Videojocs lliures: Interacció multiplataforma

Anna Monros Bellart

Director: Antoni Soto i Riera

Gener de 2010

Índex

1 Introducció **1**

1.1 Objectius i metodologia 1

1.1.1 Plataformes escollides 2

1.1.2 Tasques 2

1.1.3 LaTeX 3

1.2 Estructura del document 4

2 Anàlisi del dispositiu: iPhone 3G **5**

2.1 Anàlisi del hardware 5

2.1.1 Especificacions tècniques 5

2.1.2 Versions 7

2.2 Anàlisi del sistema operatiu: iOS 8

2.2.1 Característiques a destacar 8

2.3 Estructura d'una aplicació 10

2.3.1 Directori .app 11

2.3.2 Signatura 13

3 Anàlisi dels entorns de desenvolupament **15**

3.1 iphonedevonlinux 16

3.2 Scratchbox 16

3.3 SDK oficial 17

3.4 Llibreries multi-plataforma 18

3.4.1 Simple DirectMedia Layer 19

3.4.2 libxml2 20
3.5 Frameworks del sistema: Cocoa Touch 20
3.6 Eines de compilació 21

4 Anàlisi del joc **23**
4.1 Algorisme 24

5 Implementació **27**
5.1 Alliberament del dispositiu: Jailbreak 27
5.1.1 Passos a seguir 28
5.2 Instal·lació dels entorns de desenvolupament 34
5.2.1 iphonedevonlinux 35
5.2.2 SDK oficial 37
5.2.3 Scratchbox2 40
5.3 Adaptació del codi 45
5.3.1 SDL 45
5.3.2 SDL_net 46
5.3.3 SDL_mixer, libogg i libvorvis 47
5.3.4 libxml2 47
5.3.5 2Pong 48
5.3.6 Errors en el codi del 2Pong 54
5.4 Compilació del codi 61
5.4.1 Compilació condicional 61
5.4.2 SDL 61
5.4.3 SDL_net 63
5.4.4 2Pong 64
5.5 Depuració del codi 64
5.6 Instal·lació de les aplicacions 66
5.6.1 Signar l'aplicació 66
5.6.2 Crear o modificar l'Info.plist 68
5.6.3 Empaquetar 69

5.6.4 Copiar l'aplicació a l'iPhone 69
5.6.5 Script 72
5.7 Distribució del codi 72

6 Valoració econòmica **75**

7 Conclusions i treball futur **77**
7.1 Possibles millores i ampliacions 78

A Instal·lació i manual del joc **81**
A.1 Instal·lació del 2Pong a l'iPhone 81
A.2 Manual del joc 81

B Arxius Plist **83**

C Warnings del 2Pong **87**

Agraïments

Vull començar donant les gràcies al nostre director de projecte, en Toni, per tota la seva ajuda, dedicació i entusiasme. Vull agrair als meus amics els seus ànims i suport durant tot el projecte i en especial durant la fase final, on són més necessaris que mai, per confiar en mi i fer-me sentir capaç de superar qualsevol entrebanc. En especial a en Guiem perquè sempre puc comptar amb tu, per enganxar-me encara més al món dels videojocs i al programari lliure, pels teus bons consells, i per animar-nos a tirar endavant la idea d'aquest projecte, que sense les teves aportacions no existiria. I a tu Adrià, per tenir sempre una idea original i funcional per resoldre un problema, per estar sempre al meu costat i fer-me sobreviure amb el teu somriure. I sobretot gràcies a tu, Antonio, per ser el company perfecte de PFC, de classe, de joc, de riures i plors, de tot. Perquè sense tu no me n'hauria sortit, els moments difícils passen millor si estàs a la vora. *¿Vamos a echar unos vicios?*

Capítol 1

Introducció

Gairebé totes les grans empreses d'electrònica de consum ofereixen connectivitat amb o sense fils en els seus aparells. Aquesta connectivitat fa uns anys només la trobàvem en PC's, però cada dia es va estenent a més dispositius, com ara consoles, smartphones i fins i tot targes de memòria i bàscules. Al món dels videojocs aquesta connectivitat ha obert moltes portes al mercat. Ara gairebé tots els jocs inclouen d'alguna manera o altre aquesta opció, ja sigui per jugar en xarxa amb altres persones, mostrar els resultats en un rànquing públic, descarregar actualitzacions, comprar nous complements, etc. Fins i tot han aparegut nous gèneres basats en aquesta característica, com els MMORPG [79], en que el joc es juga exclusivament en xarxa.

Tot i l'estandardització de la pila TCP/IP (nivells 3 i 4 del model OSI) sobre un enllaç físic Wi-Fi (nivell 1 del model OSI), estàndard 802.11 de l'IEEE [57], habitualment es veta la interconnexió entre diferents plataformes per raons de màrqueting. Sabent que l'ús d'aquest protocol és igual per a totes les plataformes que l'usen hem volgut sobrepassar aquesta barrera i demostrar que es pot jugar a un videojoc en xarxa entre dos dispositius de companyies diferents.

1.1 Objectius i metodologia

L'objectiu principal del projecte és portar un videojoc lliure i multi-plataforma ja existent a dispositius de companyies diferents i permetre el joc en xarxa entre ells. Per a assolir-lo usarem només eines de programari lliure [39] i multi-plataforma [28].

Altres objectius són entendre i saber usar correctament les eines de programació multi-plataforma, la metodologia de la compilació creuada, els entorns de desenvolupament del sistemes escollits, i les nocions bàsiques de com es programa i s'estructura un videojoc senzill.

La metodologia emprada en aquest projecte és la cerca d'un videojoc lliure programat en

C++ i SDL originalment per a PC i la migració cap a les plataformes escollides. Primer s'hauran de preparar els entorns de desenvolupament per a cada plataforma, i després fer els canvis que calguin en el codi per adaptar-lo a les particularitats de cada màquina. El procediment per aconseguir aquests objectius es detalla a la secció 1.1.2.

1.1.1 Plataformes escollides

Vam decidir utilitzar la consola portàtil Nintendo DS i el dispositiu mòbil iPhone perquè disposaven de projectes per portar la llibreria SDL a aquestes dues plataformes i totes dues comptaven amb connexió de xarxa sense fils. Aquesta elecció no ens suposava despeses addicionals perquè ja disposàvem dels dispositius.

Altres raons per a l'elecció van ser que la Nintendo DS és la consola portàtil més popular del mercat i el fet de tenir dues pantalles i control tàctil la fa diferent de la resta. En el cas de l'iPhone tenim un dispositiu que no és una consola portàtil però s'ha convertit en una de les plataformes més populars per al desenvolupament de jocs. De les aplicacions disponibles per a iPhone, un 15% són videojocs [11]. La seva pantalla multi-tàctil i l'absència de botons la fa interessant.

Ambdues plataformes són sistemes tancats. En el cas de la Nintendo DS no és possible programar fàcilment donada la dificultat de l'accés al SDK oficial. Pel que fa a l'iPhone sí que podem accedir a un SDK oficial però aquest és tancat i per tant no compleix amb el requisit de treballar amb eines lliures. Aquest fet va suposar un incentiu extra per al nostre projecte.

Encara que no estava previst inicialment una vegada començat el projecte va sorgir la idea de treballar sobre una nova plataforma i vam escollir la PlayStation Portable (PSP), que és una plataforma interessant pel fet de ser la principal competidora de Nintendo DS en el mercat de les videoconsoles portàtils.

Aquest projecte ha estat fruit d'un treball conjunt amb l'Antonio Escañuela en que cada un de nosaltres ha treballat sobre una de les plataformes escollides. Aquesta memòria descriu el treball fet sobre el dispositiu iPhone 3G. La informació sobre els ports a Nintendo DS i PSP es troba descrita a [131].

1.1.2 Tasques

En aquesta secció dividirem els objectius del PFC en una llista de tasques per esclarir els passos que s'han seguit en la realització del projecte.

1. Estudiar plataformes o llibreries que permetin fer jocs portables.
2. Buscar un joc lliure ja existent, que estigui implementat en llenguatge C/C++ i amb la llibreria SDL, i que tingui joc multi-jugador en xarxa.

3. Escollir les plataformes per portar el joc.

4. Preparar entorns de desenvolupament lliures per tal de poder compilar codi per les plataformes escollides.

   (a) Entendre les nocions de compilació creuada.

   (b) Estudiar l'arquitectura de les plataformes escollides.

   (c) Preparar els dispositius per a que puguin executar codi de tercers.

   (d) Instal·lar un entorn GNU/Linux amb les eines necessàries.

   (e) Buscar i instal·lar entorns de compilació creuada lliures (toolchain) pels dispositius triats.

   (f) Provar que l'entorn ha estat instal·lat correctament.

   (g) Provar altres entorns de desenvolupament.

5. Compilar la llibreria SDL i altres llibreries necessàries per compilar el joc en les plataformes escollides.

6. Portar el joc a les màquines escollides.

   (a) Compilar el joc amb els entorns de compilació creuada.

   (b) Adaptar el codi del joc a les necessitats de cada plataforma.

   (c) Provar i arreglar la comunicació entre els dispositius.

7. Escriure la memòria del projecte.

1.1.3 LaTeX

Per escriure aquesta documentació hem usat LaTeX [75], un sistema de composició de textos utilitzat per a la creació d'articles acadèmics, tesis i llibres tècnics donat la qualitat tipogràfica dels documents realitzats amb ell.

LaTeX és un llenguatge de programació en alt nivell basat en TeX [118]. Donat que les ordres de TeX són a molt baix nivell per a l'usuari és més simple escriure en LaTeX.

A diferència d'altres eines de composició de textos, LaTeX funciona per ordres, pel que vam haver de passar per una fase d'aprenentatge, mitjançant el llibre [130], i altres fonts com [125].

LaTeX és una eina lliure pel que s'adequa perfectament als requisits del nostre projecte.

1.2 Estructura del document

Tal com s'ha esmentat anteriorment aquest projecte ha estat realitzat per dues persones, i per tant, per tenir una visió completa de tota la feina realitzada és necessària la lectura de dues memòries. En les memòries trobarem parts comunes i parts específiques segons la plataforma escollida. Les comunes són:

- Introducció (secció 1)
- Introducció del capítol "Anàlisi dels entorns de desenvolupament" (secció 3)
- Anàlisi de Scratchbox (secció 3.2)
- Llibreries multi-plataforma (secció 3.4)
- Eines de compilació (secció 3.6)
- Anàlisi del joc (secció 4)
- Instal·lació de Scratchbox2 (secció 5.2.3)
- Errors en el codi del 2Pong (secció 5.3.6)
- Distribució del codi (secció 5.7)

El gruix del document s'estructura en dos grans blocs, anàlisi i implementació. En el primer es fa un estudi del dispositiu escollit, analitzant el hardware i sistema operatiu, després s'estudien els entorns de desenvolupament utilitzats i finalment s'estudia el joc que hem adaptat. En el segon bloc es detallen totes les tasques dutes a terme.

Capítol 2

Anàlisi del dispositiu: iPhone 3G

En aquesta secció estudiarem tant el hardware com el sistema operatiu de l'iPhone 3G, per saber quines són les seves capacitats i limitacions.

L'iPhone és un smartphone [111] dissenyat i comercialitzat per Apple Inc [18]. El primer model es va llançar a Estats Units al juny de 2007. Uneix en un sol dispositiu les funcions de telèfon mòbil, reproductor multimèdia, GPS [47] i connexió per xarxa Wi-Fi [123], Bluetooth [22] i 3G [4]. El dispositiu compta amb una pantalla tàctil capacitiva [121] i no disposa de teclat físic. Com a sistema d'emmagatzematge compta amb una memòria flash que oscil·la des dels 4 GB fins als 32 GB segons el model. L'iPhone es ven amb el sistema operatiu preinstal·lat iOS, antigament anomenat iPhoneOS.

2.1 Anàlisi del hardware

2.1.1 Especificacions tècniques

Necessitem conèixer el hardware en el que anem a programar, per saber quines son les seves possibilitats i peculiaritats a l'hora de programar. No entrarem en molt de detall, perquè no és el nostre objectiu, però es pot obtenir més informació en algunes fonts com [64]. Les dades d'aquesta secció corresponen a l'iPhone 3G, però s'ha de tenir en compte que hi ha variacions en el hardware en les diferents versions del dispositiu. Tenim un resum del hardware i de les diferències entre versions a la secció 2.1.2.

CPU L'iPhone compta amb un processador Samsung 32-bit RISC ARM 1176JZ(F)-S [20] que originalment treballa a 620 MHz però al que se li ha baixat la freqüencia (underclocked [122]) fins a 412 MHz. Aquesta baixada de freqüència limita el rendiment del dispositiu però així la temperatura del dispositiu i de la bateria es manté més baixa, allargant-ne la vida útil. Una freqüència de rellotge més baixa també allarga la durada de la bateria.

GPU La GPU és un PowerVR MBX Lite 3D [80] i permet renderització 2D i 3D.

Memòria 128 MB DRAM [34]

Só Quant a só, l'iPhone usa el còdec d'àudio Wolfson Microelectronics WM6180C [126].

Connectivitat L'iPhone disposa de diverses connexions sense fils. Els protocols que implementa són:

Wifi Ofereix connectivitat estàndard Wi-Fi 802.11b/g.

2G Inclou GSM/GPRS/EDGE (850, 900, 1800, 1900 MHz) quadri-banda.

3G Inclou UMTS/HSDPA (850, 1900, 2100 MHz) tri-banda.

Bluetooth Bluetooth 2.0.

Pantalla L'iPhone disposa d'una pantalla LCD de 3.5" (262.144 colors), 480x320 píxels de resolució, relació d'aspecte 3:2 (HVGA), 163 ppi. I està coberta per un vidre resistent a ratllades.

Entrada Una de les característiques més rellevants de l'iPhone és el seu mètode de control. Gràcies a la seva pantalla capacitiva [121] multi-tàctil [82] podem controlar el dispositiu sense la necessitat de molts botons físics. De fet l'iPhone 3G només compta amb tres. Un per l'encesa i l'apagada del dispositiu, un altre per bloquejar el só i un altre per tancar les aplicacions.

A més compta amb tres sensors, un de llum, un de proximitat i un acceleròmetre de tres eixos que detecta l'acceleració del dispositiu.

Càmera L'iPhone té una càmera posterior amb 2.0 MP de resolució. Les fotos inclouen geoetiquetatge [46], cosa que permet saber les coordenades geogràfiques en el moment de la captura.

Emmagatzematge L'iPhone 3G va ser llançat al mercat en dues versions que es diferenciaven per la mida de la seva memòria flash integrada de 8 o 16 GB de capacitat.

Bateria La bateria és recarregable, d'ió liti en polímer de 3.7 V i 1150 mA·h. El principal inconvenient és que no es pot extreure. Si necessita ser canviada cal enviar tot el dispositiu a Apple. S'estima que la seva duració en hores és de 24 reproduint àudio, 7 reproduint vídeo, 5 navegant sobre 3G, 9 usant Wi-Fi i 300 en standby.

Dimensions 115.5 x 62.1 x 12.3 x mm

Pes 133 g

	iPhone	**iPhone 3G**	**iPhone 3GS**	**iPhone 4**
CPU	ARM11 a 412 MHz	ARM11 a 412 MHz	ARM Cortex-A a 600 MHz	ARM Cortex-A8 Apple A4 a 800 MHz
GPU	PowerVR MBX Lite 3D	PowerVR MBX Lite 3D	PowerVR SGX535	PowerVR SGX535
RAM	128 MB DRAM	128 MB DRAM	256 MB DRAM	512 MB DRAM
Resolució pantalla	480 × 320 px	480 × 320 px	480 × 320 px	960 × 640 px
Códec d'àudio	Wolfson Microelectronics WM8758BG	Wolfson Microelectronics WM6180C	Cirrus Logic CS42L61	Cirrus Logic CS42L61
Càmera	2.0 MP	2.0 MP	3.0 MP Permet gravar vídeo a 640×480 px a 30 fps	Càmera frontal de 3.0 MP i càmera posterior de 5.0 MP amb flash. Permet gravar vídeo a 1280×720 px a 30 fps
Emmagatzematge	4, 8 o 16 GB de memòria flash	8 o 16 GB de memòria flash	8, 16 o 32 de memòria flash	16 o 32 GB de memòria flash
Dimensions	115 × 61 × 11.6 mm	115.5 × 62.1 × 12.3 mm	115.5 × 62.1 × 12.3 mm	115.2 × 58.6 × 9.3 mm
Pes	135 grams	133 grams	135 grams	137 grams

Taula 2.1: Hardware de les diferents versions d'iPhone

2.1.2 Versions

Existeixen 4 generacions de models d'iPhone. A cada nova versió s'ha millorat el hardware i s'ha evolucionat en la versió del sistema operatiu.

iPhone: El primer model que no inclou connexió 3G ni GPS Assistit. Ja està descatalogat.

iPhone 3G: Va aparèixer al al mercat el juliol de 2008 i ja no es troba en producció.

iPhone 3GS: Apareix al juny de 2009. Inclou brúixola digital. En producció.

iPhone 4: Model més nou d'iPhone. Amb la seva aparició el sistema operatiu passa a anomenar-se iOS. Llençat al mercat al juny de 2010. Un parell de novetats de hardware a destacar són la millora de la resolució de la pantalla i la incorporació d'una càmera frontal per a videoconferències.

Per la realització d'aquest projecte s'ha treballat amb un iPhone 3G.

A la taula 2.1 es resumeixen les diferències més importants. Si volem una comparació més exhaustiva podem consultar aquest recurs [70].

2.2 Anàlisi del sistema operatiu: iOS

El sistema operatiu (SO [89]) del dispositiu és l'iOS [59], prèviament anomenat iPhoneOS. El canvi de nom ha estat degut a que la companyia ha tret al mercat més dispositius que usen el mateix sistema operatiu, l'iPod Touch, l'iPad i l'Apple TV, i que amb la seva aparició no té sentit que en el nom del sistema operatiu aparegui la paraula iPhone. L'iOS és el sistema operatiu d'Apple per als seus dispositius mòbils i deriva del Mac OS X que a la vegada està basat en Darwin BSD, una distribució de UNIX.

Existeixen diverses versions de l'iOS, a l'hora d'escriure aquest document la darrera versió estable és la 4.2. A l'inici d'aquest projecte la darrera versió era la 3.0, i per tant la que hem usat.

Durant la lectura d'aquest document es trobaran referències a *firmware* i a iOS que poden induir a confusió. Volem aclarir que quan ens referim al firmware del dispositiu estem fent menció al paquet de programari que inclou el sistema operatiu (nucli i aplicacions) de l'iPhone. En concret mantindrem que el firmware x.y correspon a la versió del sistema operatiu x.y. Per exemple, el firmware 3.0 inclou el sistema operatiu iOS 3.0.

2.2.1 Característiques a destacar

Interfície gràfica

Tots els dispositius d'Apple que funcionen sota iOS (a excepció de l'AppleTV) disposen d'una pantalla multi-tàctil [82]. La interfície d'usuari de l'iOS es basa en el concepte de la manipulació directa, reconeixent els gestos multi-tàctils que podem fer a les pantalles dels dispositius. El nombre de botons físics en els dispositius mòbils és mínim, i la interacció amb el dispositiu es fa majoritàriament a través de botons, lliscadors i interruptors digitals que activem amb els dits.

Aplicacions

El disseny de la interfície gràfica es basa en la gestió de les anomenades aplicacions. A la pantalla de tots els dispositius mòbils amb iOS només hi apareixen icones que representen aplicacions. Al prémer una icona llancem l'aplicació. Les aplicacions es poden ordenar manualment i a partir de la versió 4.0 és possible crear carpetes per agrupar-les.

Els dispositius venen de fàbrica amb un seguit d'aplicacions preinstal·lades. En el cas de l'iPhone 3G les podem veure a la taula 2.2.


Phone: per fer trucades telefòniques.


Mail: un client de correu electrònic.


Safari: un navegador web.


iPod: reproductor multimèdia.


Missatges: per enviar i rebre missatges sms i mms.

Calendari: per administrar cites o events.


Fotos: un visor de fotos.


Càmera: per capturar imatges.


Youtube: una interfície adaptada a l'iPhone del popular site de vídeos.


Borsa: visor de cotitzacions en borsa.


Maps: versió de Google Maps.

Temps: versió de Yahoo! Weather.


Notes: per escriure notes en text pla.


Rellotge: amb cronòmetre, alarma i temporitzador.


Calculadora: per realitzar operacions matemàtiques.


Ajustaments: ajustaments del dispositiu.


Contactes: agenda de contactes.


iTunes: botiga virtual per comprar música.


App Store: botiga virtual per comprar aplicacions. Aquesta darrera aplicació potser és la més important, perquè és l'única via per instal·lar contingut al nostre dispositiu, i per tant, també una de les majors limitacions d'iOS, que explicarem més endavant. A hores d'ara l'App Store compta amb 325.672 aplicacions [11].

Taula 2.2: Aplicacions preinstal·lades a l'iPhone 3G

Multitasking

El multitasking, o multitasca [26] ha estat sempre possible amb el hardware de l'iPhone, però no ha estat fins a l'aparició de la versió 4.0, instal·lada per omissió al nou iPhone 4, que se n'ha permès l'ús. Aquesta funcionalitat no estava disponible en els dispositius més antics al·ludint a que la manca d'un hardware prou potent faria que la fluïdesa i la duració de la bateria del dispositiu es veiés disminuïda notablement.

Game Center

Aparegut amb la versió 4.0. El Game Center és una xarxa social de joc online multiplayer. Consisteix en una plataforma per jugar amb altres persones, compartint puntuacions, rècords, etc.

2.3 Estructura d'una aplicació

Per a que un contingut sigui executable a l'iOS ha de complir amb una sèrie de requisits. Aquests continguts són anomenats aplicacions. Les aplicacions per a iPhone tenen una estructura que s'ha de respectar per tal que l'iOS les reconegui com a tal.

A l'igual que a Mac OS X, quan instal·lem una aplicació iOS a un dispositiu es crea un directori `home` que l'encapsa. Les aplicacions no poden accedir fora d'aquest directori, cosa que atorga seguretat al sistema i a més fa més fàcil la tasca de desinstal·lació de l'aplicació, ja que només cal esborrar aquest directori per assegurar-nos de que no queda rastre de l'aplicació al sistema [119].

Quan l'aplicació s'han instal·lat l'arbre de directoris queda de la següent manera:

```
app_home/
  app.app
  Documents/
  Library/
    Preferences/
  tmp/
```

Quan es llança l'aplicació el directori de treball passa a ser `app_home/app.app`. L'escriptura no està permesa en aquest directori, si ho volem, ho haurem de fer al directori `app_home/Documents` o si hem d'escriure preferències a `app_home/Library/Preferences`.

```
NSURL *appURL =
    [NSURL URLWithString:[NSString stringWithFormat:@"http://www.itunes.
       com/app/%@",
        [[[NSBundle mainBundle] infoDictionary] objectForKey:@"
           CFBundleDisplayName"]
            stringByAddingPercentEscapesUsingEncoding:
               NSUTF8StringEncoding]]];
```

Figura 2.1: Codi per llegir una clau de l'arxiu Info.plist

2.3.1 Directori .app

El directori .app és el que conté tots els arxius necessaris per a l'execució de l'aplicació. Els arxius que ha de contenir es detallen a continuació.

Executable

El binari executable que llança l'aplicació.

