http://vimeo.com/98888713

Capítol 6: Conclusions i línies de futur

1. Conclusions

1.1 Assoliment d'objectius

Si repassem la llista d'objectius que es van definir a l'inici del projecte, podem veure que:

- S'ha implementat correctament l'aplicació multi-jugador. L'arquitectura de la mateixa permet que un dispositiu actuï com a servidor i iniciï una partida, mentre que l'altre jugador agafa el rol de client i es connecta al servidor mitjançant el protocol TCP-IP.
- El dispositiu que actua com a servidor, quan crea una partida obre un socket TCP per escoltar el medi i trobar possibles connexions entrants per part dels clients.
- A nivell acadèmic, s'han estudiat diferents protocols i interfícies de comunicació, i s'ha triat la més adequada d'acord amb l'arquitectura de l'aplicació.
- L'aplicació requereix un nivell d'API mínim molt baix (API 8), i les prestacions necessàries per fer-la córrer són assolibles per la gran majoria de telèfons intel·ligents disponibles al mercat (gràfics molt bàsics, lògica fàcil de computar...)

Per tant, podem concloure que els objectius marcats a l'inici del projecte s'han assolit completament.

1.2 Treball realitzat

Al llarg d'aquest projecte s'han pogut treballar les competències específiques i transversals adquirides al llarg del màster i del grau, així com els coneixements de les diferents àrees o assignatures que componen el màster:

- Programació: implementació de l'aplicació en Java per a la plataforma Android.
- Promoció: model de negoci i estratègia de màrqueting
- Gestió de projectes: planificació i gestió del cicle de vida del Treball Final de Màster com a projecte.
- Ús de l'anglès en l'àmbit de les TIC: lectura de referències bibliogràfiques en llengua anglesa durant l'etapa de recerca d'informació.
- Redacció de documentació tècnica: elaboració de la memòria del projecte

En conclusió, aquest projecte ha comprès diverses àrees complementàries característiques dels projectes multimèdia, fet molt enriquidor per l'estudiant de cara al futur laboral.

1.3 Seguiment de la planificació

La modificació més important que ha sofert la planificació inicial ha estat la quantitat de temps dedicada al desenvolupament de l'aplicació. Inicialment estava prevista una dedicació de 21 dies per aquesta tasca, mentre que la dedicació real final ha estat de 32 dies (gairebé un 50% més del previst). Aquest

allargament ha provocat un endarreriment en les tasques posteriors relacionades, que s'han vist minvades en temps. A més, s'ha hagut de sol·licitar una pròrroga de 6 dies respecte al termini previst d'entrega. De cara a projectes futurs es tindrà en compte el temps dedicat a la fase de programació d'aquest projecte, amb l'objectiu de fer unes previsions més realistes.

Un canvi important que s'ha realitzat respecte a la planificació inicial és l'ús de la interfície WiFi de comunicació en comptes de la interfície Bluetooth que es va preveure al principi, ja que d'acord amb l'arquitectura client-servidor proposada, aquesta última no oferia les característiques necessàries per a establir la comunicació local amb el client executat en el mateix dispositiu. Tanmateix, la implementació i funcionament d'ambdues interfícies és molt similar, per la qual cosa l'arquitectura de l'aplicació Bluetooth que es va fer per l'entrega de la PAC 3 ha servit pel desenvolupament de la interfície WiFi de l'aplicació.

2. Línies de futur

Una possible línia de millora de l'aplicació és el disseny dels elements gràfics del joc. Per aquesta aplicació s'ha fet ús d'una interfície molt senzilla, bàsicament perquè no es va preveure temps dins la planificació inicial per desenvolupar el disseny gràfic dels elements de joc (paleta, pilota, superfície del terreny de joc...). A més del disseny, també es podrien incloure recursos sonors, com ara el soroll de la pilota o una banda sonora pels menús, ja que contribueixen positivament en la percepció de l'aplicació per part de l'usuari. Igualment, es considera la possibilitat d'afegir més opcions de joc, com ara poder triar la mida de la paleta, per tal de disposar de diferents nivells de dificultat.

Per altra banda, un altre camí a seguir en el futur és el de la promoció de l'aplicació. En l'apartat 2 del capítol 2 es proposa un model de negoci i promoció viable pel tipus d'aplicació dissenyada, el qual es podria portar a terme.

Bibliografia

AppBrain. (2014). *Number of available Android applications*. Consultat el 11 / març / 2014, a http://www.appbrain.com/stats/number-ofandroid-

Brownlee, J. (2013). *OpenSource Android Game Engines*. Consultat el 27 / març / 2014, a http://mobilegameengines.com/android/open _source_game_engines

Buford, J., Yu, H., & Lua, E. K. (2009). *P2P Networking and Applications.* Massachussets: Morgan Kaufman.

CDIO. (2014). *Worldwide CDIO Initiative*. Consultat el 17 / juny / 2014, a http://www.cdio.org/

DB-Best Tech. (2013). *Building P2P networks with the help of Bluetooth on Android and iOS devices*. Consultat el 27 / març / 2014, a http://www.dbbest.com/blog/building-p2p-networks-with-the-help-of-bluetooth -on-ios-and-android-devices/

Forbes. (2014). *Sweet And Sour: Candy Crush's Maker Is Going Public, But Is The King Still Reigning?* Consultat el 11 / març / 2014, a http://www.forbes.com/sites/markrogowsky/2014/02/18/sweet-and-sour-candy-crushs-maker-is-going-public-but-is-the-king-still-reigning/

Google - Android Developer. (2014). *Android API Reference*. Consultat el 25 / març / 2014, a http://developer.android.com/reference/packages.html

Google. (2014). *Google Play*. Consultat el 27 / març / 2014, a https://play.google.com/store

Google. (2014). *Google Project - P2P Communication Framework for Android*. Consultat el 27 / març / 2014, a http://code.google.com/p/p2p-communication-framework-for-android/

Gramlich, N. (2010). AndEngine: Free Android 2D OpenGL Game Engine. San Francisco, CA, EUA.

Gramlich, N. (2013). *AndEngine Forum*. Consultat el 25 / maig / 2014, a http://www.andengine.org /forums/development/multiple-scenes-vs-activities-t5477.html

Mulligan, M., & Card, D. (2014). *Sizing the EU app economy*. Consultat el 16 / juny / 2014, a http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=4485

Porter, A. (2014). Programming Mobile Applications for Android Handheld Systems (MOOC). MD, EUA: Coursera.

Schroeder, J., & Broyles, B. (2013). *AndEngine for Android Game Development Cookbook.* Birmingham: Packt Publishing.

SuperMonitoring. (2013). *State of mobile 2013 (infographic)*. Consultat el 11 / març / 2014, a http://www.supermonitoring.com/blog/2013/09/23/state-of-mobile-2013-infographic/

UBM LLC. (2014). *Game Career Guide - What is a Game Engine?* Consultat el 27 / març / 2014, a http://www.gamecareerguide.com/features/529/what_is_ a_game_.php

WIP Connector Ltd. (2013). *Mobile Developer's Guide to the Parallel Universe.* Consultat el 16 / desembre / 2013, a http://wip.org/blog/mobile_developers_guide_to_the_parallel_universe_3rd_edition/

Annexos

Annex A: Lliurables del projecte

Juntament amb aquesta memòria s'adjunten els següents documents:

- Presentació del projecte en format PowerPoint:
    - greines_PresentacioTFM.pptx

- Codi font de l'aplicació:
    - greines_CodiTFM.zip

- Aplicació compilada:
    - PongUOC_com.example.ponguoc_1.0_1.apk