Arxiu de propietats

Un dels arxius bàsics que s'han d'incloure és l'anomenat *Info.plist*. Els arxius *plist* [92] contenen una llista de metadata referent a l'aplicació que és interpretada pel sistema operatiu. En el cas de l'iPhone aquest arxiu pot emmagatzemar diversa informació, com l'identificador de l'aplicació, icona, versió, etc. Però també pot controlar l'accés a certes funcionalitats de l'iPhone, com per exemple vetar l'ús de la xarxa Wi-Fi o permetre reproduir sons. Nosaltres també podem llegir aquesta informació des del nostre programa, per exemple, si volem llegir una url escriurem el codi (en Objective-C) que s'indica a la figura 2.1.

Aquests arxius es poden trobar en diversos formats, el binari, l'XML i el text pla, tots ells són acceptats en incloure'ls en una aplicació. Els formats text i XML són més còmodes a l'hora de visualitzar-los, ja que els podem obrir amb qualsevol editor de textos. En canvi per visualitzar el format binari necessitarem un editor d'arxius *.plist*.

Existeixen varis editors d'aquest tipus d'arxius per a Windows i Mac. Python [96] també ofereix un mòdul que permet la lectura i l'escriptura en aquest format. Apple en proporciona un de propi, el *Property List Editor*, inclòs en les seves Developer Tools [16]. Hem usat aquest darrer per fer algunes proves.

Apple ofereix informació detallada de com crear i usar aquest arxius a la seva llibreria de referència d'iOS [93]. A l'apèndix B podem veure l'aparença d'un mateix arxiu Info.plist en els diferents formats.

Mida en píxels	Nom de l'arxiu	Ús
57x57	Icon.png	App Store i pantalla principal de l'iPhone i l'iPod touch
114x114	Icon@2x.png	Pantalla principal de l'iPhone 4
72x72	Icon-72.png	Pantalla principal de l'iPad
29x29	Icon-Small.png	Spotlight[1] i preferències[2]
50x50	Icon-Small-50.png	Spotlight a l'iPad
58x58	Icon-Small@2x.png	Spotlight i preferències a l'iPhone 4
512x512	iTunesArtwork	Informació iTunes

[1] Cercador intern de l'iPhone. Al buscar aplicacions mostra la miniatura de la icona.
[2] Algunes aplicacions tenen opcions que es poden modificar des del menú preferències de l'iPhone.

Taula 2.3: Descripció de les icones que es poden usar per a iPhone

Algunes de les claus més comunes que inclouen els *Info.plist* de l'iPhone són:

- `CFBundleDisplayName` Nom amb el que es mostra l'aplicació a la pantalla principal de l'iPhone.
- `CFBundleExecutable` Aquesta clau indica el nom del binari executable de l'aplicació que es troba dins el directori *.app*. El nom de l'executable no cal que sigui el mateix que el de l'aplicació.
- `CFBundleIdentifier` Amb aquesta propietat creem un identificador únic per a la nostra aplicació i per tant cal que sigui diferent al de la resta de les nostres aplicacions. És habitual usar la forma `"com.[companyia].[nomaplicació]"` per assegurar-se que és única.
- `MinimumOSVersion` Aqueta clau indica la versió mínima de l'iOS que cal tenir installada per poder executar l'aplicació.

Icona

Si volem que la nostra aplicació aparegui amb una icona determinada a la pantalla principal del dispositiu l'haurem d'incloure al directori *.app*. Per tal que l'iPhone la reconegui com a icona haurà de complir els requisits indicats per Apple. Aquesta informació la podem trobar a la llibreria de referència oficial d'iOS [9].

Per veure la icona només és necessària una imatge de 57x57 píxels amb nom *Icon.png*. Podem afegir més versions amb altres dimensions, que no són obligatòries però que es recomana la seva inclusió si volem que la nostra icona es vegi amb bona qualitat a altres dispositius amb iOS, com l'iPhone 4, que té una resolució major, o l'iPad. A la taula 2.3 trobem la descripció de totes les disponibles.

```
        <key>CFBundleIconFile</key>
        <string></string>
        <key>Icon files</key>
        <array>
                <string>Icon.png</string>
                <string>Icon@2x.png</string>
                <string>Icon-72.png</string>
                <string>Icon-Small.png</string>
                <string>Icon-Small-50.png</string>
                <string>Icon-Small@2x.png</string>
        </array>
```

Figura 2.2: Claus referents a les icones a l'Info.plist (XML)

```
    CFBundleIconFile = "";
        "Icon files" = (
                "Icon.png",
                "Icon@2x.png",
                "Icon-72.png",
                "Icon-Small.png",
                "Icon-Small-50.png",
                "Icon-Small@2x.png",
        );
```

Figura 2.3: Claus referents a les icones a l'Info.plist (text pla)

També haurem d'afegir la referència a les imatges a l'Info.plist afegint les claus que s'indiquen a la figura 2.2, en el cas del format XML, i a la figura 2.3, en el cas del format de text pla.

Si no especifiquem cap icona ens apareixerà una imatge en blanc.

2.3.2 Signatura

Apple obliga a que totes les aplicacions que corren sobre els seus dispositius amb iOS estiguin signades amb un certificat de desenvolupador. Donat que Apple revisa cada una de les aplicacions que es volen publicar a la seva botiga virtual aquest és un mètode per identificar al seu desenvolupador i tenir un control de tot el catàleg d'aplicacions, és a dir, no es permet la lliure distribució i per tant aquestes aplicacions mai seran lliures[1].

Que les aplicacions estiguin signades també és un requeriment del propi iOS, ja que el nucli del sistema implementa aquest tipus de control.

[1]Recentment Apple ha eliminat el reproductor multimèdia VLC de la seva botiga per estar llicenciat amb GPL

Apple justifica l'ús d'aquesta signatura dient que es tracta d'un mètode de seguretat tant per als desenvolupadors com per als usuaris, ja que, segons diuen, evita que a la seva botiga s'hi puguin trobar aplicacions fraudulentes. Existeixen diferents mètodes no oficials per signar les aplicacions que s'expliquen a la secció 5.6.1.

Capítol 3

Anàlisi dels entorns de desenvolupament

Un entorn de desenvolupament és el conjunt de processos i eines de programació usades per crear un programa o producte de software. Pot implicar també un entorn físic. En eines de programació s'inclouen eines com compiladors, llibreries i altres eines per a l'ajuda de la programació, com repositoris de codi, eines de disseny de software, eines per a la documentació, simuladors, etc.

Podem dir que la preparació d'aquests entorns no sempre és trivial. Segons la plataforma podem trobar entorns ja muntats i provats i que es distribueixen com a entorns funcionals, que es solen anomenar *toolchain*.

Un toolchain (que podríem traduir com cadena d'eines) és un conjunt de programes informàtics utilitzats per a crear un determinat producte (normalment un altre programa o sistema informàtic). Els diferents programes se solen utilitzar en cadena, de manera que la sortida de cada eina sigui l'entrada de la següent, encara que actualment es fa un abús del terme per referir-se a qualsevol tipus d'eines de desenvolupament enllaçades [120].

Per a la realització de la part d'iPhone s'han usat diversos toolchains, que s'expliquen en les seccions 5.2.1, 5.2.2 i 5.2.3.

Els nostres entorns de desenvolupament compten amb un compilador i les llibreries de sistema (secció 3.5). També necessitarem altres llibreries que són usades pel joc, i que explicarem a la secció 3.4.

Donat que les llibreries i el joc que compilarem estan escrites en llenguatge C/C++ necessitem un compilador d'aquest llenguatge que generi executables per al nostre dispositiu. En el nostre projecte usem el GCC [45], de les sigles de GNU Compiler Collection, perquè treballem sobre GNU/Linux.

Compilar en C/C++ des del mateix iPhone és possible un cop s'ha alliberat (secció 5.1) el dispositiu. A un repositori d'aplicacions per a iPhone que veurem a la secció 5.1 (Cydia),

podem trobar el GNU C Compiler i l'eina make. El problema és que quan intentem baixar-nos el GCC se'ns informa que la instal·lació no pot ser completada degut a que no es pot arreglar una dependència. Aquesta dependència és la glibc (libgcc a Cydia), la llibreria estàndard de C. La llibreria no es troba disponible per a la versió 3.0 del firmware [38] de l'iPhone, la que nosaltres usem.

Un cop s'ha descartat la possibilitat de compilar des del mateix dispositiu, ens cal estudiar el concepte de compilació creuada.

La *compilació creuada* consisteix en crear codi executable per a una plataforma diferent d'aquella en la que es compila [27]. Això és necessari quan volem executar codi en una plataforma en la que no el podem compilar (degut a les seves limitacions) o no és còmode fer-ho, com és el nostre cas. És a dir, compilarem els binaris en un PC però segons l'arquitectura d'un iPhone 3G, de manera que finalment puguin ser executats en aquesta.

Existeixen dues vies per al desenvolupament d'aplicacions per a iPhone 3G usant entorns de compilació creuada, per una part l'SDK [114] oficial ofert per Apple i per l'altre usant un toolchain lliure, en el nostre cas, iphonedevonlinux. Com que una de les premisses del PFC és fer l'adaptació del joc usant només eines lliures es donarà prioritat a iphonedevonlinux, ja que l'SDK oficial és un entorn privatiu. Tot i això també s'explicarà el desenvolupament usant l'SDK donada la seva importància a la comunitat de desenvolupadors i perquè una gran part de les eines que usa són de GNU, i per tant lliures. També s'ha experimentat amb un entorn lliure creat amb la combinació d'iphonedevonlinux i scratchbox, a la secció 3.2 hi trobarem la informació.

3.1 iphonedevonlinux

No es poden trobar molts toolchains per desenvolupar aplicacions per a iPhone en GNU/-Linux. Quan busquem informació a la xarxa sobre com programar per a iPhone, gairebé tots els exemples usen l'SDK oficial d'Apple.

En un primer moment es va optar per usar iphone-dev [68] ja que estava referenciat a [132] i es va intentar fer la instal·lació, però durant aquesta ens vam trobar amb un error de compilació que segons la documentació encara no estava resolt [72]. Al veure que no obteníem una solució al cap d'un temps es va optar per buscar un altre toolchain. Finalment es va optar per instal·lar el toolchain **iphonedevonlinux**, que té suport per al firmware 3.0.

3.2 Scratchbox

La compilació creuada pot comportar problemes, per exemple, a l'hora d'usar llibreries que estan presents a ambdós sistemes (on es compila i on s'executa el codi), pot ser que

perdem el control i no sapiguem quina versió de la llibreria estem usant. Per això ens vam plantejar l'ús d'una eina que ens permetés evitar aquests problemes. El nostre director de PFC ens va recomanar l'estudi d'Scratchbox [98] donat que coneixia el seu ús per al desenvolupament per a Maemo [78], el sistema operatiu d'alguns dispositius Nokia [84], com el 770, N800, N810 i N900.

Scratchbox crea un entorn segur de compilació creuada sobre GNU/Linux basat en la idea de chroot [24]. Chroot en un sistema operatiu Unix canvia el directori arrel a un de diferent de "/". Un programa que s'executa en un entorn chroot no pot accedir a arxius fora d'aquest directori creant una mena de gàbia.

Suporta arquitectura ARM i x86, cosa que el fa idoni per al nostre cas, ja que l'arquitectura de la CPU de l'iPhone i la Nintendo DS és ARM. Encara que es pot instal·lar sobre les dues arquitectures és més habitual usar-lo per fer desenvolupament per a ARM sobre una màquina x86. Scratchbox usa QEmu [97] per emular un processador ARM. Està llicenciat sota LGPL i per tant compleix amb les premisses del PFC.

Scratchbox ofereix diversos toolchains per ser usats amb ell i també ofereix l'opció d'usar-ne un de propi, com és el nostre cas. Durant la instal·lació vam tenir problemes en aquest punt, ja que no sabíem com incloure'l, per això vam demanar informació a la llista de correu oficial. Un dels usuaris ens va recomanar Scratchbox2, un projecte separat de l'Scratchbox original. Donada la seva senzilla instal·lació ens vam decantar per aquesta opció.

Cal dir que Scratchbox2 disposa de molt poca documentació i no s'especifica amb claredat les diferències que presenta amb Scratchbox. El projecte va ser abandonat pel seu creador (Lauri Leukkunen) durant la realització d'aquest PFC, al març del 2010. Segons s'informa a la seva web ara el projecte està mantingut per Nokia.

Finalment no s'ha usat Scratchbox per a la realització del PFC per un tema de comoditat, ja que amb els makefiles creats per nosaltres teníem un major control. Encara que l'hem instal·lat i hem fet proves sobre ell que es detallen a la secció 5.2.3.

3.3 SDK oficial

Apple no permet que l'iOS corri en hardware d'altres companyies. Tampoc permet que el software creat sense seguir les directrius d'Apple corri sota l'iOS. Tal com hem comentat, per tal d'instal·lar programari al nostre dispositiu hem d'adquirir-lo a través de l'App Store que està regulada per Apple. Per saltar-se aquesta segona restricció hem de fer un jailbreak al nostre iPhone. El procediment es descriu a la secció 5.1.

Les directrius d'Apple per crear contingut executable a l'iPhone consisteixen en:

1. Registrar-se com a desenvolupador d'Apple [13]. Aquest registre és gratuït i ens permet la descàrrega de totes les eines necessàries per al desenvolupament en iOS, Mac OS X i Safari, així com l'accés a informació relacionada.

2. Descarregar l'SDK, accessible a l'iOS Dev Center [60], el portal específic d'iOS, una vegada s'ha completat el registre. L'ús d'aquest SDK també implica una sèrie de requeriments de hardware i software [61]. Necessitem disposar d'un ordinador Mac amb arquitectura Intel, i tenir instal·lat com a SO mínim Mac OS X Snow Leopard (versió 10.6).

   L'SDK inclou:

   - Xcode [127], que inclou un IDE [58], un editor gràfic per al disseny d'interfícies, un simulador d'iOS i l'Organizer, un gestor de dispositius amb iOS amb el que podem accedir a una consola i als logs d'error de les aplicacions.
   - Suport per a Objective-C 2 [85], C i C++.
   - Les llibreries de sistema de l'iPhone.

Amb la subscripció tenim dret a crear programari per a iOS i a testejar-lo sobre el simulador, però no sobre el dispositiu en sí. Per poder fer això cal fer un segon registre al programa de desenvolupadors d'iOS [14], que té un cost de 99$ a l'any si contractem l'opció de programador individual, també existeixen opcions per a empreses [62]. Aquest registre, a més, ens dóna l'opció de poder publicar el nostre programari a la seva botiga virtual, l'App Store [10]. Cal dir que el contingut que vulguem publicar a l'App Store primer haurà de ser validat per Apple, per veure si compleix amb tots els requeriments indicats a les seves normes de publicació. Aquestes normes només són accessibles des de la web d'Apple si estem afiliats al programa [12]. Si l'aplicació no és acceptada el desenvolupador rep una notificació amb els motius per a que pugui subsanar els errors.

Existeix una versió de SDK per a cada una de les versions d'iOS. Tal com s'ha comentat en aquest projecte s'ha usat la versió 3.0 d'iOS, per tant també usarem la versió 3.0 de l'SDK.

3.4 Llibreries multi-plataforma

L'objectiu d'aquest projecte és portar un joc multi-plataforma, per tant es buscarà un joc que estigui implementat d'una manera fàcilment portable i que usi llibreries també portables i preferiblement disponibles a la nostra plataforma. A l'hora de buscar el joc ens hem centrat en que estigués implementat amb una llibreria gràfica que no sigui depenent del hardware. Existeixen llibreries que permeten tractar el hardware de forma transparent, de la mateixa manera sigui quina sigui la màquina en la que es treballa. Les eines lliures multi-plataforma més conegudes són les Simple DirectMedia Layer (SDL) [109]. Existeixen altres alternatives com Allegro [6], però no estan presents a tantes plataformes com SDL. En aquesta secció farem un breu estudi de SDL per conèixer una mica més la llibreria amb la que treballarem.

El nostre joc també usa una altra llibreria multi-plataforma, la libxml2, que s'explica després de SDL, a la secció 3.4.2.

3.4.1 Simple DirectMedia Layer

SDL és una llibreria multimèdia, multi-plataforma i lliure, sota llicència LGPL [51]. Presenta una interfície senzilla per accedir a gràfics, só i dispositius d'entrada de vàries plataformes. S'usa majoritàriament per escriure videojocs o altres aplicacions multimèdia que puguin ser executats en moltes plataformes diferents. El seu nom fa referència a aquesta capa d'abstracció multimèdia simple. Es va començar a desenvolupar l'any 1998 per Sam Lantinga [110].

Per exemple, utilitzant SDL l'àudio es programarà igual per a un PC amb GNU/Linux que per a una Nintendo DS o un iPhone. Això és perfecte per al nostre projecte, perquè haurem de fer un mínim de modificacions al codi del videojoc a l'hora de portar-lo.

La llibreria té per propòsit proporcionar accés transparent a baix nivell de diversos dispositius hardware, com per exemple teclat, ratolí, tarja de só, tarja de vídeo (framebuffer 2D nativament, 3D amb extensions d'OpenGL).

SDL està disponible de manera oficial per GNU/Linux, Windows, Windows CE, BeOS, Mac OS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. El codi també conté suport per AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS, and OS/2, encara que aquests no són suportats oficialment. Per últim, existeixen ports de tercers per a plataformes com Nintendo DS, iPhone, PlayStation 2, PlayStation Portable, GP32, GP2X, entre d'altres.

SDL està escrita en C, però funciona nativament amb C++, i té interfícies a diversos llenguatges, incloent Ada, C#, D, Eiffel, Erlang, Euphoria, Guile, Haskell, Java, Lisp, Lua, ML, Objective C, Pascal, Perl, PHP, Pike, Pliant, Python, Ruby, Smalltalk, i Tcl.

La llibreria principal es divideix en diversos subsistemes, concretament els subsistemes de vídeo (tant funcions bàsiques com les de OpenGL), àudio, CD-ROM, joystick, events, threads i rellotge. A més, també existeixen altres mòduls que ofereixen funcionalitats addicionals. Aquests es proporcionen a la web oficial i s'inclouen en la documentació oficial:

SDL_image estén les capacitats per treballar amb diferents formats d'imatge (BMP, JPEG, PNG...)

SDL_mixer estén les capacitats per la gestió i ús de só i música, com per exemple barreja de só. Suporta formats com Wave, MP3, MOD, S3M...

SDL_net proporciona funcions i tipus de dades multi-plataforma per aplicacions que treballin amb la xarxa.

SDL_ttf suport per renderitzat de fonts TrueType

SDL_rtf suport per renderitzar arxius de text amb format Rich Text Format (RTF)

En aquest projecte usarem SDL per fer dibuix 2D, carregar imatges, gestionar events, reproduir àudio i per funcions de xarxa. S'ha treballat amb els mòduls SDL_mixer i SDL_net, a més de la llibreria bàsica.

Actualment existeixen dues versions de SDL, la 1.2 que és la estable, i la 1.3 que encara està en desenvolupament. SDL 1.3 suposa una actualització important del codi base de la SDL 1.2. Substitueix diverses parts de l'API de la versió 1.2 oferint un suport més general per a l'entrada múltiple i opcions de sortida. Algunes novetats a destacar són el suport per a múltiples finestres, suport de múltiples dispositius d'entrada (varis ratolins, teclats, etc), acceleració hardware 3D o suport per a OpenGL 3.0+. També introdueix una nova opció de llicència, a part de la LGPL ofereix una llicència comercial per permetre que codis tancats l'enllacin estàticament. Això obre possibilitats de distribució de la llibreria a plataformes que no permeten l'enllaçat dinàmic. Galaxy Gameworks s'encarrega de la gestió d'aquesta nova llicència [42].

3.4.2 libxml2

La llibreria libxml2 [77] serveix per analitzar documents XML [129]. Està escrita en llenguatge de programació C i és altament portable ja que només depèn de la llibreria estàndard de C. El número 2 del nom fa referència a que va existir una versió prèvia, actualment en desús, que tenia una interfície de programació diferent.

El joc que hem triat l'usa per carregar un fitxer XML que conté el menú d'opcions.

3.5 Frameworks del sistema: Cocoa Touch

Abans de començar a desenvolupar aplicacions per a iOS hem d'introduir el concepte dels frameworks, però abans necessitem saber què és Cocoa.

Cocoa és un entorn d'aplicació tant per al sistema operatiu Mac OS X com per iOS, el sistema operatiu utilitzat en els dispositius multitàctils d'Apple com l'iPhone, l'iPad, i l'iPod touch (veure secció 2.2). És un conjunt de programes i llibreries dinàmiques que permeten executar programes desenvolupats amb aquestes llibreries. Cocoa està programat en Objective-C i per tant està dissenyat per a que les aplicacions que creem també estiguin escrites en Objective-C o bé comptin amb un connector. Cocoa Touch és la versió específica d'iOS.

A aquests conjunts de llibreries se'ls anomena frameworks. Els podem trobar al sistema com a directoris on el nom té la forma `nom.framework`. A dintre hi trobarem les capçaleres

i el binari de la llibreria, entre d'altres tipus de dades.

Per a la creació de llibreries i aplicacions a iOS haurem d'usar alguns d'aquests frameworks segons les funcionalitats a les que necessitin accedir.

Un dels frameworks més importants a iOS és l'UIKit. Aquest framework proporciona classes i mètodes per a la gestió de la interfície gràfica i per implementar aplicacions basades en events. Realitza la mateixa funció que Application Kit en Mac OS X. Degut a les peculiaritats del sistema d'entrada i sortida de l'iPhone l'Application Kit i el UIKit es diferencien substancialment en que per exemple, aquest últim introdueix "vistes" i la funcionalitat d'"arrossegar i deixar anar" usada en la interfície tàctil.

Altres frameworks de Cocoa Touch, segons les funcionalitats a les que donen suport, són:

Àudio i vídeo: Core Audio, OpenAL, Media Library i AV Foundation.

Manegament de dades: Core Data i SQLite.

Gràfics: Core Animation, OpenGL ES i Quartz 2D.

Xarxa: Bonjour, WebKit i BSD Sockets.

3.6 Eines de compilació

Els entorns de compilació creuada que estem utilitzant es basen en les eines que es descriuen a continuació. Coneixent aquestes eines podem preparar un entorn de compilació creuada nosaltres mateixos en comptes d'usar un toolchain ja prefabricat. Però cal dir que aquest procediment no és trivial i ens caldrien moltes proves per donar amb un entorn funcional. Per crear un entorn de compilació creuada necessitarem les eines Binutils [48], GCC i una llibreria de C.

Binutils És una col·lecció d'eines de programació per a la manipulació d'objectes. Les principals eines són l'enllaçador (`ld`) i l'assemblador (`as`).

GCC El compilador de GNU, del que hem parlat a la introducció d'aquest capítol 3 suporta vàries plataformes i llenguatges de programació, i pot ser configurat com a un compilador creuat. En el moment d'escriure aquestes línies l'última versió estable de GCC és la 4.5.2.

Llibreria de C Necessitarem una implementació de les crides a sistema i funcions que permetin reservar i alliberar memòria o gestionar l'entrada i sortida del programari, com per exemple: open, malloc, free, printf o exit. Existeixen vàries implementacions d'aquesta llibreria. La estàndard és la glibc [49], encara que hi ha versions més reduïdes com la newlib [83] pensades per sistemes encastats.

Unes altres eines a destacar són les autotools (GNU build system), que serveixen per automatitzar la creació dels scripts de configuració de paquets i la compilació. Les eines que conté aquest paquet són autoconf, automake i libtool.

Autotools usen les següents notacions:

build platform Per referir-se a on es compila el codi.

host platform Per referir-se a on s'executa el codi.

target platform Per referir-se als compiladors, es refereix al tipus d'objecte que es produirà, com per exemple, compilar un compilador creuat.

De totes formes, aquestes eines s'han d'usar amb cautela per fer compilació creuada, ja que no són gaire fiables per obtenir les dades d'un entorn diferent al natiu del sistema, i les hem de proporcionar nosaltres.

Generalment el codi font que es distribueix inclou un fitxer de configuració anomenat `configure`, algunes vegades, a través d'aquest fitxer (mitjançant paràmetres) es poden generar `Makefile` per a altres plataformes, però no sempre és així.

Capítol 4

Anàlisi del joc

Per acabar amb la fase d'anàlisi, abans de començar amb la implementació del projecte, necessitàvem un joc que portar a les respectives plataformes. En un primer moment vam pensar en desenvolupar un videojoc nosaltres mateixos, però després vam decidir que aquest no era l'objectiu principal del projecte i vam decidir buscar un joc lliure que s'adaptés a les nostres necessitats, és a dir, que complís les següents condicions:

- Fàcilment portable (preferiblement programat en C/C++ utilitzant les llibreries SDL)
- Lliure
- Opció de joc multi-jugador en xarxa

Després de molt buscar, vam trobar diferents opcions que s'adaptaven a les nostres necessitats, però finalment ens vam decidir per un joc anomenat 2Pong. El vam escollir perquè complia totes les característiques que demanàvem, i per la seva simplicitat.

El 2Pong [1] es tracta d'un clon del mític Pong [90], considerat per molts com el primer videojoc de la història. Pel qui no el conegui, és un joc en dues dimensions que simula un tennis de taula. El jugador controla una paleta que es mou de dalt a baix, i competeix contra una altra paleta, que pot estar controlada per la computadora o per un altre jugador. Els jugadors usen aquestes paletes per fer rebotar una bola d'un costat a l'altre. Cada cop que un jugador deixa passar la bola, fa guanyar un punt a l'adversari.

La peculiaritat del 2Pong és que es juga amb dues boles en comptes d'una sola, d'aquí el seu nom. És un joc molt senzill. Consta d'un menú, en el que es pot escollir jugar un sol jugador contra la intel·ligència artificial (amb tres nivells de dificultat diferents), i el mode multi-jugador en xarxa, en el que el jugador pot escollir entre fer de servidor o de client, i que es comunica via una IP i un port. Pel que fa a l'àudio, no té música d'ambient, només te efectes de só quan rebota la bola. Per últim, remarcar que té uns "powerups", que són

uns objectes modificadors que apareixen a la pantalla i que tenen diversos efectes sobre la bola: augmentar la seva velocitat, disminuir-la, dividir la bola en dues, etc.

El 2Pong està escrit en C++ i utilitza les següents llibreries: SDL, SDL_net, SDL_mixer, iconv, zlib i libxml2. Per tant, si volem compilar el codi font, haurem de preparar el nostre sistema perquè suporti totes aquestes llibreries.

L'última versió és la 1.0.1a i data del maig de 2005, pel que fa bastant temps que el projecte no està en desenvolupament, encara que s'anunciaven millores per a la següent versió. A la web oficial del projecte estan disponibles els binaris de la darrera versió per a GNU/Linux i Windows.

El projecte es distribueix sota llicència GPL [50], que permet la còpia, distribució (comercial o no) i modificació del codi, sempre que qualsevol modificació es continuï distribuint amb la mateixa llicència GPL. La llicència GPL no permet la distribució de programes executables sense el codi font corresponent o la indicació de com obtenir-lo gratuïtament. És important conèixer la llicència donat que nosaltres haurem de fer modificacions al codi, i si les volem publicar ho haurem de fer de la manera que ens marca la llicència.

El seu creador és Itay Kirshenbaum i hem contactat amb ell per tal d'informar-lo dels ports realitzats i de la nostra intenció de publicar el codi. No ens ha posat cap problema i ha mostrat interès pel projecte.

4.1 Algorisme

A continuació es resumeixen les fases per les que passa el joc.

1. Inicialització de SDL
2. Creació de la finestra SDL
3. Càrrega dels sprites
4. Creació del menú XML
   (a) Si es tria **regular** (mode humà vs CPU), elecció de la dificultat
   (b) Si es tria **network** (mode humà vs humà), elecció de client/server
      i. Especificar IP i port a connectar
5. Inici del joc (bucle principal)
   (a) Captura i tractament d'events de SDL
   (b) Actualització de l'estat del joc
   (c) Actualització dades xarxa (en cas d'haver triat mode de xarxa)

   (d) Pintat de l'escena

6. Sortida del joc

A la figura 4.1 es mostra l'esquema del menú del 2Pong. S'indiquen les opcions que cal triar per a efectuar les transicions entre pantalles.

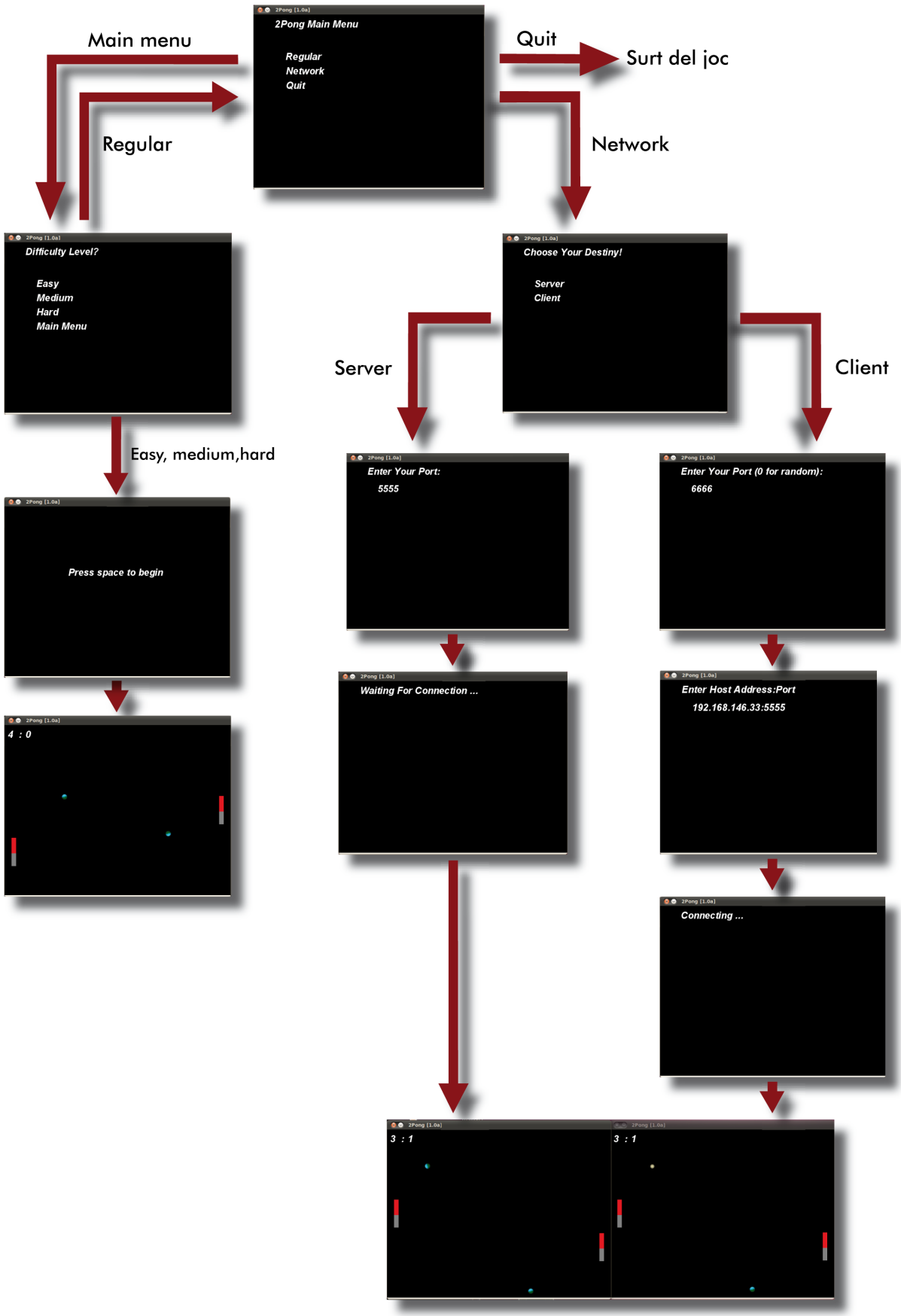


Figura 4.1: Mapa dels menús del 2Pong.

Capítol 5

Implementació

Després de tota la fase d'estudi previ, ja estem preparats per realitzar la implementació del projecte.

Aquest capítol reflecteix tot el treball fet per nosaltres i en ell es detalla tota la programació i els experiments realitzats, de manera que aquests procediments puguin ser reproduïts pel lector.

Hem dividit procés en diverses fases. Primer s'aplicarà el jailbreak a l'iPhone 3G, després instal·larem els diferents entorns de desenvolupament i en provarem el seu funcionament. Després passarem a adaptar el codi tant del joc com de les llibreries que necessitem compilar i en farem la depuració. També s'explicarà com instal·lar una aplicació a l'iPhone.

5.1 Alliberament del dispositiu: Jailbreak

El terme anglès jailbreak significa fuga o literalment "trencar la presó". S'usa aquest terme ja que filosòficament Apple aplica una presó al sistema de fitxers de l'iPhone no sent accessible per l'usuari i vetant-li els permisos d'administrador.

Un cop fet el jailbreak l'usuari pot accedir al sistema de fitxers del SO amb permisos de *root* [117], i així aconseguir instal·lar aplicacions sense passar per l'aprovació d'Apple.

El procés consisteix en manipular el firmware subministrat per Apple per eliminar aquestes limitacions. Des de l'aparició de l'iPhone la comunitat de desenvolupadors s'ha dedicat a buscar *exploits* [37] per poder fer aquest jailbreak i permetre un accés total al dispositiu a tots els usuaris.

Aplicar el jailbreak al nostre iPhone és legal a països com Espanya o els Estats Units. Aquest procés ens permet usar aplicacions *crackejades* [113], però això sí que és il·legal i és responsabilitat de l'usuari.

Quan fem un jailbreak perdem la garantia de l'iPhone però això és fàcilment solucionable

restaurant el dispositiu amb un firmware original a través d'iTunes.

Per tal de modificar el firmware haurem d'usar una eina que ho faci per nosaltres aplicant les opcions que vulguem i després instal·lar aquest nou firmware al nostre dispositiu. En aquest projecte s'ha usat l'eina PwnageTool 3.0 [94] que només està disponible per Mac OS X. És un programa amb llicència freeware [40] desenvolupat per iPhone Dev Team [66,67].

A l'inici d'aquest projecte no hi havia cap eina d'aquest tipus que fos programari lliure per tal de complir amb les premisses del projecte. Al final es va optar per aquesta eina perquè era la que disposava de més informació i tutorials a la xarxa. També hi ha altres programes disponibles per a GNU/Linux i Windows, com Greenpois0n [54] o Sn0wbreeze [112].

Si el nostre iPhone té una versió del firmware inferior a la 3.0 no tindrem cap problema. Però si és superior haurem de fer un *downgrade* (baixada de versió). Això és degut a que Apple no permet instal·lar un firmware en un dispositiu iOS si aquest és una versió inferior a la instal·lada. Existeixen varis programes per fer aquest downgrade i multitud de tutorials a la xarxa [32]. Aquest procediment s'usa quan volem fer el jailbreak al nostre dispositiu iOS però encara no existeix la possibilitat per fer-ho a la nostra versió de firmware. Per tant haurem de canviar a una versió on el jailbreak estigui disponible. A hores d'ara la darrera versió del firmware per a iPhone 3G és la 4.2 i ja se li pot aplicar el jailbreak.

Cal dir que el jailbreak és diferent al procés d'alliberament de la tarja SIM [116], que ens permet usar una xarxa telefònica diferent a la prèviament contractada. Existeixen eines per desbloquejar la SIM de l'iPhone, però aquest tema no es tractarà, donat que no influeix en el nostre PFC.

5.1.1 Passos a seguir

A continuació es detallen els passos per a fer el jailbreak al nostre iPhone 3G. Existeixen multitud de tutorials a la xarxa que ho expliquen, tot i així volem explicar-ho en aquest document ja que en molts tutorials no es descriuen les opcions que ens trobem durant el procés i no es justifica la seva elecció o omissió.

1. Descarreguem la darrera versió d'iTunes per a Mac OS X i la instal·lem [73]
2. Baixem el firmware 3.0 [69] de l'iPhone, que és un arxiu amb l'extensió *.ipsw* que són les sigles d'*iPhone Software*.
3. Ens descarreguem el PwnageTool 3.0. L'arxiu no es troba disponible per descàrrega directa a la seva web, però ens apunten a una web de torrents [21] des d'on el podem baixar [95]. Triem `PwnageTool_3.0.dmg`. L'extensió *dmg* [17] indica que és una imatge de disc, una mena d'unitat virtual.
   (a) Fem doble click a aquest arxiu i se'ns muntarà la imatge com a un nou volum que pareixerà a l'escriptori.

(b) Fem doble click al volum per tal de veure el contingut.

(c) Extraiem el PwnageTool a una altra ubicació del nostre Mac.

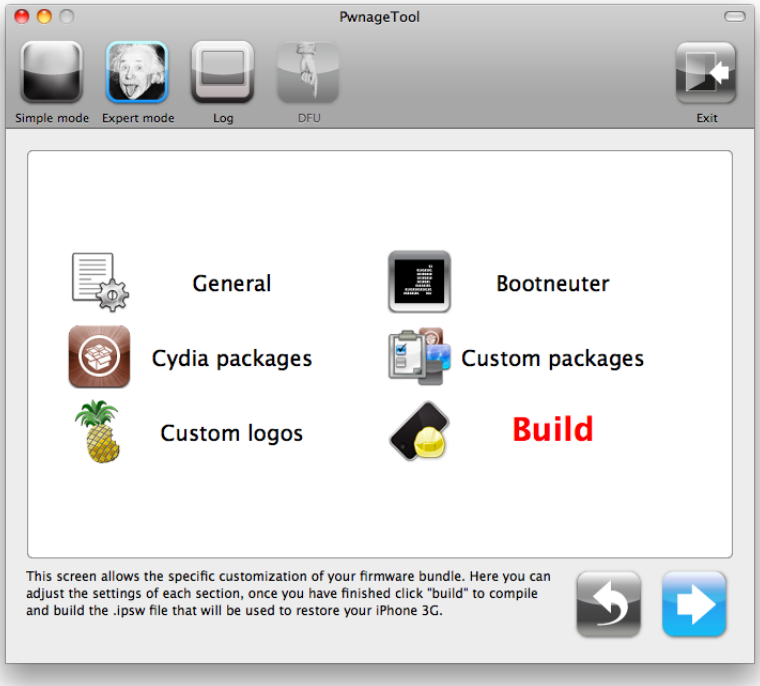
4. Abans de començar amb la modificació del firmware de l'iPhone haurem de fer una còpia de seguretat del nostre sistema, incloses les dades, ja que quan instal·lem el nou firmware tota aquesta informació es perdrà. La còpia la farem des d'iTunes i la podrem carregar de nou posteriorment. Per fer-la obrim l'iTunes i connectem l'iPhone per usb al PC. L'iTunes el detectarà i ens apareixerà a la zona esquerra de la pantalla. Per iniciar la còpia fem click amb el botó dret del ratolí sobre la icona de l'iPhone i triem "Guardar copia de seguridad".

5. Llancem l'aplicació PwnageTool, acceptem l'advertiment i triem **Expert mode** per tal de poder canviar vàries configuracions. Si triéssim *Simple mode* se'ns crearia el firmware modificat amb les opcions per omissió.

6. Triem el nostre model d'iPhone, en aquest cas iPhone 3G, i clickem a la fletxa blava per continuar.

7. En aquest pas el programa intenta trobar el firmware 3.0 original, que posteriorment modificarà. Si no el troba ens deixa navegar el sistema de fitxers per indicar-li la ubicació.

8. Ara ens trobem amb un menú amb 6 opcions. Triem **General** i clickem a la fletxa blava. En aquesta opció podrem establir la mida de la partició root del sistema.



9. A la següent pantalla ens trobem amb tres caselles.

   Activate the Phone Permet precarregar uns ajustaments d'operadora telefònica sense haver de tenir una targeta SIM d'aquesta companyia instal·lada. Aquesta opció era especialment útil quan l'iPhone no es trobava disponible a tots els territoris. L'elecció queda en mans de l'usuari.

   Enable baseband update Permet l'actualització de la part del firmware de l'iPhone que s'encarrega de la part telefònica. Aquesta opció no és rellevant en el nostre projecte però hem de tenir en compte que si el nostre dispositiu no és lliure de fàbrica podem perdre la possibilitat d'usar qualsevol operador de telefonia.

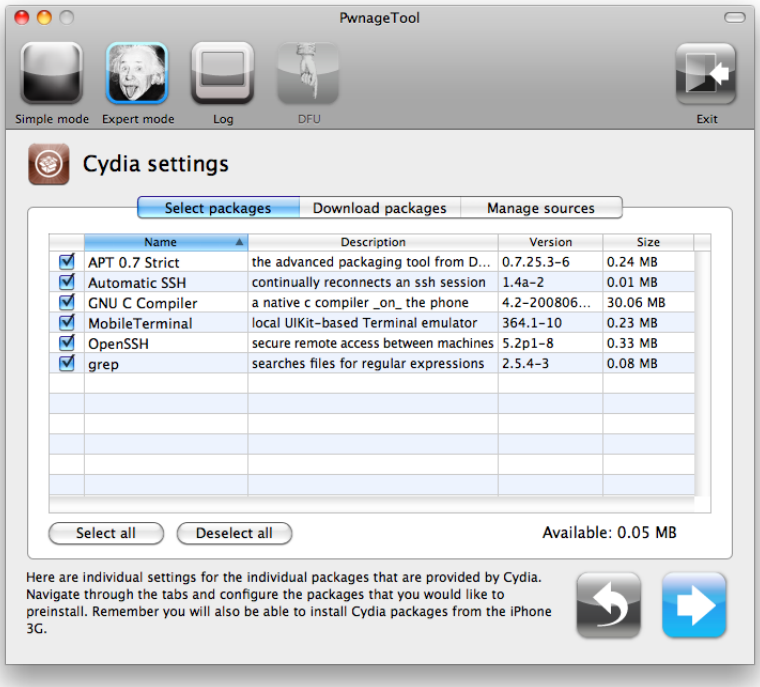
   Disable partition wipe-out Aquesta opció actualment no és seleccionable, ens permetria esborrar el sistema de fitxers arrel.

   Amb la darrera opció podem redimensionar la partició del super usuari de Unix. La deixem com està. Continuem amb la fletxa blava.

10. Passem sense modificar la secció **Bootneuter settings**.

11. A la secció de **Cydia settings** podem triar quines aplicacions volem instal·lar al nostre iPhone per no haver-ho de fer posteriorment des del dispositiu. Instal·lem les següents:

- **OpenSSH** [88] Servidor ssh per poder treballar sobre l'iPhone remotament.
- **Automatic SSH** Manté oberta la sessió ssh. Molt útil per quan fem feina durant força estona i no volem reconnectar constantment.
- **Link Identity Editor (ldid)** Aplicació per a signar les aplicacions, veure la secció 5.6.1 per a més informació.
- **Mobile Terminal** [81] Un emulador de terminal.
- **grep** [55] Un cercador d'expressions regulars al sistema.
- **libxml2** [76] Una llibreria per a analitzar documents XML que necessitarem per al nostre joc. Per a més informació consultar la secció 3.4.2.
- **APT 0.7 Strict** [5] Sistema de gestió de paquets de Debian.



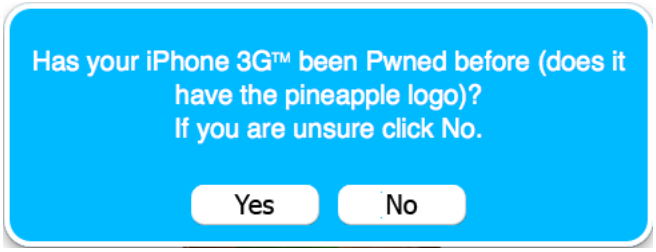
12. A la següent pantalla només seleccionem **Cydia** [29]. Aquesta aplicació és un gestor de paquets que permet instal·lar-nos aplicacions no regulades per Apple i que no es troben a la seva botiga, l'App Store. La instal·lació dels paquets es fa via *APT* i *dpkg* [33]. Icy es un altre gestor que pretenia ser compatible amb els paquets de Cydia però el projecte finalment va ser tancat a l'octubre de 2009, per tant no cal que la instal·lem.

13. Al següent pas podem triar un logo per canviar-lo per la clàssica poma d'Apple a l'iniciar l'iPhone. Obviem aquesta opció clickant a la fletxa blava.

14. Triem l'opció **Build**. PwnageTool comença a crear el nostre firmware modificat. Aquest procés pot trigar uns 15 minuts.

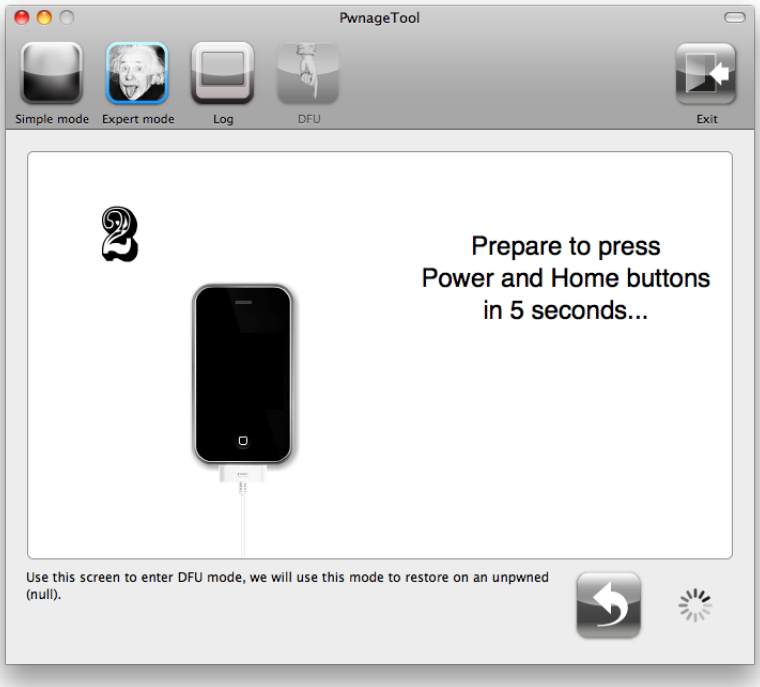
15. Un cop finalitzat guardem el nou firmware de manera que el puguem diferenciar de l'original, per exemple triem *iPhone1,2_3.0_7A341_Custom_Restore.ipsw*.

16. Quan el programa ens demani si el nostre iPhone ha estat **Pwned** alguna vegada diem **no**. Això farà que accedim a un tutorial per posar l'iPhone en mode DFU o recuperació. Aquest pas és necessari perquè serà el que ens permetrà que l'iTunes instal·li el nou firmware al nostre iPhone.



17. Tal com se'ns indica connectem nostre iPhone amb el cable USB al Mac i l'apaguem.

18. Aquest pas és el més enrevessat. Ara haurem de seguir les indicacions de pantalla per posar l'iPhone en mode DFU. Hem de prémer el botó **home** i el botó **power** durant 10 segons, i després deixar anar el botó power i seguir prement el botó home. És provable que no ens surti a la primera, el programa ens repetirà els passos tantes vegades com calgui. Si volem realitzar aquest procediment més endavant hi podem accedir clickant a l'opció DFU del menú principal de PwnageTool.



19. Quan ho aconseguim se'ns informa per pantalla. Tanquem el PwnageTool i obrim l'iTunes.

20. L'iTunes ens informa que ha detectat un iPhone en mode recuperació. Clickem a **Aceptar**.

21. Mentre mantenim premuda la tecla Alt/Option del Mac clickem a **Restaurar**. Això obrirà una finestra per triar el firmware que volem usar per a la restauració. Triem el firmware que acabem de crear.

22. L'iTunes restaurarà l'iPhone amb el firmware modificat. Aquest procés també pot durar uns 15 minuts.

23. Una vegada completada la restauració se'ns demanarà que configurem l'iPhone, bé com a un dispositiu nou o bé fent servir un backup.

24. Triem la segona opció i restaurem la còpia de seguretat que hem fet al punt 4.

25. Reiniciem l'iPhone i el jailbreak ja s'ha completat. Si tot ha anat bé trobarem aquestes dues aplicacions a la pantalla principal de l'iPhone.

5.2 Instal·lació dels entorns de desenvolupament

En aquesta secció explicarem els passos que hem de dur a terme per a la instal·lació dels tres entorns de desenvolupament que hem explicat al capítol 3 així com les proves que s'han dut a terme per verificar el seu funcionament correcte.

5.2.1 iphonedevonlinux

Tal com s'ha comentat a la secció 3.1 finalment es va decidir usar el toolchain lliure **iphonedevonlinux**. A la seva web de Google code [71] podem trobar les instruccions per a la seva instal·lació, que es detallen a continuació.

El toolchain està preparat per ser usat sobre Debian GNU/Linux o Ubuntu GNU/Linux. Per a la seva instal·lació usarem eines disponibles en aquestes distribucions, tant si usem un sistema de 32 bits com de 64.

Obtenir els arxius

Primer creem un directori on descarregar els arxius necessaris per a preparar l'entorn i els baixem a través del *svn* [8].

```
mkdir -p ~/Projects/iphone/toolchain
cd ~/Projects/iphone/toolchain
svn checkout http://iphonedevonlinux.googlecode.com/svn/trunk/ ./
```

Ens haurem de baixar l'SDK i el firmware 3.0 de l'iPhone. L'adreça [69] des d'on ens podem descarregar el firmware també la podem trobar a l'arxiu `firmware.lst` que trobarem a `/Projects/iphone/toolchain`. Encara que ja hauríem de disposar d'ell si hem realitzat el jailbreak.

Donat que a la pàgina web d'Apple només està disponible la darrera versió de l'SDK, en el moment d'escriure aquestes línies la versió 4.2 [63], ens haurem de baixar la 3.0 usant un torrent o una descàrrega directa. Existeixen diverses fonts i les podem trobar cercant "iphone_sdk_3.0__leopard__9m2736__final.dmg download".

Hem de copiar els arxius de l'SDK i el firmware als següents directoris del toolchain:

```
cd ~/Projects/iphone/toolchain
mkdir -p files/firmware
mv /path/to/iphone_sdk_3.0__leopard__9m2736__final.dmg files/
mv /path/to/iPhone1,2_3.0_7A341_Restore.ipsw files/firmware
```

Paquets necessaris per a compilar el toolchain

Abans de compilar el toolchain necessitarem un seguit d'eines instal·lades al nostre sistema.

Si el nostre sistema és de 32 bits usarem la següent comanda:

```
apt-get install \
  automake \
  bison \
  cpio \
  flex \
  g++ \
  g++-4.3 \
  g++-4.3-multilib \
  gawk \
  gcc-4.3 \
  git-core \
  gobjc-4.3 \
  gzip \
  libbz2-dev \
  libcurl4-openssl-dev \
  libssl-dev  \
  make \
  mount \
  subversion \
  sudo \
  tar \
  unzip \
  uuid \
  uuid-dev \
  wget \
  xar \
  zlib1g-dev \
```

L'autor comenta que va tenir problemes instal·lant el toolchain amb gcc-4.3 a una màquina Debian amb arquitectura amd64. Per tant per a la instal·lació sobre una màquina de 64 bits recomana usar els compiladors amb la versió 4.3-multilib. La diferència entre una versió i l'altre és que la multilib suporta altres arquitectures [43, 44]. Per tant de la llista d'eines a instal·lar haurem de canviar les versions normals per les multilib, instal·lant els següents paquets:

```
apt-get install g++-4.3-multilib gcc-4.3-multilib gobjc-4.3-multilib
```

Posada en marxa i compilació

Entre els arxius que ens hem descarregat al principi hi trobem un script que haurem d'executar per tal de compilar el toolchain. Abans d'executar-lo podem configurar un seguit de variables.

BUILD_DIR: Directori on es guardaran els binaris (gcc, otool, etc).

Per omissió s'usa: `$TOOLCHAIN/bld`

PREFIX: Es creen els directoris ./bin i ./lib per als executables sota aquest prefix.

Per omissió s'usa: `$TOOLCHAIN/pre`

SRC_DIR: Directori on es guarda el codi font de les eines (gcc, etc).

Per omissió s'usa: `$TOOLCHAIN/src`

SYS_DIR: Es posen els arxius de sistema de l'iPhone sota aquest directori.

Per omissió s'usa: `$TOOLCHAIN/sys`

Si les volem modificar usarem per exemple:

```
sudo BUILD_DIR="/tmp/bld" SRC_DIR="/tmp/src" PREFIX="/usr/local" SYS_DIR
   =/usr/local/iphone_sdk_3.0 ./toolchain.sh all
```

Per córrer l'script, primer li donem permisos d'execució i l'executem amb el flag **all**.

```
chmod u+x toolchain.sh
./toolchain.sh all
```

Testejar el toolchain

El toolchain conté una aplicació d'exemple, el típic Hello World, que consisteix en mostrar a la pantalla de l'iPhone un missatge simple. En aquest cas es mostra "Hello Toolchain", i el missatge pot ser modificat tocant la pantalla de l'iPhone, on apareix el teclat virtual.

Per a compilar-lo només haurem d'executar les següents comandes.

```
cd ~/Projects/iphone/toolchain/apps/HelloToolchain
make
```

Per tal d'executar-la a l'iPhone seguirem els passos descrits per a iphonedevonlinux a la secció 5.6.

5.2.2 SDK oficial

Instal·lar l'SDK oficial d'Apple és molt senzill. Tal com s'ha explicat a la secció 3.3 per poder descarregar-nos el darrer SDK ens hem de registrar com a desenvolupadors d'Apple [15].

Donat que en aquest projecte s'ha usat la versió 3.0 de l'SDK 3.0 i aquesta ja no es troba disponible per descarregar des de la web d'Apple l'haurem de cercar a la xarxa. Tal com

hem comentat a la secció d'instal·lació d'iphonedevonlinux el podem trobar en torrent o descàrrega directa. Haurem de fer una cerca usant "iphone_sdk_3.0__leopard__9m2736__final.dmg download".

Un cop descarregat fem doble click a l'arxiu i se'ns muntarà la imatge com a un nou volum que pareixerà a l'escriptori. Dintre del volum trobarem un instal·lador. Ara només haurem de seguir les instruccions d'aquest per a completar la instal·lació. Un cop finalitzada tindrem al directori `/Developer` del nostre sistema tot el necessari per al desenvolupament. A `/Developer/Applications` trobarem eines com Xcode o Property List Editor. I a `/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS3.0.sdk` l'SDK 3.0 amb totes les llibreries per al firmware 3.0.

Testejar l'SDK oficial

Per testejar el correcte funcionament de l'entorn de desenvolupament s'ha intentat compilar el mateix exemple usat per a testejar el toolchain iphonedevonlinux. Per tant abans de començar amb el test haurem de disposar del codi font de l'exemple al nostre Mac. El podem descarregar tal com s'explica a la secció 5.2.1.

Per a la compilació del codi podem usar l'Xcode, l'IDE per omissió que inclou l'SDK. Donat que no teníem experiència en l'ús de l'Xcode vam decidir començar usant una plantilla que proporciona el mateix programa. Seguirem els passos que es descriuen a continuació. Provarem de compilar el codi per a que corri a l'iPhone i també en el simulador que inclou l'SDK oficial.

1. Iniciem l'Xcode des de `/Developer/Applications/Xcode`.

2. A la pantalla inicial triem "Create a new Xcode project".

3. En aquest punt podem triar per a quina plataforma volem desenvolupar la nostra aplicació i quina plantilla volem usar. A la part esquerra de la pantalla podem triar si volem desenvolupar una aplicació per iOS (iPhone OS) o bé per Mac OS X. Triem una aplicació d'iOS. D'entre les plantilles possibles triarem "View-based Application", una de les més simples, que encara que no s'adapta a la perfecció a la nostra aplicació, ens servirà de guia.

4. Triem un nom per al nostre projecte, per exemple "HelloToolchain" i el guardem.

5. Veurem que l'Xcode ha creat dos blocs importants, per una part el projecte en sí, on ha creat una estructura de directoris que es divideixen en classes, altres fonts, recursos, frameworks, i productes. I per l'altre, un directori anomenat *Targets*, o objectius.

   Els arxius que trobem al projecte són plantilles per programar la nostra aplicació, però donat que ja disposem del codi font en uns altres arxius els haurem d'eliminar

i substituir-los pels nostres. A productes veurem que hi ha un arxiu en vermell anomenat "HelloToolchain.app", que serà l'aplicació que obtindrem després de la compilació del projecte. Per veure els detalls d'aquest producte ens hem de fixar en l'apartat targets, on podem desglossar HelloToolchain en recursos a copiar al paquet de l'aplicació (imatges, sons, etc), codi font a compilar i frameworks amb els que enllaçar l'executable. Veurem que per omissió ens hi ha distribuït els arxius del projecte.

6. Primer esborrem els arxius innecessaris del projecte, al fer-ho també desapareixeran de la secció target. Podem esborrar el directori classes, altres fonts i al directori recursos només deixar-hi l'arxiu Info.plist. No hem de tocar el directori amb els frameworks. Quan el programa ens demani si volem eliminar només les dependències o esborrar els arxius en sí, triem aquesta, ja que no els necessitarem posteriorment.

7. Ara és el moment d'afegir els nostres arxius. Anem al menú `Project→Add to project...` i triem els dos arxius necessaris per a l'exemple, `HelloToolchain.m` i `HelloToolchain.h` que estan al directori `src` del codi font. Els seleccionem i ens apareixerà la pantalla que es mostra a la figura 5.1. Triem les mateixes opcions que s'indiquen a la imatge. Això farà que els dos arxius que hem seleccionat es copiïn al directori del projecte i s'afegeixin al nostre target. Un cop afegits la nostra finestra d'Xcode hauria de ser semblant a la que es mostra a la figura 5.2.

8. Donat que la plantilla no s'adapta a la nostra aplicació haurem de fer alguns canvis a les preferències del projecte. Obrim la finestra de preferències amb `Project→Edit Active Target ''HelloToolchain''`. Veiem que hi ha un munt d'opcions, però només n'haurem de modificar algunes. Si es vol aprofundir en l'estudi de les opcions es recomana la lectura d'aquest recurs [128].

   Al submenú "Code Signing" triem **Don't Code Sign** en comptes d'iPhone Developer, això farà que les nostres aplicacions no es signin i les puguem usar en el nostre dispositiu. Per a més informació consultar la secció 5.6.1.

   Per últim al submenú "GCC-4.2 Language" modificarem opcions pròpies del compilador. Haurem de desmarcar l'opció "Precompile prefix Header" i esborrar l'arxiu referenciat a "Prefix Header", ja que ja no existeix.

9. També haurem de modificar l'arxiu Info.plist eliminant la darrera clau (Main nib file base name) ja que no ens interessa usar un arxiu que defineixi una finestra. La creació de la finestra de l'aplicació ja està definida al codi font de l'exemple.

10. Abans de compilar triarem l'SDK pel que volem crear l'aplicació. A través del menú desplegable de l'esquerra primer triarem **Release** com a "Active Configuration" ja que no hi volem afegir símbols per a la depuració. I com a "Active SDK", primer triarem **iPhone Device 3.0** per a la primera compilació i després el canviarem a **iPhone Simulator 3.0** per obtenir la versió per al simulador.

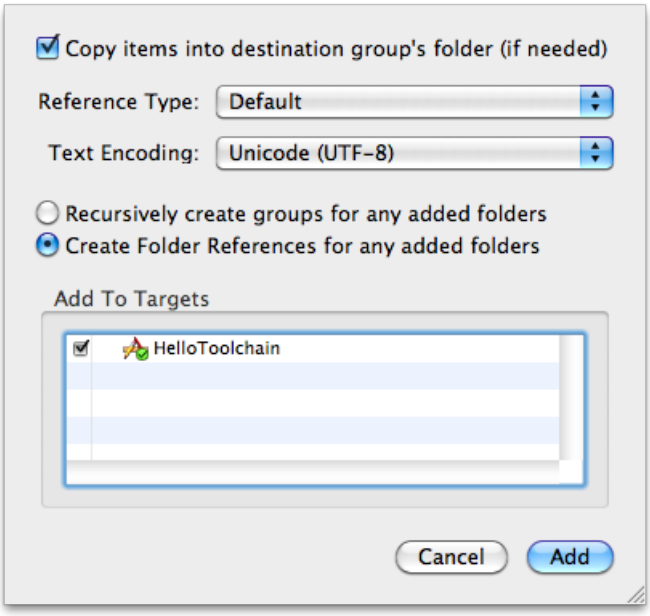


Figura 5.1: Afegint arxius al nostre project d'Xcode.

11. El darrer pas que ens queda és compilar i enllaçar l'aplicació, només ens caldrà clickar al menú `Build`→`Build Results` i se'ns informarà si ha hagut cap problema durant el procés.

L'Xcode desa les aplicacions generades al directori `~/HelloToolchain/build` organitzades segons la versió de l'SDK utilitzat. Per tant podrem trobar l'aplicació per a iPhone a `~/HelloToolchain/build/Release-iphoneos` i la versió per al simulador a `~/HelloToolchain/build/Release-iphonesimulator`.

La versió per al simulador la podem testejar des del mateix Xcode clickant al botó Build and Run que trobarem a la part de dalt de la finestra de l'Xcode. Si tot ha anat bé hauríem de veure la imatge que es mostra a la figura 5.3.

Per a executar la versió d'iPhone haurem de seguir les instruccions descrites a la secció 5.6.

5.2.3 Scratchbox2

En aquest punt (si el lector ha avançat seqüencialment) hem vist com instal·lar un entorn de desenvolupament lliure i un de privatiu. Però ja hem dit diverses vegades que experimentaríem amb altres tipus d'entorns de desenvolupament de compilació creuada. El nostre propòsit inicial era provar entorns diferents per estudiar-los i aprofitar-nos dels seus avantatges. A la secció 3.2 vam dir que experimentaríem amb Scratchbox2, i encara que finalment no s'ha utilitzat per un tema de comoditat, (personalment m'agradava més el

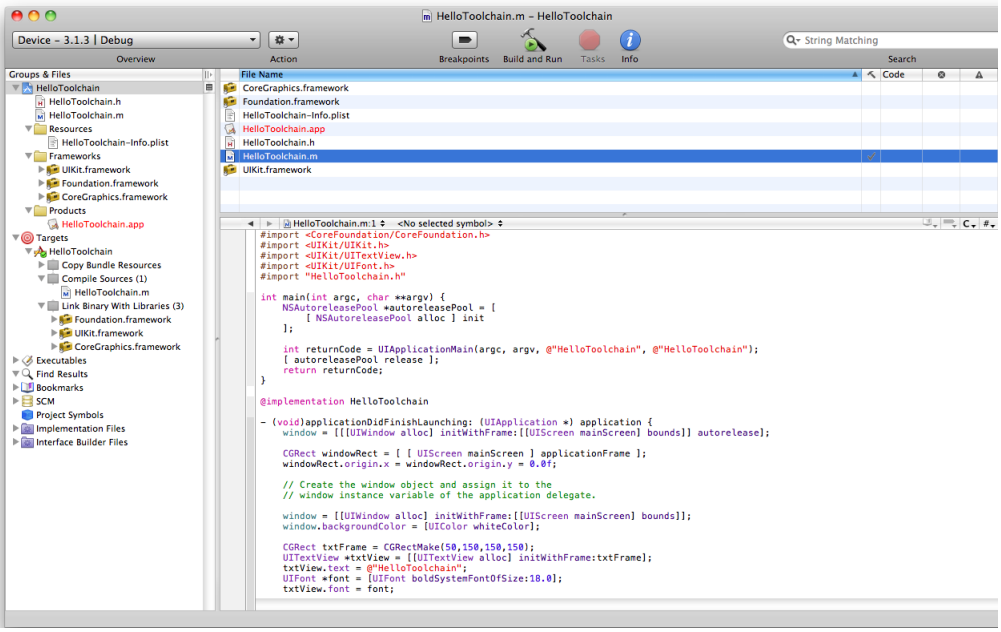


Figura 5.2: Xcode amb el projecte HelloToolchain.

Figura 5.3: Simulador de l'iPhone executant HelloToolchain.

Makefile que proveeix iphonedevonlinux), en aquesta secció detallarem com instal·lar-lo i usar-lo.

Hem obtingut la informació principalment d'aquest tutorial [56]. En ell s'usa un toolchain per ARM genèric però nosaltres volem usar el toolchain lliure que hem instal·lat (explicat a la secció 5.2.1), pel que variarem una mica el procediment. Instal·larem Scratchbox2 en sí i QEmu, un emulador d'ARM.

Paquets necessaris

Per a la instal·lació completa necessitarem els següents paquets, la majoria d'ells ja els hem instal·lat durant la instal·lació del toolchain: autoconf, autogen, automake, autotools-dev, binutils, fakeroot, git-core, g++, make i sbrsh.

El paquets específics per a la instal·lació de QEmu són: gcc-3.4, libsdl1.2-dev i subversion.

Si estem en un entorn Debian GNU/Linux o derivats podem usar el gestor de paquets APT (Advanced Packaging Tool) [5] per assegurar-nos de que els tenim tots instal·lats. Cal dir que si ens trobem sobre una altra distribució, haurem d'usar un altre mètode de descàrrega.

```
sudo apt-get install autoconf autogen automake autotools-dev binutils
   fakeroot gcc-3.4 git-core g++ libsdl1.2-dev make sbrsh subversion
```

Crear directoris

Abans de la instal·lació hem de crear la següent estructura de directoris:

- sbox2
  - bin (Contè tots els binaris necessaris per a l'entorn)
    - qemu (Binaris de QEmu)
  - rootfs (Conté els sistemes de fitxers arrels per a diferents arquitectures)
    - iphone (Sistema de fitxers arrel per a iPhone)
  - src (Codi font per a Scratchbox2)

Per crear els directoris executem:

```
mkdir -p ~/sbox2 ~/sbox2/bin/qemu ~/sbox2/rootfs/iphone ~/sbox2/src
```

Crear el sistema de fitxers arrel

El directori ~/sbox2/rootfs/iphone ha de contenir el sistema de fitxers arrel del nostre iPhone. Si hem realitzat la instal·lació del toolchain tal com es detalla a la secció 5.2.1 ja el tindrem a $TOOLCHAIN/sys pel que l'únic que hem de fer és copiar-lo d'allà.

```
cp -r $TOOLCHAIN/sys ~/sbox2/rootfs/iphone
```

Usar toolchain

Per a que Scratchbox2 pugui usar el nostre toolchain primer hem d'incloure la localització del nostre compilador creuat al PATH del sistema. Per a que el canvi sigui permanent editem l'arxiu .bashrc que es troba a la nostra home i hi afegim la següent línia. Si no tenim l'arxiu el creem. La variable $TOOLCHAIN, tal com s'ha explicat a la secció anterior, representa el path on trobem el nostre toolchain i a $TOOLCHAIN/pre/bin hi trobem els compiladors creuats per a iPhone.

```
vim $HOME/.bashrc
export PATH=$PATH:$TOOLCHAIN/pre/bin
```

Tanquem la consola on hem editat l'arxiu i n'obrim una de nova per tal que s'apliqui el canvi.

Scratchbox2

Per a instal·lar Scratchbox2 ens descarreguem la darrera versió del codi font i la compilem usant les comandes:

```
cd ~/sbox2/src
git clone git://gitorious.org/scratchbox2/scratchbox2.git sbox2
cd sbox2
git checkout 2.1 -b devel_env
./autogen.sh
./configure --prefix=$HOME/sbox2/bin/scratchbox
make install
```

Per finalitzar hem d'afegir algunes opcions de configuració: necessitem definir l'arquitectura del sistema host. Per fer-ho, s'ha d'editar el fitxer:

~/sbox2/bin/scratchbox/bin/sb2-build-libtool

I buscar una línia que executa el script de configuració i afegir-hi el sistema host de la forma --host=i386. En el nostre cas la línia queda de la següent manera:

```
./configure --prefix=$HOME/.scratchbox2/$TARGET --build=$(uname -m)-
   unknown-linux-gnu --host=i386
```

QEmu

Una part de l'entorn és l'emulador, usarem QEmu. Si ja disposem de QEmu al nostre sistema l'haurem d'eliminar usant la comanda:

```
sudo apt-get --purge remove qemu
```

Usem Git per descarregar el codi font:

```
cd ~/sbox2/src
git clone git://git.savannah.nongnu.org/qemu.git
cd qemu
git checkout v0.13.0-rc1 -b devel_env
```

Ara ja podem compilar el codi de QEmu:

```
./configure --prefix=$HOME/sbox2/bin/qemu
make
make install
```

Ús

Abans de poder usar Scratchbox2 ens situem al directori on hem copiat el sistema de fitxers de l'iPhone i inicialitzem allà Scratchbox2 indicant quin compilador creuat usarem, en el nostre cas el `arm-apple-darwin9-gcc`. Això farà que a l'iniciar Scratchbox2 es crei un entorn chroot en aquest directori i quan usem la comanda gcc aquesta apunti a `arm-apple-darwin9-gcc`.

```
cd ~/sbox2/rootfs/iphone
sb2-init iphone $TOOLCHAIN/pre/bin/arm-apple-darwin9-gcc
```

Per usar Scratchbox2 hem d'usar la comanda `sb2`. L'executem a una consola i veurem que l'aspecte del prompt canvia:

```
user@laptop:~/sbox2$ sb2
[SB2 simple iphone] user@laptop iphone $
```

Podem fer una petita prova de que pot compilar i enllaçar:

```
echo 'int main() { return 0; }' > test.c
gcc test.c
file a.out
a.out: ELF 32-bit LSB executable, ARM, version 1 (SYSV), statically
   linked, not stripped
```

Un cop ja hem provat Scratchbox2 ja hem acomplert l'objectiu d'experimentar amb altres entorns de compilació creuada. Tot i que ens ofereix avantatges com un entorn segur de desenvolupament, ens sembla menys còmode que altres mètodes de compilació, pel que no l'hem usat normalment. Però sempre és bo tenir coneixement d'aquest tipus d'eines per si en volem fer ús més endavant.

5.3 Adaptació del codi

Un cop tenim instal·lat els entorns de desenvolupament ja podem començar a fer els ports de les llibreries i el joc. Però abans haurem de preparar els codis per a que puguin ser executats a l'iPhone.

En aquesta secció s'expliquen els canvis que s'han dut a terme per adaptar tant el codi de les llibreries com del joc a l'iPhone 3G.

5.3.1 SDL

Tal com hem introduït a la secció 3.4.1 una part indispensable d'aquest projecte és l'ús de la llibreria SDL. A l'iPhone usarem la versió 1.3, encara en desenvolupament. Tot i que no és una versió estable es va decidir usar-la ja que existeix un port oficial per a iPhone. Encara que s'inclou de forma oficial al codi font de la versió 1.3 correspon a un port realitzat durant el Google Summer of Code [52] 2008 (GSoC2008). Aquest port inclou gairebé totes les funcionalitats necessàries per adaptar el nostre joc.

Entrada tàctil: Donat que el mètode d'entrada bàsic de l'iPhone són les pulsacions a la seva pantalla multitàctil aquesta característica és imprescindible. Sense ella no hauríem pogut accedir a cap mètode de control en el nostre joc. Aquest mètode d'entrada al port de l'iPhone es representa com la d'un ratolí, un mètode d'entrada ja implementat a SDL. Arrossegar el dit per la pantalla representa passar-hi el cursor, i una pulsació amb el dit representa un click. Ja que SDL 1.3 inclou suport per a múltiples ratolins podem aprofitar la pantalla multitàctil de manera que podem usar gestos multitàctils, això vol dir que podrem usar més d'un dit a la vegada i cada un representarà un ratolí diferent.

Acceleròmetre: Tal com hem comentat l'iPhone inclou un acceleròmetre. És molt habitual que les aplicacions per a iPhone facin ús d'ell, amb SDL ara també podrem. En aquest cas SDL 1.3 tradueix l'entrada de l'acceleròmetre com si es tractés d'un joystick, també implementat prèviament a SDL.

Teclat: L'iPhone no compta amb un teclat físic però si és possible usar un teclat virtual de sistema. A aquesta versió s'implementen un seguit de funcions per al seu control.

OpenGL ES: Les aplicacions SDL per a iPhone usen OpenGL ES [87] per a la gestió dels gràfics. OpenGL ES és una versió reduïda d'OpenGL [86] pensada per a dispositius encastats com telèfons intel·ligents o videoconsoles.

Aquest port també té un seguit de limitacions. La majoria no ens influeixen en l'adaptació del nostre joc, a excepció del mode apaïsat.

Mode apaïsat: La resolució original de l'iPhone és 320 x 480 píxels, és a dir, en mode *portrait* o retrat, en que l'alçada és major que l'amplada. Les directrius de pintat de la SDL pintaran en aquest mode, si volem pintar en mode *landscape* o apaïsat haurem de fer un seguit de modificacions, tal com s'explica a la secció 5.3.5.

Múltiples finestres: No es poden crear aplicacions que tinguin més d'una finestra. Hauran de tenir-ne només una que omplirà tota la pantalla.

Textures: Donat que el suport per a textures es realitza a través d'OpenGL ES, que suporta menys formats que OpenGL, l'iPhone només accepta textures en els formats SDL_PIXELFORMAT_ABGR8888 i SDL_PIXELFORMAT_RGB24.

Entrada d'àudio: Encara no en suporta.

El codi font està disponible a la web oficial [99] en forma de paquet comprimit. Donat que no disposen d'una versió estable de la llibreria anomenen aquesta descàrrega *snapshot* o foto, ja que cada cert temps empaqueten el codi que va canviant constantment. El darrer snapshot el podem descarregar a través d'aquest enllaç [100].

Finalment no hem hagut de fer cap modificació al codi, però hem utilitzat molt de temps en la preparació d'un makefile per a la seva compilació. Tot aquest procediment s'explica a l'apartat 5.4.2.

5.3.2 SDL_net

Tal com s'ha explicat a la secció 3.4.1 la llibreria SDL_net es distribueix per separat de la llibreria SDL i segueix un desenvolupament a part. A diferència de SDL, actualment no existeix cap port oficial de SDL_net per a iPhone.

Al trobar-nos en aquesta situació ens vam plantejar portar nosaltres mateixos la llibreria, és a dir, adaptar el codi per a que funcioni correctament en iPhone. Per no començar de zero vam decidir basar-nos en el port per a Mac OS X donat que iOS parteix d'un subconjunt de codi de Mac OS X adaptat a una arquitectura de computador diferent. Per tant és lògic pensar que la majoria de funcionalitats bàsiques de Mac OS X també ho seran de iOS. Al final aquesta suposició va ser certa ja que no vam haver d'aplicar cap canvi al codi.

El que sí que vam haver de crear de zero va ser un Makefile i el projecte per a Xcode per a la compilació. Ambdós s'expliquen a la secció 5.4.3.

5.3.3 SDL_mixer, libogg i libvorvis

SDL_mixer és una altra llibreria que necessitem per a la compilació del 2Pong, ja que aquest fa ús de la part d'àudio del sistema.

En aquest cas tampoc es va trobar un port oficial, però sí un de personal, fet per gamehaxe. Aquest port forma part d'un projecte més gran en que l'autor ha portat vàries llibreries a iPhone i n'ha compilat la versió estàtica. També proporciona les versions per ser usades al simulador d'iPhone, que s'explica a la secció 5.5. El projecte s'anomena sdl-static [106] i a més SDL_mixer inclou les llibreries: libfreetype, libjpg, libogg, libpng, libSDL_image, libSDL_ttf, libSDL i libvorbis.

La llibreria SDL_mixer depèn de dues llibreries més, la libogg i la libvorbis i que per tant també haurem d'incloure en la compilació del 2Pong. Aquestes llibreries són necessàries per al suport de ogg i vorbis, dos formats de compressió d'àudio.

Gràcies a aquest projecte personal ja disposem dels ports de les llibreries llestes per ser enllaçades amb el 2Pong, per tant no necessitarem fer cap modificació en el codi font. Tot i així ens haurem de descarregar el codi de SDL_mixer per disposar de les capçaleres que necessitarem incloure. En el codi s'inclouen les capçaleres de libogg i libvorbis.

Ens podem descarregar el codi font de SDL_mixer des de la seva pàgina oficial [102].

5.3.4 libxml2

En el cas d'aquesta llibreria tampoc hem hagut de fer cap canvi en el codi ni compilar-la ja que existeix un port no oficial disponible al repositori Cydia de l'iPhone. Si hem instal·lat tots els paquets indicats al procés de jailbreak (secció 5.1) ja l'hauríem de tenir al nostre sistema, si no es així la baixem cercant libxml2 a l'opció **Search** de Cydia.

Tot i tenir-la a l'iPhone necessitarem copiar-la al nostre PC per tal d'enllaçar-la amb el 2Pong. Cal dir que la versió instal·lada al nostre iPhone no és una versió estàtica de la llibreria, sinó que és la versió dinàmica. Podem veure la diferència en aquests recursos [35,

36]. Això no afectarà a l'hora d'enllaçar-la en el nostre joc.

Sabem que està al directori `/usr/lib` del nostre iPhone. Usem la següent comanda des del nostre PC per copiar-la:

```
scp root@"IP iPhone":/usr/lib/libxml2.2.dylib .
```

5.3.5 2Pong

El 2Pong és un joc que està pensat per ser executat en un PC, donades les grans diferències entre un PC convencional i el nostre dispositiu caldran una sèrie de canvis per adaptar el codi a les característiques de l'iPhone. A continuació es detallen els canvis específics per a la versió d'iPhone 3G.

Adaptar la resolució

El joc per omissió està preparat per treballar amb una resolució de 640x480 píxels, però nosaltres necessitem una resolució menor per a la versió d'iPhone i per tant hem de canviar aquests valors a la inicialització del mode gràfic. Això no hauria de suposar cap problema, el joc s'hauria de mostrar en la resolució triada només canviant els valors. Però això no és tan fàcil amb el 2Pong ja que el codi és completament depenent de la resolució. Les mides dels bitmaps i les posicions on es poden pintar a la pantalla es tracten pensant només en aquesta resolució. Si la canviem haurem de fer altres canvis al codi.

El joc defineix dues constants a l'arxiu `defs.cpp` que indiquen inequívocament la resolució del joc, però que no són usades. En comptes d'això el joc assigna els valors numèrics 480 i 640 a diferents variables. Creiem que és millor fer ús de les definicions per si hem d'adaptar la resolució del joc, com és el nostre cas. Així sabrem segur que canviant el valor numèric a un sol lloc s'aplicaran els canvis a tot el codi. Les definicions al codi original són les següents:

```
#define WINDOW_HEIGHT 480
#define WINDOW_WIDTH 640
```

Les hem canviat per aquestes, adaptant-les a la resolució de l'iPhone en mode portrait o retrat, on l'alçada de la pantalla és major que l'amplada.

```
#define WINDOW_HEIGHT 480
#define WINDOW_WIDTH 320
```

Hem hagut de buscar totes les aparicions dels valors 480 o 640 i substituir-los per les definicions. Els trossos de codi que hem modificat estan al fitxer `defs.cpp`. En el primer

es defineixen dues variables que són usades en tot el codi i que representen l'alçada i l'amplada de la pantalla.

```
defines::defines():
[...]
windowheight(WINDOW_HEIGHT),
windowwidth(WINDOW_WIDTH)
{}
```

Al segon es tracta la posició dels bitmaps, si la posició assignada està fora de la pantalla es pinten al límit d'aquesta.

```
SDL_Rect FixRect (SDL_Rect rect) {
  if (rect.x <= 0) rect.x = 0;
  if (rect.x >= WINDOW_WIDTH) rect.x = WINDOW_WIDTH;
  if (rect.y <= 0) rect.y = 0;
  if (rect.y >= WINDOW_HEIGHT) rect.y = WINDOW_HEIGHT;
  if (rect.x + rect.w >= WINDOW_WIDTH) rect.w = WINDOW_WIDTH - rect.x;
  if (rect.y + rect.h >= WINDOW_HEIGHT) rect.h = WINDOW_HEIGHT - rect.y;
  return rect;
}
```

Aquesta modificació afectarà a l'enviament de dades en el joc en xarxa, ja que les posicions dels bitmaps a la pantalla es calculen en coordenades de dispositiu i no de món, pel que s'espera que tots dos jugadors estiguin usant la mateixa resolució i poder pintar els bitmaps a la mateixa posició. Per poder solucionar aquest problema haurem de modificar la posició a l'eix X dels bitmaps multiplicant aquesta posició per una constant, el valor de la qual dependrà de la resolució de la plataforma on s'executa el joc. Els canvis s'han fet a l'arxiu `net.cpp` a la funció `UpdateClient`.

```
out->data[0]=balls[1].GetX()*AJUST/3;
[...]
out->data[4]=balls[2].GetX()*AJUST/3;
```

Per tal d'evitar aquest problema i a més mantenir la relació d'aspecte original del joc la nostra idea era poder usar el mode landscape o apaïsat, usant una resolució de 480x320 píxels, però no ha estat possible, a la secció 5.3.5 s'expliquen els detalls.

Adaptar els controls

Com ja hem comentat l'iPhone no disposa de cap botó per ser usat com a mètode de control per a una aplicació. Els seus mètodes d'entrada són la pantalla multitàctil, a través de tocs amb els dits i l'acceleròmetre, detectant l'acceleració del dispositiu. Això era un inconvenient per a l'ús del 2Pong, ja que el control de les pales és realitza amb

les fletxes de direcció del teclat. L'iOS disposa d'un teclat virtual però el seu ús hauria resultat molt incòmode i a més ens restaria visibilitat de la pantalla.

Tal com hem explicat a la secció 5.3.1 SDL 1.3 tracta l'acceleròmetre com si fos un joystick, raó per la que vam pensar que seria interessant poder-ho usar com a mètode de control de les pales.

En la versió original del 2Pong no està implementada l'opció del joystick com a mètode de control, només es defineixen el ratolí i el teclat. Per incloure el nou mètode hem de fer una sèrie de canvis. A la inicialització de SDL hem d'incloure la de SDL_JOYSTICK.

```
SDL_Init(SDL_INIT_AUDIO|SDL_INIT_VIDEO|SDL_INIT_JOYSTICK)
```

Afegim el mètode de control `Joystick` a `defs.h`.

```
enum control_type { Mouse, Keyboard, Joystick};
```

A `game.cpp` implementem el control de les pales, dintre de la funció `GameMain`. Definim un joystick i una variable (ax) on emmagatzem el valor que pren el joystick en cada moment. Per acabar obrim el joystick per usar-lo.

```
SDL_Joystick *joy;
float ax = 0;
[...]
joy = SDL_JoystickOpen(0);
if (joy == NULL) {
  printf("No s'ha pogut obrir el joystick\n");
}
```

Dintre del bucle principal traduïm els valors del joystick en moviments de la pala, tal com es mostra a la figura 5.4. Per obtenir els valors de l'acceleròmetre en un eix concret usem la funció `SDL_JoystickGetAxis`. Aquesta funció retorna un Sint16 (valor de 16 bits amb signe) i per tant pren valors entre -32768 i 32767. Per una altra banda, a la implementació de SDL, s'especifica una macro anomenada `SDL_IPHONE_MAX_GFORCE` que pren per valor 5. Aquesta macro indica la màxima acceleració que pot detectar l'iPhone en unitats de força G [41]. Un objecte estacionari a la Terra al nivell del mar està suportant un força 1G, per tant quan mantinguem l'iPhone recte respecte l'eix vertical estarà suportant una força 1G en aquest eix. Tenint tota aquesta informació podem saber quins valors prendrà `SDL_JoystickGetAxis` depenent de la inclinació de l'iPhone segons l'eix vertical.

Saber aquesta informació és indispensable per programar un bon control ja que haurem de buscar una posició òptima d'inici, on la pala es mantingui aturada. Hem establert que quan el dispositiu estigui inclinat 45 graus respecte l'eix vertical estarà en posició de repòs. Hem deixat 5 graus de marge en cada sentit per a que no sigui tan sensible, és a dir, quan inclinem el dispositiu més de 50 graus la pala es mourà cap abaix i quan l'inclinem menys

```
else if (paddles[i].GetControl() == Joystick) {
  ax = SDL_JoystickGetAxis(joy, 1);
  if ( ax >= downforce)
     paddles[i].SetVelocity(0,paddles[i].GetVelocity().y-2);
  else if (ax <= upforce)
      paddles[i].SetVelocity(0,paddles[i].GetVelocity().y+2);
  else paddles[i].SetVelocity(0,0);
  if (paddles[i].GetVelocity().y > 12)
      paddles[i].SetVelocity(0,12);
  else if (paddles[i].GetVelocity().y < -12)
      paddles[i].SetVelocity(0,-12);
}
```

Figura 5.4: Tractament del joystick al bucle principal.

```
#define DEGREETORAD M_PI/180
#define MAX_JOYSTICK -32768
[...]
float upforce = 0.0;
float downforce = 0.0;
float upangle = 50*DEGREETORAD;
float downangle = 40*DEGREETORAD;
[...]
upforce = (MAX_JOYSTICK / SDL_IPHONE_MAX_GFORCE) * cos(upangle);
downforce = (MAX_JOYSTICK / SDL_IPHONE_MAX_GFORCE) * cos(downangle);
```

Figura 5.5: Calibratge de l'acceleròmetre.

de 40 graus es mourà cap amunt augmentant la velocitat com més estona mantinguem el dispositiu inclinat. A la figura 5.5 es mostra el càlcul que es guarda en unes variables amb les que es compara el valor de `SDL_JoystickGetAxis`. Amb l'ús d'aquestes variables es podria canviar la posició de repòs del joc per adaptar-la al gust del jugador.

Per fer efectiu el canvi cal modificar totes les aparicions de `SetControl(Keyboard)` per `SetControl(Joystick)`.

Pel que fa al control del menú del joc no usarem el joystick.

El mètode d'entrada quan ens trobem al menú del joc és el ratolí. En aquest cas podem aprofitar aquesta implementació per a l'iPhone ja que els tocs a la pantalla tàctil són traduïts per SDL 1.3 com a events de ratolí. Degut a que SDL 1.3 dóna suport a més d'un ratolí a la vegada, gestos multitàctils en el cas de l'iPhone, haurem d'especificar quin estem usant abans de poder capturar els events. Per tant haurem de fer algunes modificacions al codi.

A `game.cpp`, a la funció `GameMain` definim un índex per identificar el ratolí i el seleccionem. Com només volem usar un dit per desplaçar-nos pels menús el nostre joc no farà ús de

múltiples ratolins. Ara ja es capturen els moviments del dit sobre la pantalla com a un cursor i els tocs curts com a clicks.

```
int mouseIndex = 0;
[...]
mouseIndex = event.button.which;
SDL_SelectMouse(mouseIndex);
```

Un altre canvi que hem hagut d'aplicar és el mètode d'entrada per a les pantalles de pausa. Entre partida i partida el joc s'atura i ens demana que premem la tecla espai per començar de nou. Per a fer-ho més còmode en la versió d'iPhone s'ha canviat aquest event per un toc a la pantalla. S'ha modificat la condició per a començar el joc de nou.

```
if ((event.type == SDL_MOUSEBUTTONDOWN) && !game->GetBegin() && net < 2)
   {
  game->SetBegin(true);
  [...]
}
```

A part d'això s'han canviat tots els missatges "Press space to continue" per "Touch screen to continue" per fer-ho entenedor per a l'usuari.

Una altra modificació en els controls que volíem aplicar era l'ús del teclat virtual de l'iPhone per a la introcucció de les dades al mode de joc en xarxa. Finalment aquesta adaptació no ha estat possible al 2Pong, a la secció 5.3.5 s'expliquen els detalls.

SDL_Surface vs. SDL_Window

Tal com s'acaba d'explicar, durant l'adaptació del joc ens hem trobat amb un gran problema que ens ha impedit fer una sèrie de modificacions al codi per a la versió d'iPhone.

Ens vam adonar del problema quan vam intentar usar el teclat virtual de l'iPhone com a mètode d'entrada de les dades del mode de joc en xarxa del 2Pong. SDL 1.3 inclou funcions per al seu control, però com podem veure a les definicions (figura 5.6) totes tenen per paràmetre una SDL_Window.

A l'inici del 2Pong es crea una SDL_Surface on es pinten tots els bitmaps, puntuacions, missatges, etc. SDL 1.3 ja no dóna suport a aquesta funció encara que es manté a les definicions pel llegat de SDL 1.2. Com a contenidor de l'escena s'ha d'usar SDL_Window.

Passar a usar SDL_Window en comptes de SDL_Surface suposava canviar tota la rutina de pintat del joc i es va descartar, ja que suposaria reescriure tot el codi del joc, cosa que no estava contemplada a l'inici d'aquest projecte [74, 101]. A més trenca amb la nostra idea de poder unir els codis per a les diferents plataformes (iPhone, Nintendo DS i PSP) i poder-lo compilar fent ús de la compilació condicional (secció 5.4.1).

```
int SDL_iPhoneKeyboardShow(SDL_Window * window)
-- Mostra el teclat de sistema per pantalla. Retorna 0 si ha funcionat i
   -1 si dona error.
int SDL_iPhoneKeyboardHide(SDL_Window * window)
-- Amaga el teclat de sistema. Retorna 0 si ha funcionat i -1 si dona
   error.
SDL_bool SDL_iPhoneKeyboardIsShown(SDL_Window * window)
-- Retorna cert si el teclat de sistema es visualitza actualment.
int SDL_iPhoneKeyboardToggle(SDL_Window * window)
-- Canvia entre la visibilitat del teclat i la pantalla. Retorna 0 si ha
   funcionat i -1 si dona error.
```

Figura 5.6: Definicions de les funcions de SDL 1.3 per al control del teclat de l'iPhone.

Tot i així hem volgut testejar les funcions per a la gestió del teclat de l'iPhone per demostrar que funcionen (sempre que usem SDL_Window), creant una senzilla aplicació. Cal dir que abans de poder ser usades vam haver de modificar el codi de SDL per a compilar-la amb suport per al teclat d'iPhone, opció desactivada per omissió. Vam haver d'activar la definició d'una macro a l'arxiu `SDL_config_iphoneos.h` assignant-li valor 1.

```
/* enable iPhone keyboard support */
#define SDL_IPHONE_KEYBOARD 1
```

Aquesta limitació ens va impedir dur a terme una altra millora, adaptar el joc en mode landscape (480x320). Abans de saber la limitació vam estar buscant informació i vam trobar que SDL 1.3 encara no implementa aquest mode a la versió d'iPhone, però vam trobar l'explicació de com fer uns canvis a SDL i a la nostra aplicació per obtenir aquesta funcionalitat [105]. El problema és que els canvis funcionaran per aplicacions que funcionen sobre una SDL_Window. A continuació es detallen els canvis a aplicar a la llibreria SDL.

A l'arxiu `src/video/uikit/SDL_uikitview.m` afegim el següent codi després de la línia 88. Aquest codi gira els eixos del ratolí per a que detecti correctament la posició dels tocs a la pantalla tàctil.

```
CGFloat oldX = locationInView.x;
locationInView.x = locationInView.y;
locationInView.y = 320-oldX;
```

A l'arxiu `src/video/uikit/SDL_uikitwindow.m` modifiquem les línies 59 i 69 de la següent manera. Aquest codi modifica la resolució de la finestra posant-la en mode landscape (480x320).

```
window->w = 480;//(int)uiwindow.frame.size.width;
window->h = 320;//(int)uiwindow.frame.size.height;
```

Al mateix arxiu, després de la línia 69, afegim el següent codi. Aquesta línia afegeix una nova opció a la definició de la finestra per obtenir aplicacions en mode pantalla completa, o sense marcs.

```
window->flags |= SDL_WINDOW_BORDERLESS;
```

Al codi de la nostra aplicació haurem d'inicialitzar una SDL_Window amb mida 480x320 i després aplicar el següent codi. Aquestes directrius d'OpenGL giren 90 graus tota l'escena i per tant podem veure la finestra en mode landscape. Cal recordar que haurem d'enllaçar el framework OpenGLES si no ho fèiem.

```
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glViewport(0, 0, 320, 480);
glRotatef(-90, 0, 0, 1);
glOrthof(0.0, (GLfloat) 480, (GLfloat) 320, 0.0, 400, -400);
```

Aquests canvis s'han testejat en una aplicació programada originalment en mode portrait, en concret l'aplicació "Accel" que podem trobar al directori `Demos` del codi font de SDL 1.3. Aquesta aplicació usa SDL_Window i no té cap problema per adaptar-se al mode landscape després d'haver aplicat els canvis que acabem de mencionar.

5.3.6 Errors en el codi del 2Pong

Al analitzar el codi del joc que vam triar vam descobrir que tenia errors que afectaven tant a la compilació com a la usabilitat. Cal remarcar que aquests errors són comuns a totes les versions, i no només específics de la versió d'iPhone 3G.

Errors de compilació

Obtindrem com a mínim dos errors si intentem compilar el 2Pong. En primer lloc ens trobem aquest error:

```
game.h:6:27: error: SDL/SDL_mixer.h: No such file or directory
```

Això és perquè tenim includes d'aquesta forma:

```
#include <SDL/SDL_mixer.h>
```

Aquesta línia ens donarà error si la nostra estructura de fitxers no es correspon amb la indicada. Seria més correcte:

```
#include <SDL_mixer.h>
```

I serà a l'hora de compilar quan ens encarregarem d'indicar la ruta cap a aquest header.

L'altre missatge d'error és el següent:

```
defs.h:228: error: extra qualification 'defines::' on member 'UpperBound'
```

Aquest error, comunment anomenat *extra qualification* [25], ve d'aquest codi:

```
class defines
{
public:
rectangle defines::UpperBound();
[...]
};
```

En aquest tros de codi s'està definint la funció `UpperBound` dintre de la classe `defines`. Amb `defines::` estem indicant que aquesta funció pertany a la classe `defines`, cosa que és redundant. Aquesta redundància dóna error a partir de la versió 4 del compilador GCC però estava acceptada en versions anteriors.

Per tant, la solució és treure el qualificador extra:

```
rectangle UpperBound();
```

Error de teclat

En el mode de joc en xarxa, el client ha d'escriure l'adreça i el port del servidor en el format `adreça:port`. El problema és que no es captura correctament l'event de teclat que indica que s'han premut els dos punts.

El codi erroni és el següent:

```
if (event.key.keysym.sym == SDLK_SEMICOLON && (KMOD_RSHIFT || KMOD_LSHIFT
   )) {
str[len]=58;
len++;
}
```

La condició de l'if és errònia per dos motius. Primer, està programat pensant només en un teclat americà, on els dos punts comparteixen tecla amb el punt i coma (semicolon). Una solució ràpida seria adaptar el codi al teclat espanyol, però no seria la més correcta, s'hauria d'implementar una solució que no fos depenent del teclat que s'utilitza. SDL inclou suport per a teclats internacionals. Permet traduir events de teclat i posar els equivalents unicode a la variable event.key.keysym.unicode. Aquesta funcionalitat ha de ser activada amb `SDL_EnableUNICODE(1)`.

En segon lloc no està ben implementada la captura de la tecla Shift, ja que hi escriu la constant de la tecla Shift, però no la funció per llegir el seu estat, que és `SDL_GetModState()` i per tant no està fent res. Si implementem la solució proposada no ens caldrà mirar l'estat de Shift.

Fent els dos canvis el codi quedaria així:

```
SDL_EnableUNICODE(1);
[...]
else if ( str_cmp(event.key.keysym.unicode, ':') == 0 ) {
str[len]=58;
len++;
}
```

Paquet de mida insuficient

Durant el bucle principal del joc, si estem en mode multi-jugador, el client envia les seves dades contínuament al servidor. És un paquet de 3 bytes, i es reserva aquest espai quan es declara:

```
out=SDLNet_AllocPacket(3);
```

Però quan es preparen les dades per enviar, diu que és un paquet de 4 bytes, encara que finalment només se n'envien 3:

```
out->len=4;
out->data[0]=paddles[i].GetY()/3;
out->data[1]=abs(paddles[i].GetVelocity().y);
out->data[2]=sign(paddles[i].GetVelocity().y);
```

Això no provoca cap problema a simple vista en el joc de PC, però és un error. I en el cas concret de la Nintendo DS, fa que el joc no funcioni (bloqueja l'aplicació). És un error molt fàcil de solucionar: només hem d'enviar la mida correcta del paquet:

```
out->len=3;
```

Reserva de memòria insuficient

Per tal de connectar-se al servidor, el client ha d'escriure un hostname o adreça. En algunes ocasions l'algorisme falla en guardar aquesta informació en memòria. Això és degut a que no es reserva suficient espai de memòria. Ho veurem en el següent fragment de codi:

```
host = (char *) calloc(len,sizeof(char));
for (i=0;i<len;i++)
host[i]=str[i];
```

Com podem veure, l'algorisme copia el nom del host des d'un altra cadena anomenada `str`. Però primer reserva espai en memòria. El problema és que reserva `len` posicions, que és el nombre de caràcters que té la cadena `str`, i hauria de reservar una posició més, `len+1`, per guardar espai pel caràcter de final de cadena `'\0'`. Així doncs, la solució serà reservar un espai més de memòria:

```
host = (char *) calloc(len+1,sizeof(char));
```

Port no autoritzat

En el mode multi-jugador, tant el client com el servidor han d'escriure el port de comunicació que volen usar. Si en qualsevol cas es posa un port més petit de 1024 l'aplicació avisa d'un error de segmentació i es tanca.

Els ports amb número inferior a 1024 són els well-known ports [91], aquests ports estan reservats per als protocols de xarxa del sistema. Només es poden usar per establir connexions entrants si tenim privilegis de xarxa.

En concret, l'error està localitzat en el moment d'enllaçar el socket al port escollit:

```
*sock=SDLNet_UDP_Open(port);
if(!(*sock)) {
printf("ERROR SDLNet_UDP_Open: %s.\n", SDLNet_GetError());
}
```

L'anterior codi mostraria el missatge per consola:

```
ERROR SDLNet_UDP_Open: Couldn't bind to local port.
```

Una solució a aquest error seria no permetre a l'usuari introduir un port inferior a 1024. O bé que qualsevol port inferior a 1024 es transformés automàticament en 1024, que és la solució que s'ha implementat, donada la seva simplicitat.

```
port = (port<1024 ? 1024 : port);
```

Encara que creiem que la solució ideal seria que l'aplicació escollís el port mes adient automàticament, per evitar que l'usuari l'hagi d'escriure, ja que no té perquè conèixer aquesta limitació. Es podria permetre el canvi d'aquest port mitjançant un menú d'opcions.

Un altre problema de ports: dos ports iguals

De manera similar a l'apartat anterior si el port del servidor i el del client coincideixen també dóna un error de segmentació. Aquest error ocorre a la màquina del client, que és el segon en connectar-se. L'error es dóna en el mateix punt que abans, i ens dóna el mateix missatge:

```
ERROR SDLNet_UDP_Open: Couldn't bind to local port.
```

Això és degut a que el port de connexió ja està utilitzat pel servidor i no es pot enllaçar un altre socket al mateix port. La solució seria no deixar que el client escrivís el mateix port per al client que pel servidor.

El joc no acaba per al client

En acabar el joc normalment surt un missatge durant uns segons amb la puntuació final i després es torna al menú principal. En el mode multi-jugador això passa per al servidor, però no per al client. Aquest no rep el missatge de final de partida, i no torna al menú principal, pel que no pot seguir jugant.

L'error és degut a que no es comprova l'estat del joc per al client. Hi ha una funció `CheckState` que comprova l'estat de la partida, i si aquesta ha acabat (algun jugador ha arribat a 5 punts), fa totes les gestions per sortir del joc (mostrar el missatge de puntuació, tornar al menú principal). En aquest tros de codi és quan es crida a la funció `CheckState`:

```
if (net < 2) {
state=CheckState(game,balls,paddles,def);
[...]
}
```

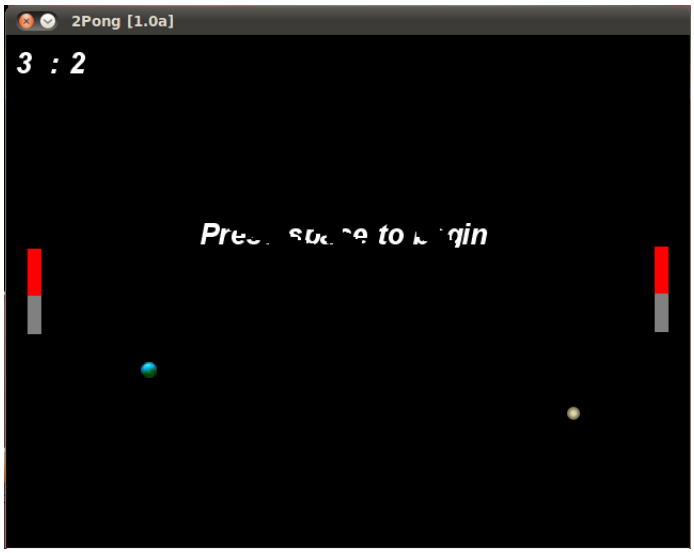


Figura 5.7: Error de visualització: El missatge "Press Space to begin" es va esborrant.

Com podem veure la crida es fa si la variable `net` és inferior a 2. Però en el cas de ser client aquesta variable té valor 2 i per tant no passa la condició.

Una solució seria comprovar l'estat del joc també quan `net` té per valor 2. Encara que la solució més correcta és que el servidor li enviï l'estat del joc al client un cop ha finalitzat.

Errors de visualització al client

En el mode multi-jugador, abans de començar a jugar cada bola, es segueix un protocol de sincronització. Tant a la pantalla del servidor com a la del client surt un missatge dient "Press Space to begin" (Prem la tecla espai per començar). El joc no comença fins que el servidor no prem la tecla Espai. Això comporta diferents errors. En primer lloc, donat que el client no ha de prémer cap tecla, se li hauria de mostrar un altre missatge, com ara "Esperant a l'altre jugador".

En segon lloc el missatge de "Press Space to begin" no s'esborra de la pantalla del client. Per un defecte en el codi aquest no rep l'ordre de refrescar la pantalla i el missatge es mostra permanentment. Després, quan la bola va passant per sobre el missatge el va esborrant i provoca un efecte molt lleig (veure figura 5.7).

El problema és que no es fa una sincronització correcta. Si el client no està en mig d'un joc està esperant rebre dades de xarxa, i quan rep qualsevol dada de xarxa comença el joc.

La solució serà la següent: reiniciar la pantalla del client quan comença el joc, tal i com es fa amb el servidor. I només fer-ho una vegada per evitar l'efecte de parpelleig o blinking.

```
estat_ant = estat_nou;
estat_nou = game->GetBegin();
if (estat_nou==1 && estat_ant==0){
InitBackground(screen);
SDL_Flip(screen);
}
```

Warnings i possibles millores

1. Existeixen més modes de joc que no es poden arribar a usar ja que no es troben a les opcions del menú. Existeix codi relatiu a aquests modes de joc, però no està acabat d'implementar. Per exemple existeix un mode en el que només hi ha una pala horitzontal a l'estil Arkanoid [19].

2. La fase d'establiment de la connexió entre client i servidor és bastant millorable. El client envia un paquet al servidor, que està esperant. Aquest rep el paquet, i li contesta. Després d'aquesta fase comença el joc. A vegades si qualsevol dels dos falla en rebre el paquet per qualsevol motiu, ja no s'estableix la connexió i ambdós es queden esperant en bucles infinits. A més el jugador que fa de servidor haurà de començar la fase de connexió primer, perquè sinó també fallarà en rebre el paquet del client. Una solució seria que si passat un temps, després de l'enviament, no s'obté la confirmació de l'arribada es tornés a enviar el paquet.

3. El joc usa com a forma de control només el teclat encara que està implementada una opció per usar el ratolí però que no es pot escollir. Donat que la idea del joc és que sigui multi-plataforma seria bona idea afegir la opció de SDL_Joystick, ja que algunes de les plataformes amb les que hem treballat tenien la interfície SDL_Joystick portada (i cap de les plataformes tenia teclat). A més de que afegiríem una opció més de control també per al jugador de PC.

4. Els missatges que apareixen al finalitzar el joc i que indiquen al jugadors si han guanyat o perdut no s'entenen, "Moshe says: YOU SUCK", "Moshe shuts the fuck up". Seria més senzill usar els típics "You win", "You lose". A més els missatges són ofensius.

5. Per últim tenim els warnings que produeix la compilació, que podem veure a l'apèndix C. Podem veure que la majoria d'ells són causats perquè es defineixen constants i variables que després no s'usen, o bé perquè s'usa una conversió d'`string` a `char*` que està obsoleta.

5.4 Compilació del codi

5.4.1 Compilació condicional

A vegades necessitarem separar el codi específic d'iPhone de la resta de plataformes. Això passarà per exemple si utilitzem una funció d'inicialització que només s'ha d'executar en iPhone i que no existeix per PC. Si intentéssim compilar per PC donaria error de compilació, donat que la funció no existeix per aquesta plataforma.

Per fer-ho utilitzarem compilació condicional gràcies a les directives de preprocés de C. Utilitzarem una macro que només està definida al Makefile d'iPhone amb el nom de `__IPHONE__`, i com que només estarà definida per aquesta plataforma, amb ella podrem controlar l'accés a determinades parts del codi.

Amb el següent exemple s'entendrà millor. Primer, es comprova si la macro `__IPHONE__` està definida. Si és així (significa que estem compilant per iPhone), es definirà l'alçada de la finestra com 320, sinó (s'està compilant per a una altra plataforma), es definirà com 480.

```
#ifdef __IPHONE__
#define WINDOW_WIDTH 320
#else
#define WINDOW_WIDTH 640
#endif
```

5.4.2 SDL

Al baixar-nos el codi font de la web oficial [99] hi trobem un directori específic per a iPhone on hi ha un projecte per a Xcode preparat per ser compilat. El projecte inclou tots els arxius font així com referències a les llibreries de l'iPhone que necessita. En canvi no es subministra cap Makefile preparat per ser executat en un entorn GNU/Linux, ja que donen per suposat que usarem l'SDK oficial per a compilar-lo.

El codi font inclou una plantilla de Makefile però no especifica els arxius necessaris per a la versió d'iPhone ni les llibreries del sistema amb les que hem d'enllaçar. Donat que l'Xcode no ens permet obtenir un Makefile a partir d'un projecte, per poder compilar SDL a iphonedevonlinux vam haver de modificar aquesta plantilla basant-nos en els arxius, llibreries i opcions definides al projecte d'Xcode.

El primer que vam fer va ser veure quins arxius de codi font i capçaleres incloïa a la compilació ja que per a iPhone s'usa un subgrup dels arxius que es subministren amb el codi font de la llibreria. Després vam mirar si es necessitava enllaçar amb frameworks del sistema (secció 3.5) i quins. En concret el projecte enllaça amb els frameworks: AudioToolbox, CoreAudio, CoreGraphics, Foundation, OpenGLES i UIKit. La manera d'enllaçar-los serà definint la següent línia al Makefile:

```
EXTRA_LDFLAGS = -framework AudioToolbox -framework CoreAudio -framework
   CoreGraphics -framework Foundation -framework OpenGLES -framework
   UIKit
```

Donada la llargada del Makefile hem decidit no incloure'l a la documentació, però el podem trobar al projecte "2pong-multiplatform" [2]. Per a més informació consultar la secció 5.7.

Ja que disposàvem del projecte d'Xcode el vam aprofitar per a la compilació de la llibreria usant l'SDK oficial. Abans de poder-lo compilar vam haver de fer alguns ajustaments al projecte.

Quan obrim el projecte veurem que tots els frameworks que apareixen estan en vermell, això és perquè l'Xcode no ha trobat l'arxiu allà on està referenciat, haurem de canviar aquesta referència per tal que apuntin als frameworks de la versió 3.0 de l'SDK. Això ho fem clickant amb el botó dret a sobre de cada framework i triant l'opció "Get Info". Ens apareixerà una finestra amb informació relacionada amb el framework, a la pestanya "General" podem canviar la localització de l'arxiu. Premem el botó "Choose" i naveguem pels directoris. Donat que usarem l'SDK 3.0 haurem de triar els frameworks d'aquesta versió que s'hauran copiat durant la instal·lació de l'SDK (secció 5.2.2). El directori és `/Developer/Platforms/iPhoneOS.platform/Developer/SDKs/iPhoneOS3.0.sdk/System/Library/Frameworks`. A "Path Type" triem **Absolute Path** per si canviem la localització del projecte no perdre la ruta relativa. Tanquem la finestra i el framework ja hauria d'aparèixer en negre.

Ara haurem de triar per quin SDK volem compilar la SDL. Des del desplegable de l'esquerra de la pantalla triem **iPhone Device 3.0** com a SDK actiu i **Release**.

Proves

Amb el codi font de la llibreria es subministra un conjunt de programes molt senzills que proven diverses funcionalitats de la llibreria i que serveixen per testejar si s'ha creat correctament.

El codi font d'aquests exemples està al directori `test` del paquet. Allà hi trobem una plantilla de Makefile que hem modificat per la compilació a iphonedevonlinux. A més existeix el projecte d'Xcode per a iPhone.

Donat que aquests tests són més genèrics no tots funcionen correctament a l'iPhone. Podem trobar un arxiu on s'informa de les limitacions per a aquesta plataforma. A més, molts d'ells no tenen part gràfica pel que per veure'n els resultats haurem d'usar la consola d'Xcode.

Al directori específic d'iPhone trobem un projecte d'Xcode preparat per compilar aquests tests. Haurem de modificar la referència a la llibreria SDL per la que acabem de compilar.

Per a iPhone, a més trobem un altre directori amb més programes de demostració específics

per aquesta plataforma. El directori es diu `Demos` i els programes posen a prova funcionalitats com la pantalla tàctil, l'àudio, l'acceleròmetre i el teclat de sistema. També s'inclou un projecte d'Xcode on també haurem d'enllaçar la SDL.

5.4.3 SDL_net

Per a la compilació d'aquesta llibreria ens hem de descarregar el codi font de la seva web oficial [103]. Com ja hem comentat a la secció 5.3.2, donat que no existeix un port per iPhone hem intentat usar el port per a Mac OS X. Tal com hem fet per a la compilació de SDL, per crear el Makefile per compilar SDL_net sota l'entorn iphonedevonlinux ens hem basat en el projecte d'Xcode que es subministra amb el codi font.

Per a la compilació en l'SDK oficial hem de crear un nou projecte d'Xcode basant-nos en el de Mac OS X. Obrim l'Xcode (`/Developer/Applications/Xcode`) i triem com a plantilla pel projecte "Cocoa Touch Static Library", seleccionant Library a l'apartat iPhone OS. Un cop creat el projecte eliminem les plantilles de codi font que s'han creat i afegim els arxius inclosos en la versió de Mac OS X. Haurem d'afegir la llibreria estàtica de SDL que hem creat a la secció 5.4.2 ja que n'és un requisit. No cal que enllacem el projecte amb cap altre framework. Pel que fa a les opcions del projecte haurem de canviar la configuració a **Release** i l'SDK a **iPhone Device 3.0**.

El Makefile i el projecte d'Xcode es poden trobar a "2pong-multiplatform" [2]. Per a més informació consultar la secció 5.7.

Proves

Per tal de comprovar el correcte funcionament de SDL_net ens podem baixar un seguit de programes de prova des d'aquesta font [104]. Entre ells trobem un exemple anomenat dnr que és el que hem usat. És un programa molt senzill que donat un hostname ens retorna la seva adreça IP, després fa l'operació inversa i ens retorna el hostname d'aquesta IP, és a dir, fa un cicle.

Aquest programa està pensat per ser executat per consola passant el hostname com a paràmetre. Donat que des de l'iPhone no podem executar l'aplicació des de consola vam haver de parsejar aquest paràmetre dintre del codi per fer la prova. El programa no té interfície gràfica, només escriu els resultats de la conversió pel canal estàndard de sortida. Per aquesta raó vàrem haver de compilar l'exemple en l'SDK oficial, usant l'Xcode per poder veure els missatges a la consola que implementa.

5.4.4 2Pong

El codi font del joc es pot descarregar des de la seva web oficial [3]. La compilació del 2Pong primer es va provar usant l'SDK oficial, creant un projecte d'Xcode des de zero, tal com s'explica a la secció 5.2.2. Podem seguir els mateixos passos que s'indiquen usant el codi font del 2Pong i enllaçant amb les llibreries necessàries. Quan hàgim d'importar els arxius ens hem d'assegurar que marquem les opcions que es mostren a la figura 5.1 per assegurar-nos que els importa respectant l'arbre de directoris. Les llibreries (secció 3.4) amb les que haurem d'enllaçar el nostre projecte són: SDL, SDL_net, SDL_mixer, libxml2, libogg, libvorbis. I els frameworks (secció 3.5) són: UIKit, Foundation, OpenGLES, AudioToolbox, QuartzCore i CoreGraphics.

Per a la compilació per a iphonedevonlinux s'ha usat el Makefile que acompanya el codi font de l'aplicació i s'ha modificat amb els paràmetres adequats.

Tots els recursos utilitzats es poden trobar a "2pong-multiplatform" [2]. Per a més informació consultar la secció 5.7.

5.5 Depuració del codi

Programar en una màquina diferent de la que executarà el codi presenta un altre problema: la depuració. Desenvolupar una aplicació sempre implica una fase de depuració per resoldre els possibles errors de programació que apareixen en temps d'execució.

Per al projecte s'han provat vàries vies per fer la depuració, buscant la més acurada i còmode per nosaltres. N'hem trobat diverses segons l'entorn de desenvolupament usat.

Missatges de control per consola: El mètode més bàsic de depuració és especificar missatges dins del codi, usant-los com a punts de control, i executar l'aplicació des d'una consola de comandes. Segons si veiem aparèixer aquests missatges per la consola o no, sabrem si l'execució ha arribat als punts on hem incrustat els missatges. Aquest mètode no és molt acurat però ens permet acotar la zona on es produeix l'error d'una manera senzilla. Pensada la idea només necessitàvem una consola on executar el codi. Ja que disposàvem de connexió ssh i accés al sistema de fitxers vam pensar en executar l'aplicació remotament, però no va funcionar, ja que donada l'estructura de les aplicacions per a iOS, abans d'executar el binari en sí, cal llegir una sèrie d'opcions especificades a l'arxiu Info.plist (secció 2.3.1) i no existeix cap mètode que ens ho permeti fer des de consola.

Escriptura de fitxers: Degut a la limitació anterior vam pensar en escriure aquests missatges en un arxiu de text mentre s'executa l'aplicació. Al finalitzar la depuració podem obrir aquest arxiu per comprovar quins missatges hi ha. Les proves van ser infructuoses ja que intentàvem escriure dintre del mateix directori de l'aplicació, cosa que no està permesa (secció 2.3). Per tant també vam descartar aquest mètode.

```
Mac OS X:
/Library/Logs/CrashReporter/MobileDevice/
Windows XP:
C:\Documents and Settings\Application Data\Apple computer\Logs\
   CrashReporter\
Windows Vista:
C:\Users\[USERNAME]\AppData\Roaming\Apple computer\Logs\CrashReporter\
   MobileDevice\
```

Figura 5.8: Directoris del pc on trobem els logs d'error.

```
Thread 0 Crashed:
0   com.yourcompany.TempConverter  0x00002b84 0x1000 + 7044
1   com.apple.Foundation           0x929ea9c8 _NSSetObjectValueAndNotify
...
28  com.yourcompany.TempConverter  0x000029dc 0x1000 + 6620
29  com.yourcompany.TempConverter  0x000026e0 0x1000 + 5856
```

Figura 5.9: Volcat d'informació d'un error a una aplicació.

***Logs* del sistema:** Una característica de l'iOS és que té un control d'errors mitjançant l'escriptura de fitxers informatius, o logs. Quan una aplicació falla durant l'execució el sistema en guarda la informació en un fitxer que podem consultar posteriorment. Per obtenir aquests arxius haurem de sincronitzar l'iPhone amb iTunes en el nostre PC, cosa que ens limita a usar Mac OS X o Windows. Un cop sincronitzat trobarem els arxius als directoris que s'especifiquen a la figura 5.8.

El sistema anota la pila d'execució usant símbols de l'aplicació. Si copiem aquests arxius directament de l'iPhone el resultat que veurem serà semblant al de la figura 5.9. Per poder traduir aquests símbols de sistema a una forma entenedora per a nosaltres haurem d'usar una comanda anomenada *atos*. També podem accedir a aquesta informació a través de l'Xcode, usant l'Organizer (`Window→Organizer`) on també hi trobarem una consola de sistema. Haurem de connectar l'iPhone per usb al Mac per a que accedeixi als arxius. Per a més informació consultar [30].

IDE: Potser la forma més còmoda sigui usant un IDE, ja que ens ofereix la possibilitat de depurar línia a línia usant un sistema de parades mentre s'està executant el codi i podem veure els valors que prenen les diferents variables en cada moment. L'SDK oficial ofereix aquesta possibilitat mitjançant l'Xcode. Connectem l'iPhone per usb al Mac i amb l'Xcode compilem el codi, si clickem a l'opció de depuració, l'Xcode instal·larà l'aplicació al dispositiu, l'executarà i la pararà quan trobi una marca feta per nosaltres al codi. Lamentablement aquesta opció només està disponible si ens hem registrat al programa de desenvolupadors (secció 3.3), per tant la vam descartar.

Simulador: Una altra eina de la que disposa l'SDK oficial és un simulador d'iOS. Amb

aquest programa podem simular el comportament de l'aplicació que hi correrem.

Un simulador és diferent a un emulador, l'emulador intenta replicar el comportament del hardware mitjançant software.

Per fer proves el simulador pot ser una bona solució, però al no ser exactament el mateix hardware ens podem trobar amb problemes al provar l'aplicació en el dispositiu que no ens apareixien en el simulador, i al revés. A més, com que el simulador no emula el hardware de l'iPhone les llibreries i les aplicacions que hàgim compilat per a iPhone no seran compatibles amb el simulador i les haurem de recompilar. Com que això ens suposava molta feina extra també vam descartar aquesta opció.

Consola de sistema Xcode: L'Xcode sí que ofereix una consola del sistema on podem veure els missatges que produeixen les aplicacions. Podem usar aquesta utilitat en combinació amb l'escriptura de missatges de control al codi per fer la depuració. Per accedir a la consola hem de connectar l'iPhone per usb al Mac i accedir a l'Organizer (`Window→Organizer`), tal com hem comentat aquí també trobarem els logs del sistema, que en combinació amb la consola ens pot brindar un bon mètode de depuració.

5.6 Instal·lació de les aplicacions

Un cop hem compilat el nostre executable haurem de crear l'aplicació i passar-la al nostre iPhone. Tal com s'ha explicat a la secció 2.3 les aplicacions d'iPhone tenen una estructura per tal que siguin reconegudes per l'iOS. En aquesta secció s'expliquen els passos a seguir usant iphonedevonlinux i l'SDK oficial.

5.6.1 Signar l'aplicació

iphonedevonlinux

Existeixen diverses maneres de signar una aplicació [23]. Per a la realització del PFC ens hem decantat per una eina anomenada ldid (Link Identity Editor) creada per Jay Freeman (saurik), donat que és l'opció que la majoria de gent que no usa l'SDK oficial recomana i a més està referenciada a la instal·lació d'iphonedevonlinux.

Aquesta aplicació, entre d'altres coses, és capaç de generar les SHA1 [108] (funció de hash criptogràfica) que són comprovades pel kernel de l'iOS. Hauríem de disposar d'ella si hem instal·lat totes les aplicacions que es detallen a la secció 5.1 durant el procés del jailbreaking. Si no ho hem fet l'haurem d'instal·lar o a través de Cydia, buscant "ldid" a la secció **Search**, o bé usant l'eina apt que també hem instal·lat durant el jailbreaking.

```
ssh root@"IP iPhone"
apt-get install ldid
```

La signatura l'haurem de fer manualment després d'haver creat l'executable i haver copiat tots els arxius al nostre iPhone donat que l'eina ldid està instal·lada al dispositiu i l'executarem des d'allà via ssh [107].

```
ssh root@"IP iPhone"
cd /Applications/nomaplicacio.app
ldid -S nomexecutable
```

SDK oficial

Tal com hem comentat a l'apartat 3.3 per tal de poder córrer una aplicació en el mateix dispositiu necessitem registrar-nos com a desenvolupadors i pagar una quota anual. Un dels passos del registre consisteix en crear un certificat de desenvolupador validat per Apple amb el que signar les nostres aplicacions. Per tal d'evitar aquest registre seguirem el següent procés.

Crear una signatura digital al nostre Mac usant l'assistent de certificats de Mac OS X.

1. Obrim `Aplicacions`→`Utilitats`→`Accés a Clauers`.
2. Del menú d'Accés a clauers triem l'opció `Assistent per a certificats`→`Crear un certificat`.
3. Triem un nom per al nostre certificat, per exemple **iPhone developer**, i com a tipus de certificat triem **Signatura en codi**.
4. També marquem l'opció de permetre ignorar els valors per omissió.
5. A la següent pantalla podem especificar el període de validesa del certificat, deixem els valors per omissió.
6. A la següent pantalla podem especificar més informació sobre nosaltres, però no és imprescindible. La podem deixar en blanc.
7. Per a la resta de pantalles deixem els valors per omissió.

Haurem d'incloure un script al nostre projecte a través d'Xcode que s'executarà després de la compilació. Això farà que la nostra aplicació es signi amb el certificat que acabem de crear.

Des de l'Xcode anem a `Project→New Buildphase→New Run Script Buildphase` i escrivim el següent codi. Si hem triat un altre nom per al nostre certificat haurem de canviar "iPhone developer" del codi amb el nom que hàgim triat.

```
if [ "${PLATFORM_NAME}" == "iphoneos" ]; then
platform=/Developer/Platforms/iPhoneOS.platform
allocate=${platform}/Developer/usr/bin/codesign_allocate
export CODESIGN_ALLOCATE=${allocate}
codesign -fs "iPhone developer" ${BUILT_PRODUCTS_DIR}/\${WRAPPER_NAME}
fi
```

Per a signar l'aplicació haurem de compilar-la i se'ns mostrarà un missatge advertint-nos que el codi es signarà amb el certificat que hem creat. Premem **Permetre**.

5.6.2 Crear o modificar l'Info.plist

iphonedevonlinux

Usant iphonedevonlinux hem de crear l'arxiu Info.plist usant aquesta plantilla bàsica canviant les següents claus. El seu significat s'explica la secció 2.3.1.

- `CFBundleDisplayName`
- `CFBundleExecutable`
- `CFBundleIdentifier`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.
   com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>CFBundleDisplayName</key>
        <string>example</string>
        <key>CFBundleExecutable</key>
        <string>example</string>
        <key>CFBundleIdentifier</key>
        <string>com.yourcompany.example</string>
        <key>CFBundleInfoDictionaryVersion</key>
        <string>6.0</string>
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleSignature</key>
        <string>????</string>
</dict>
</plist>
```

SDK oficial

Si hem seguit els passos per compilar el joc tal com es descriu a l'apartat 5.3.5 l'arxiu Info.plist s'haurà creat automàticament. L'únic que haurem de fer és modificar-lo. Haurem d'afegir la següent línia referent a la signatura de l'aplicació i, si volem, les línies explicades a la secció 2.3.1.

```
        <key>SignerIdentity</key>
        <string>Apple iPhone OS Application Signing</string>
```

5.6.3 Empaquetar

iphonedevonlinux

En aquest pas hem de crear un directori amb nom *nomaplicacio*.app.

```
mkdir nomaplicacio.app
```

Dintre hi haurem de copiar com a mínim el binari executable i l'arxiu Info.plist. Si es vol mostrar una icona també haurem de copiar la imatge. Aquests arxius s'expliquen a l'apartat 2.3.1.

```
cp arxius nomaplicacio.app
```

Si l'aplicació fa ús d'algun recurs com ara imatges, arxius de text, etc, també els haurem d'incloure. Podem crear subdirectoris dintre del directori .app depenent de l'estructura d'arxius de la nostra aplicació.

SDK oficial

Si compilem l'aplicació en l'Xcode tal com s'explica a la secció 5.3.5 el resultat serà l'aplicació ja empaquetada. Per tant no ens hem de preocupar d'aquest pas. Trobarem el *.app* a `directoridelprojecte/build/Release-iphoneos/`

5.6.4 Copiar l'aplicació a l'iPhone

Aquest pas és comú per a iphonedevonlinux i l'SDK oficial i per dur-lo a terme tenim vàries opcions segons el sistema operatiu.

Existeixen diferents programes que ens permeten l'accés al sistema de fitxers de l'iPhone. Per exemple iPhoneBrowser [65] per a Windows o DiskAid [31] per a Windows i Mac.

Durant la realització d'aquest PFC s'han copiat les aplicacions via ssh ja que aquest protocol està disponible en la majoria de sistemes operatius. El paquet ha d'estar instal·lat en ambdós dispositius, per a iPhone usarem OpenSSH. Si no l'hem instal·lat durant el procés de jailbreak, descrit a la secció 5.1, el podem obtenir a través de Cydia buscant "openssh" a la secció **Search**. És molt recomanable que abans d'usar-lo fem un canvi de contrasenyes que s'explica a la subsecció 5.6.4.

Copiem l'aplicació per ssh [107] usant la següent comanda. Totes les aplicacions es situen en el directori `/Applications`.

```
scp -r nomaplicacio.app root@"IP iPhone":/Applications
```

Ens demanarà la contrasenya de root, si no l'hem canviat escrivim **alpine**.

Reiniciem l'SpringBoard [115], que és l'aplicació que maneja la pantalla principal de l'iOS, per tal de veure l'aplicació que acabem de copiar sense la necessitat de reiniciar el dispositiu.

```
ssh root@"IP iPhone"
killall SpringBoarg
exit
```

Canvi contrasenya usuaris

L'iPhone compta amb dos usuaris per omissió al seu sistema operatiu, *mobile* (amb permisos d'usuari) i *root* (amb tots els permisos). Aquest canvi es fa per seguretat ja que la contrasenya per als dos usuaris és la mateixa i pública, *alpine*. Si no la canviem algú amb males intencions podria accedir al nostre dispositiu amb permisos de root.

Aquest canvi es pot fer des del mateix iPhone usant l'aplicació MobileTerminal instal·lada durant el jailbreaking. L'iniciem i escrivim:

```
su root
```

Quan ens demani el password escrivim **alpine** i a continuació canviem la contrasenya usant la comanda *passwd*.

```
iPhone:~ root$ passwd
Changing password for root.
Old password:
New password:
Retype new password:
iPhone:~ root$ exit
```

Per a canviar la contrasenya de mobile repetim els passos obviant la primera comanda.

Si volem fer el canvi remotament usant una connexió ssh primer haurem d'activar la xarxa Wi-Fi de l'iPhone i triar la mateixa xarxa en la que està el PC des del que executem la comanda ssh. Un cop ho hàgim fet consultarem la IP que se li ha assignat a l'iPhone. Aquesta informació la podem trobar a Preferències, Wi-Fi, i prement la fletxa blava que apareix al costat de la xarxa a la que ens hem associat. Un cop la tenim executem la següent comanda i quan ens demani el password de root escrivim **alpine**.

```
ssh root@"IP iPhone"
```

Se'ns mostrarà un missatge semblant al següent a la consola. Si és la primera vegada que establim la connexió se'ns demanarà que ho confirmem, escrivim **yes**. El sistema marcarà aquesta IP com a coneguda i no se'ns demanarà més aquesta confirmació. A continuació ens demanarà la contrasenya de root de l'iPhone, escrivim **alpine**.

```
The authenticity of host '192.168.146.4 (192.168.146.4)' can't be
   established.
RSA key fingerprint is a7:e6:fb:26:0d:03:a1:3a:4c:17:9a:15:a3:6c:78:eb.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.146.4' (RSA) to the list of known
   hosts.
root@192.168.146.4's password:
iPhone:~ root#
```

Un cop idenificats com a root canviem la contrasenya amb la comanda *passwd*.

```
iPhone:~ root$ passwd
Changing password for root.
Old password:
New password:
Retype new password:
iPhone:~ root$ exit
```

Per a canviar el password de mobile repetim els passos usant:

```
ssh mobile@"IP iPhone"
```

Podem evitar haver d'introduir la contrasenya cada vegada que ens connectem per ssh usant claus conegudes. El procés consisteix en compartir la nostra clau pública (si no la tenim la generem) amb els hosts amb els que vulguem connectar automàticament. Això funcionarà sempre que el host destinació tingui la mateixa IP i nosaltres ens connectem des de la mateixa màquina. Usarem la següent comanda per generar la nostra clau pública:

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
b1:f8:09:78:f6:43:ac:f7:16:1a:c7:14:ff:95:52:a4 user@user-laptop
```

Copiem aquesta clau pública a l'iPhone perquè ens autoritzi la connexió ssh. Usem les següents comandes:

```
ssh root@"IP iPhone" mkdir /var/root/.ssh
scp .ssh/id_rsa.pub root@"IP iPhone":/var/root/.ssh/authorized_keys
```

5.6.5 Script

Per estalviar-nos feina en el cas d'iphonedevonlinux hem creat un script per evitar-nos les tasques de signar i copiar les aplicacions manualment. Necessitarem tenir la nostra aplicació empaquetada amb tots els arxius necessaris i executar la següent comanda:

```
./iPhone.sh nom_aplicacio_sense_.app ip_iphone
```

Per usar l'script a la nostra màquina creem un arxiu amb nom, per exemple, iPhone.sh i hi copiem el codi, que podem veure a la figura 5.10. Després li hem de donar permisos d'execució de la següent manera:

```
sudo chmod a+x iPhone.sh
```

```
#! /bin/bash
#Script per passar aplicacions a l'iPhone i signar-les
if [ $# -lt 1 ]
then
echo "Us de l'script: ./iPhone.sh nom_aplicacio_sense_.app ip_iphone"
exit 1
fi
if [ $# -gt 2 ]
then
echo "Us de l'script: ./iPhone.sh nom_aplicacio_sense_.app ip_iphone"
exit 1
fi
if [ $# -eq 2 ]
then
ip=$2
fi
app=$1
echo "Es copiara l'aplicacio $app a root@$ip:/Applications i es signara"
echo "Copiant aplicacio"
scp -r $app.app root@$ip:/Applications
echo "Signant aplicacio"
ssh root@$ip ldid -S /Applications/$app.app/iPhone$app
```

Figura 5.10: Codi de l'script per automatitzar la còpia i signatura d'aplicacions.

5.7 Distribució del codi

Donat que aquest projecte s'ha intentat realitzar utilitzant només eines lliures també volíem que el codi generat ho fos. Per seguir amb la filosofia del programari lliure hem decidit distribuir tots els canvis i millores que hem fet en els codis, per a que aquests puguin ser usats i millorats si algú ho desitja.

Hem allotjat el codi a Google code ja que ens ofereix un entorn col·laboratiu còmode per projectes de codi obert o programari lliure. Aquest sistema ens ofereix un repositori amb 2 GB de capacitat per allotjar el codi, amb control de versions (subversion o mercurial), seguiment de qüestions relacionades amb el projecte (*issue tracker*), una wiki i una secció de descàrregues.

Hem anomenat el nostre projecte "2pong-multiplatform" i ja hi podem trobar tot el codi generat. Està disponible a [2].

Per descarregar el codi ho podem fer per exemple mitjançant el sistema de control de versions Subversion [8], amb la comanda:

```
svn checkout http://2pong-multiplatform.googlecode.com/svn/trunk/ 2pong-
    multiplatform
```

Encara que també es pot descarregar de la secció Downloads.

Capítol 6

Valoració econòmica

Un dels costos que comporta un projecte d'aquest tipus és l'adquisició del material necessari per a la realització del projecte. A la taula 6.1 podem consultar les despeses que suposaria adquirir aquests materials. S'ha de tenir en compte que els costos fan referència als preus en el seu moment d'adquisició, poden variar respecte als preus actuals.

El programari utilitzat no comporta cap cost addicional donat a que no s'ha usat cap programari de pagament.

Encara que sigui habitual en un projecte, no es va fer una planificació prèvia molt exhaustiva. Això és degut a que el nostre projecte implementava una idea de la que teníem suposicions que funcionaria però no en teníem proves. A més no havíem treballat prèviament amb aquests dispositius i no sabíem amb quins problemes ens trobaríem. Per últim volíem tenir una porta oberta per a possibles modificacions en els plans originals. No obstant ens hem esforçat en fer un recompte real d'hores.

Per a la realització d'aquest projecte s'hi han dedicat aproximadament unes 471 hores, que podem veure a la taula 6.2 com es distribueixen segons les tasques que s'han realitzat.

Concepte	**Descripció**	**Preu**
iPhone 3G 16 GB	Actualment només disponible al mercat de segona mà	210 €[1]
Netbook eeepc	Intel Atom N280 1.66 GHz, 1GB RAM DDR2 amb GNU/Linux Ubuntu 10.04 Lucid Lynx	395 €
MacBook	Intel Core 2 Duo 2.26 GHz, 4GB RAM DDR3 amb Mac OS X 10.6 Snow Leopard	1.080 €
Router Wi-Fi	Linksys WRT120N	49,90 €
Total		1.734,9 €

[1] Amb un contracte de Movistar de 2 anys que inclou tarifa de veu i dades (Preu de finals de 2008)

Taula 6.1: Despeses que suposarien el material utilitzat.

Concepte	**Hores**
iPhone 3G	155
Estudi de la plataforma i dels entorns de desenvolupament	23
Instal·lar i provar l'entorn iphonedevonlinux	32
Instal·lar i provar l'SDK oficial	34
Compilar i provar ports de SDL, SDL_net, SDL_mixer	50
Adaptar codi 2Pong: Problemes resolució pantalla	10
Adaptar codi 2Pong: Adaptar controls	4
Adaptar codi 2Pong: Altres	2
2Pong i entorn de desenvolupament	100
Cercar i triar el joc	15
Estudiar el codi	17
Arreglar errors generals en el codi	23
Estudiar i instal·lar Scratchbox	25
Estudiar SDL	20
Documentació i altres	216
Instal·lar i aprendre LaTeX	10
Escriure informe previ	13
Escriure memòria	158
Reunir-nos amb el director	35
Total	**471**

Taula 6.2: Distribució d'hores treballades segons tasques realitzades.

Capítol 7

Conclusions i treball futur

L'objectiu inicial del nostre projecte era portar un videojoc lliure i multi-plataforma a Nintendo DS i iPhone i permetre que els dos dispositius interaccionessin en un mode de joc en xarxa. Podem concloure que aquest objectiu ha estat assolit, ja que disposem d'una versió jugable del joc en els dos dispositius i la comunicació entre ells funciona. Per tant podem dir que hem trencat les barreres del màrqueting de les que parlàvem a la introducció.

A més, durant la realització d'aquest projecte s'han assolit altres objectius que no estaven previstos al començament com ara portar llibreries, aplicar millores de codi o fins i tot portar el joc a una altra plataforma, PSP.

La realització d'aquest projecte m'ha aportat molts coneixements en l'àmbit del desenvolupament d'aplicacions i de tot el procés que implica. He après que muntar un entorn de desenvolupament no és gens fàcil i menys encara muntar un entorn de compilació creuada. També vull destacar l'ús de LaTeX, un format que desconeixia. He de dir que el seu aprenentatge ha valgut la pena i segur que el tornaré a usar. En especial m'ha agradat programar per a iPhone ja que és un dispositiu que dia a dia està guanyant popularitat com a plataforma de desenvolupament i això m'atreia i també em resultava un repte.

Aquest projecte no s'hauria pogut dur a terme sense la contribució de les comunitats de desenvolupadors de programari lliure. La seva feina gratuïta i desinteressada ens ha permès obtenir informació de forma ràpida que hauria estat molt difícil obtenir per qualsevol altra via. Volem agrair aquesta ajuda d'igual manera, compartint el codi i els coneixements adquirits amb la comunitat.

Aquest document es pot usar de guia de programació per a iPhone usant eines lliures i multiplataforma, però cal advertir que aquestes eines estan en constant evolució i per tant alguns punts del document quedaran obsolets en poc temps. Segurament apareixeran més mètodes per dur a terme la feina feta, per això tenim la responsabilitat de compartir-la per tal que el projecte segueixi amb vida. Això també ha influït en la cerca d'informació, ja que aquesta només la podem trobar a la xarxa. Cal tenir cura de discernir la veracitat de les fonts.

Usar el codi programat per una altra persona ha tingut pros i contres. Encara que ens hauria agradat programar el nostre propi videojoc vam decidir cercar-ne un de fet que complís amb les premisses del projecte per poder centrar els nostres esforços en la seva adaptació per aconseguir la connexió entre els dispositius. Al final, a més d'invertir temps en adaptació, en vam haver d'invertir per a l'estudi del codi, tant per entendre'l al principi com per corregir-lo després, ja que contenia molts errors. Va sorgir un altre gran problema. Donat que el joc fa ús de funcions de SDL 1.2 que ja no es suporten a SDL 1.3, moltes de les noves característiques que ofereix SDL 1.3 pròpies d'iPhone no han pogut ser aprofitades.

Programar per a iPhone usant entorns de desenvolupament i eines diferents a les oficials ha estat molt difícil donat a les limitacions imposades per Apple. Per començar, en la recerca d'informació ja vaig tenir problemes donat que no es troben gaires opcions per fer desenvolupament amb toolchains lliures sobre GNU/Linux i les opcions que hi ha no estan molt mantingudes. A més en la majoria de llocs a la xarxa, quan es tracta d'exemplificar casos de programació per iPhone, usen l'SDK oficial sobre Mac OS X i no s'explica com es faria usant un entorn lliure sobre GNU/Linux. Per tant la cerca de solucions a la xarxa també ha estat complicada. Les coses sempre semblen funcionar si usem l'SDK oficial.

Tot i que el procés ha estat molt dur el fet de poder veure com l'aplicació s'executa a l'iPhone 3G i als altres dispositius ha estat molt gratificant. El millor d'haver triat un videojoc és que hem pogut alliberar la tensió jugant unes partides en xarxa.

7.1 Possibles millores i ampliacions

Com en tot projecte teníem un temps limitat i per tant no hem pogut fer totes les coses que ens hauria agradat. Per aquest motiu vam haver de donar prioritat a algunes tasques sobre d'altres i és per això que encara queda marge per a un futur treball a partir d'aquesta entrega. A més, el fet que tot el programari creat sigui lliure fa que qualsevol pugui continuar treballant-hi. A continuació s'enumeren una sèrie de possibles ampliacions.

- Portar el joc usant la darrera versió del firmware de l'iPhone, actualment la 4.2.
- Portar el joc a SDL 1.3 per obtenir funcionalitats com les descrites a la secció 5.3.5.
- Aportar millores al port de SDL 1.3 per a iPhone ja que actualment es troba en desenvolupament.
- Millorar el sistema d'entrada del 2Pong per a iPhone incloent una opció de calibració de l'acceleròmetre. També es podria afegir un mode de control usant el dit per moure la pala i crear un nou mode multijugador on es podria explotar la característica de la pantalla multitàctil per a que dos jugadors poguessin jugar en un sol iPhone, controlant cada un una pala.

- El codi del 2Pong també accepta moltes millores encara, s'hi podrien afegir noves funcionalitats no dependents de la plataforma, com per exemple:
  - Millores gràfiques.
  - Acabar d'implementar algunes funcionalitats que no estan acabades, com per exemple controlar el joc amb el ratolí, o un nou mode de joc en què es pot jugar en horitzontal a l'estil Arkanoid.
  - Incloure més opcions al menú com per exemple per desactivar els power-ups, la possibilitat d'afegir un port de connexió per omissió, etc.

  Són només alguns suggeriments, però es podria explotar encara molt la idea.
- Encara que ja hem distribuït el codi font generat durant el projecte ens agradaria publicar l'aplicació per a iPhone a Cydia, el repositori d'aplicacions no vinculat a Apple, accessible després d'haver fet el jailbreak a l'iPhone.
- Per acabar es podria portar el joc a més plataformes i de tipus més variats, per així experimentar amb nous entorns de treball. Algunes idees són: una plataforma més oberta com GP2X Caanoo [53], alguna plataforma amb Android [7] o potser a alguna videoconsola de sobretaula com Wii [124].

Apèndix A

Instal·lació i manual del joc

A.1 Instal·lació del 2Pong a l'iPhone

Per tal d'instal·lar el 2Pong al nostre iPhone haurem de seguir les indicacions detallades a la secció 5.6.

Si hem usat iphonedevonlinux per a compilar l'aplicació l'haurem d'empaquetar manualment. A continuació es detallen els arxius que haurem d'incloure com a mínim en el directori `2Pong.app`. Si volem podem incloure més resolucions d'icones, tal com es detalla a la secció 2.3.1. Dintre del directori `2pong.app/src/data` hi copiarem tots els recursos que usa el joc, com bitmaps, efectes de so, etc. Aquests arxius es troben al mateix directori relatiu al codi font original del 2Pong.

- 2pong.app
  - 2pong
  - Info.plist
  - conf.xml
  - Icon.png
  - src
    - data

      ...

A.2 Manual del joc

Iniciem l'aplicació prement sobre l'icona del 2Pong a la finestra principal de l'iPhone.

El joc comença amb el menú principal, on podem triar entre els dos modes de joc principals: Regular (un jugador) i Network (multi-jugador en xarxa). Per saber com navegar pels menús, es pot seguir l'esquema 4.1 que hem vist durant l'anàlisi del joc.

Si entrem en el mode Network, tant el servidor com el client hauran d'introduir el port que volen usar, i el client a més haurà d'escriure l'adreça IP i el port del servidor separats per dos punts. El port ha d'estar entre 1024 i 65535. En comptes d'una adreça IP també podem fer servir un hostname, com per exemple localhost. Per exemple:

- El servidor introdueix el port **5555** (i té adreça 192.168.1.33).
- El client introdueix el port **6666**. Després introdueix les dades del servidor **192.168.1.33:5555**.

Cal remarcar que si volem fer una partida en xarxa el primer en començar la partida i introduir les dades haurà de ser el servidor.

El joc implementa un seguit de power-ups:

DoubleSpeed Fa que la velocitat de la bola es dupliqui.

HalfSpeed Fa que la velocitat de la bola es redueixi a la meitat.

IncSpeed Incrementa un 25% la velocitat de la bola.

DecSpeed Decrementa un 25% la velocitat de la bola.

ChangePos Mou la bola a una posició aleatòria de la pantalla.

Split Duplica la bola.

Controls de la versió per a iPhone

La posició més còmoda per al joc és subjectant verticalment l'iPhone amb una mà.

Per seleccionar les opcions dels menús usarem la pantalla tàctil. Arrossegant un dit per sobre de la pantalla les opcions s'aniran ressaltant quan hi passem per sobre, un cop ressaltada hem de prémer amb un dit a qualsevol altra zona de la pantalla per seleccionar-la. Per al control de la pala s'usa l'acceleròmetre de l'iPhone. La posició de repòs, on la pala no es mou, es situa a uns 45 graus d'inclinació respecte l'eix vertical, si inclinem l'iPhone més de 50 graus la pala es mou cap abaix i si l'inclinem menys de 40 graus es mou cap amunt augmentant la velocitat com més estona mantinguem la inclinació. Per sortir de l'aplicació premem el botó *home* (situat sota la pantalla) de l'iPhone.

Cal dir que en el mode d'un jugador controlem la pala de la dreta, i en el mode en xarxa controlem la pala dreta si fem de servidor i l'esquerra si fem de client.

Apèndix B

Arxius Plist

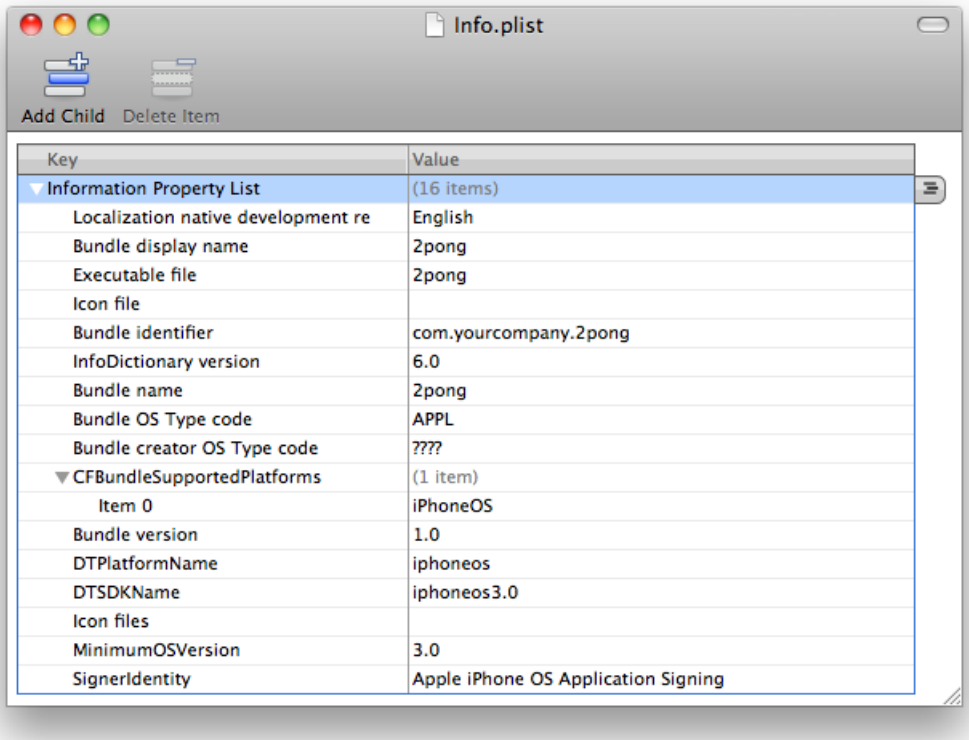


Figura B.1: Captura de pantalla del Property List Editor visualitzant un arxiu *.plist*, el mateix arxiu en format XML a la figura B.2, i en text pla a la figura B.3

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.
    com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>CFBundleDevelopmentRegion</key>
        <string>English</string>
        <key>CFBundleDisplayName</key>
        <string>2pong</string>
        <key>CFBundleExecutable</key>
        <string>2pong</string>
        <key>CFBundleIdentifier</key>
        <string>com.yourcompany.2pong</string>
        <key>CFBundleInfoDictionaryVersion</key>
        <string>6.0</string>
        <key>CFBundleName</key>
        <string>2pong</string>
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleSignature</key>
        <string>????</string>
        <key>CFBundleSupportedPlatforms</key>
        <array>
                <string>iPhoneOS</string>
        </array>
        <key>CFBundleVersion</key>
        <string>1.0</string>
        <key>DTPlatformName</key>
        <string>iphoneos</string>
        <key>DTSDKName</key>
        <string>iphoneos3.0</string>
        <key>MinimumOSVersion</key>
        <string>3.0</string>
        <key>SignerIdentity</key>
        <string>Apple iPhone OS Application Signing</string>
</dict>
</plist>
```

Figura B.2: Exemple d'arxiu Info.plist en format XML, per veure la versió editor a la figura B.1 i la versió text pla a la figura B.3

```
{
        "CFBundleDevelopmentRegion" = English;
        CFBundleDisplayName = 2pong;
        CFBundleExecutable = 2pong;
        CFBundleIdentifier = "com.yourcompany.2pong";
        "CFBundleInfoDictionaryVersion" = "6.0";
        CFBundleName = 2pong;
        CFBundlePackageType = APPL;
        CFBundleSignature = "????";
        "CFBundleSupportedPlatforms" = (
                iPhoneOS,
        );
        CFBundleVersion = "1.0";
        DTPlatformName = iphoneos;
        DTSDKName = "iphoneos3.0";
        MinimumOSVersion = "3.0";
        SignerIdentity = "Apple iPhone OS Application Signing";
}
```

Figura B.3: Exemple d'arxiu Info.plist en format text pla, per veure la versió editor a la figura B.1 i la versió xml a la figura B.2

Apèndix C

Warnings del 2Pong

```
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
     --cflags` `sdl-config --cflags` -o defs.o defs.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
defs.h: In constructor 'defines::defines()':
defs.h:242: warning: defines::powerups' will be initialized after
defs.h:240: warning: 'int defines::windowheight'
defs.cpp:168: warning: when initialized here
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
     --cflags` `sdl-config --cflags` -o CSpriteBase.o CSpriteBase.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
CSpriteBase.cpp: In member function 'int CSpriteBase::init(char*)':
CSpriteBase.cpp:24: warning: ignoring return value of 'char* fgets(char*, int, FILE*)',
    declared with attribute warn_unused_result
CSpriteBase.cpp:33: warning: ignoring return value of 'char* fgets(char*, int, FILE*)',
    declared with attribute warn_unused_result
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
     --cflags` `sdl-config --cflags` -o CSprite.o CSprite.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
     --cflags` `sdl-config --cflags` -o font.o font.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
font.cpp: In function 'SDLFont* initFont(char*, float, float, float, float)':
font.cpp:57: warning: ignoring return value of 'int fscanf(FILE*, const char*, ...)',
    declared with attribute warn_unused_result
font.cpp: In function 'void drawString(SDL_Surface*, SDLFont*, int, int, char*, ...)':
font.cpp:154: warning: array subscript has type 'char'
font.cpp:183: warning: array subscript has type 'char'
font.cpp:194: warning: array subscript has type 'char'
font.cpp: In function 'int stringWidth(SDLFont*, char*, ...)':
font.cpp:214: warning: array subscript has type 'char'
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
     --cflags` `sdl-config --cflags` -o ai.o ai.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
ai.cpp:11: warning: unused parameter 'difficulty'
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
     --cflags` `sdl-config --cflags` -o physics.o physics.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
     --cflags` `sdl-config --cflags` -o net.o net.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
net.cpp: In function 'void UpdateClient(Game*, Ball*, Paddle*, _UDPsocket*, int)':
net.cpp:39: warning: unused variable 'num'
net.cpp: In function 'IPaddress GetAddress(SDL_Surface*, SDLFont*)':
```

```
net.cpp:165: warning: deprecated conversion from string constant to 'char*'
net.cpp:195: warning: deprecated conversion from string constant to 'char*'
net.cpp: In function 'int GetPort(SDL_Surface*, SDLFont*, int)':
net.cpp:228: warning: deprecated conversion from string constant to 'char*'
net.cpp:229: warning: deprecated conversion from string constant to 'char*'
net.cpp:243: warning: deprecated conversion from string constant to 'char*'
net.cpp:255: warning: deprecated conversion from string constant to 'char*'
net.cpp:256: warning: deprecated conversion from string constant to 'char*'
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
    --cflags` `sdl-config --cflags` -o menu.o menu.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
menu.cpp: In function 'char* GetNodeValue(xmlNode*, char*)':
menu.cpp:13: warning: deprecated conversion from string constant to 'char*'
menu.cpp: In function 'xmlNode* GetPlayNode(xmlNode*, char*, char*)':
menu.cpp:23: warning: deprecated conversion from string constant to 'char*'
menu.cpp:24: warning: deprecated conversion from string constant to 'char*'
menu.cpp:25: warning: deprecated conversion from string constant to 'char*'
menu.cpp:26: warning: deprecated conversion from string constant to 'char*'
menu.cpp: In function 'void DrawMenu(xmlNode**, int, int, SDL_Surface*, SDLFont*)':
menu.cpp:65: warning: deprecated conversion from string constant to 'char*'
menu.cpp:67: warning: deprecated conversion from string constant to 'char*'
menu.cpp:69: warning: deprecated conversion from string constant to 'char*'
menu.cpp:71: warning: deprecated conversion from string constant to 'char*'
menu.cpp:79: warning: deprecated conversion from string constant to 'char*'
menu.cpp:80: warning: deprecated conversion from string constant to 'char*'
menu.cpp: In function 'xmlNode* ShowNode(xmlNode*, char*, SDL_Surface*, SDLFont*)':
menu.cpp:95: warning: deprecated conversion from string constant to 'char*'
menu.cpp:100: warning: deprecated conversion from string constant to 'char*'
menu.cpp: In function 'options* ParseNode(xmlNode*)':
menu.cpp:155: warning: deprecated conversion from string constant to 'char*'
menu.cpp:156: warning: deprecated conversion from string constant to 'char*'
menu.cpp:157: warning: deprecated conversion from string constant to 'char*'
menu.cpp:158: warning: deprecated conversion from string constant to 'char*'
menu.cpp:159: warning: deprecated conversion from string constant to 'char*'
menu.cpp: In function 'int InitMenu(char*, Game*, Ball*, Paddle*, Powerup*, defines,
    CSpriteBase*, Mix_Chunk*, Mix_Chunk*)':
menu.cpp:193: warning: deprecated conversion from string constant to 'char*'
menu.cpp:224: warning: deprecated conversion from string constant to 'char*'
menu.cpp:249: warning: deprecated conversion from string constant to 'char*'
menu.cpp:299: warning: deprecated conversion from string constant to 'char*'
menu.cpp:303: warning: deprecated conversion from string constant to 'char*'
menu.cpp:311: warning: deprecated conversion from string constant to 'char*'
menu.cpp:311: warning: deprecated conversion from string constant to 'char*'
menu.cpp:313: warning: deprecated conversion from string constant to 'char*'
menu.cpp:313: warning: deprecated conversion from string constant to 'char*'
menu.cpp:320: warning: deprecated conversion from string constant to 'char*'
menu.cpp:179: warning: unused variable 'aout'
menu.cpp: In function 'xmlNode* GetNode(xmlNode*, char*)':
menu.cpp:53: warning: control reaches end of non-void function
menu.cpp: In function 'int InitMenu(char*, Game*, Ball*, Paddle*, Powerup*, defines,
    CSpriteBase*, Mix_Chunk*, Mix_Chunk*)':
menu.cpp:184: warning: 'name' may be used uninitialized in this function
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
    --cflags` `sdl-config --cflags` -o draw.o draw.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
draw.cpp: In function 'void DrawScene(Game*, Ball*, Paddle*, Powerup*, defines)':
draw.cpp:7: warning: unused variable 'font'
draw.cpp: In function 'void DrawResults(Game*, Paddle*, int, int)':
draw.cpp:49: warning: deprecated conversion from string constant to 'char*'
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
    --cflags` `sdl-config --cflags` -o powerups.o powerups.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
powerups.cpp:8: warning: unused parameter 'mode'
powerups.cpp:23: warning: unused parameter 'mode'
```

```
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
    --cflags` `sdl-config --cflags` -o balls.o balls.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
balls.cpp:23: warning: unused parameter 'difficulty'
balls.cpp:113: warning: unused parameter 'puk'
balls.cpp:113: warning: unused parameter 'pluck'
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
    --cflags` `sdl-config --cflags` -o paddles.o paddles.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
paddles.cpp:51: warning: unused parameter 'balls'
paddles.cpp:51: warning: unused parameter 'difficulty'
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
    --cflags` `sdl-config --cflags` -o game.o game.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
game.cpp:3: warning: unused parameter 'address'
game.cpp: In function 'void InitGame(Game*, Ball*, Paddle*, Powerup*, int, int, defines)':
game.cpp:178: warning: deprecated conversion from string constant to 'char*'
game.cpp:178: warning: deprecated conversion from string constant to 'char*'
game.cpp: In function 'void ResetGame(Game*, Ball*, Paddle*, Powerup*, int, int, defines)
    ':
game.cpp:201: warning: deprecated conversion from string constant to 'char*'
game.cpp:201: warning: deprecated conversion from string constant to 'char*'
game.cpp: In function 'int CheckState(Game*, Ball*, Paddle*, defines)':
game.cpp:223: warning: deprecated conversion from string constant to 'char*'
game.cpp:225: warning: deprecated conversion from string constant to 'char*'
game.cpp:228: warning: deprecated conversion from string constant to 'char*'
game.cpp:230: warning: deprecated conversion from string constant to 'char*'
game.cpp:256: warning: control reaches end of non-void function
game.cpp: In function 'int GameMain(Game*, int, Ball*, Paddle*, Powerup*, defines,
    CSpriteBase*, Mix_Chunk*, Mix_Chunk*, IPaddress, _UDPsocket*, int, int)':
game.cpp:16: warning: 'in' may be used uninitialized in this function
game.cpp:16: warning: 'out' may be used uninitialized in this function
g++ -c -pipe -fno-exceptions -Wall -W -O2 -march=i486 -mcpu=i686 -D_REENTRANT `xml2-config
    --cflags` `sdl-config --cflags` -o main.o main.cpp
`-mcpu=' is deprecated. Use `-mtune=' or '-march=' instead.
main.cpp:25: warning: unused parameter 'powerups'
main.cpp: In function 'void FirstInitPowerups(Powerup*, defines, CSpriteBase*)':
main.cpp:40: warning: deprecated conversion from string constant to 'char*'
main.cpp:41: warning: deprecated conversion from string constant to 'char*'
main.cpp:42: warning: deprecated conversion from string constant to 'char*'
main.cpp:43: warning: deprecated conversion from string constant to 'char*'
main.cpp:44: warning: deprecated conversion from string constant to 'char*'
main.cpp:45: warning: deprecated conversion from string constant to 'char*'
main.cpp:46: warning: deprecated conversion from string constant to 'char*'
main.cpp: At global scope:
main.cpp:36: warning: unused parameter 'powerups'
main.cpp:36: warning: unused parameter 'def'
main.cpp: In function 'void FirstInitBalls(Ball*, SDL_Surface*, defines)':
main.cpp:50: warning: deprecated conversion from string constant to 'char*'
main.cpp: In function 'void FirstInitPaddles(Paddle*, SDL_Surface*, defines)':
main.cpp:58: warning: deprecated conversion from string constant to 'char*'
main.cpp: In function 'int main(int, char**)':
main.cpp:117: warning: deprecated conversion from string constant to 'char*'
main.cpp:122: warning: deprecated conversion from string constant to 'char*'
main.cpp: At global scope:
main.cpp:76: warning: unused parameter 'argc'
main.cpp:76: warning: unused parameter 'argv'
g++ -fno-exceptions -Wl,-rpath,/usr/lib/qt/lib -o 2pong defs.o CSpriteBase.o CSprite.o
    font.o ai.o physics.o net.o menu.o draw.o powerups.o balls.o paddles.o game.o main.o `
    xml2-config --libs` `sdl-config --libs` -lSDL_net -lSDL_mixer
mv 2pong ../
```

Bibliografia

[1] 2Pong. http://twopong.sourceforge.net/. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[2] 2pong-multiplatform. http://code.google.com/p/2pong-multiplatform/. [Online; Visitada per darrera vegada el 13 de gener de 2011].

[3] 2Pong source code. http://prdownloads.sourceforge.net/twopong/2pong-src-1.0.1a.tar.gz?download. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[4] 3G — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=3G&oldid=405215108. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[5] Advanced Packaging Tool. http://wiki.debian.org/Apt. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[6] Allegro - A game programming library. http://alleg.sourceforge.net/. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[7] Android.com. http://www.android.com/. [Online; Visitada per darrera vegada el 13 de gener de 2011].

[8] Apache Subversion. http://subversion.apache.org/. [Online; Visitada per darrera vegada el 31 de desembre de 2010].

[9] App icons on iPad and iPhone. http://developer.apple.com/library/ios/#qa/qa2010/qa1686.html. [Online; Visitada per darrera vegada l'1 de gener de 2011].

[10] App Store — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=App_Store&oldid=404054549. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[11] App Store metrics. http://148apps.biz/app-store-metrics/. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[12] App Store Review Guidelines. http://developer.apple.com/appstore/guidelines.html. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[13] Apple Developer. http://developer.apple.com/programs/register/. [Online; Visitada per darrera vegada el 10 de gener de 2011].

[14] Apple Developer Program. http://developer.apple.com/programs/which-program/. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[15] Apple Developer Registration. http://developer.apple.com/programs/start/register/create.php. [Online; Visitada per darrera vegada el 7 de gener de 2011].

[16] Apple Developer Tools. http://developer.apple.com/technologies/tools/. [Online; Visitada per darrera vegada el 4 de gener de 2011].

[17] Apple Disk Image — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Apple_Disk_Image&oldid=404054957. [Online; Visitada per darrera vegada el 28 de desembre de 2010].

[18] Apple Inc. http://www.apple.com/. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[19] Arkanoid. http://es.wikipedia.org/wiki/Arkanoid. [Online; Visitada per darrera vegada el 8 de gener de 2011].

[20] ARM1176 Processor. http://www.arm.com/products/processors/classic/arm11/arm1176.php. [Online; Visitada per darrera vegada el 3 de gener de 2011].

[21] BitTorrent (protocol) — Wikipedia,The Free Encyclopedia. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[22] Bluetooth — wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Bluetooth&oldid=405548815. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[23] Bypassing iPhone Code Signatures. http://www.saurik.com/id/8. [Online; Visitada per darrera vegada el 31 de desembre de 2010].

[24] Chroot — Wikipedia, La enciclopedia libre. http://es.wikipedia.org/w/index.php?title=Chroot&oldid=42505152. [Online; Visitada per darrera vegada el 8 de gener de 2011].

[25] Common syntax errors in C++. http://womble.decadent.org.uk/c++/syntax-errors.html. [Online; Visitada per darrera vegada el 7 de gener de 2011].

[26] Computer multitasking — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Computer_multitasking&oldid=402715430. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[27] Cross compiler — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Cross_compiler&oldid=402608600. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[28] Cross-platform — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Cross-platform&oldid=405027135. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[29] Cydia. http://cydia.saurik.com/. [Online; Visitada per darrera vegada el 5 de gener de 2011].

[30] Debugging and Symbolizing Crash Dumps in Xcode. http://developer.apple.com/tools/xcode/symbolizingcrashdumps.html. [Online; Visitada per darrera vegada el 14 de gener de 2011].

[31] DiskAid. http://www.digidna.net/products/diskaid. [Online; Visitada per darrera vegada el 7 de gener de 2011].

[32] Downgrade iPhone 3G o 3GS del iOS 4 al 3.1.3. http://www.iphonediario.com/28-06-2010/downgrade-iphone-3g-o-3gs-del-ios-4-al-3.1.3. [Online; Visitada per darrera vegada el 9 de gener de 2011].

[33] Dpkg — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Dpkg&oldid=404646353. [Online; Visitada per darrera vegada el 5 de gener de 2011].

[34] Dynamic random access memory — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Dynamic_random_access_memory&oldid=404143010. [Online; Visitada per darrera vegada el 3 de gener de 2011].

[35] Enllaç dinàmic — Viquipèdia, l'Enciclopèdia Lliure. http://ca.wikipedia.org/w/index.php?title=Enlla%C3%A7_din%C3%A0mic&oldid=5342111. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[36] Enllaç estàtic — Viquipèdia, l'Enciclopèdia Lliure. http://ca.wikipedia.org/w/index.php?title=Enlla%C3%A7_est%C3%A0tic&oldid=5342112. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[37] Exploit (computer security) — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Exploit_(computer_security)&oldid=403667730. [Online; Visitada per darrera vegada el 3 de gener de 2011].

[38] Firmware — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Firmware&oldid=405511347. [Online; Visitada per darrera vegada el 9 de gener de 2011].

[39] Free software. http://www.gnu.org/philosophy/free-sw.html. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[40] Freeware — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Freeware&oldid=404260800. [Online; Visitada per darrera vegada el 28 de desembre de 2010].

[41] G-force — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=G-force&oldid=406265139. [Online; Visitada per darrera vegada el 16 de gener de 2011].

[42] Galaxy Gameworks. http://www.galaxygameworks.com/index.html. [Online; Visitada per darrera vegada el 12 de gener de 2011].

[43] gcc-4.3. http://packages.debian.org/lenny/gcc-4.3. [Online; Visitada per darrera vegada el 14 de gener de 2011].

[44] gcc-4.3-multilib. http://packages.debian.org/unstable/devel/gcc-4.3-multilib. [Online; Visitada per darrera vegada el 14 de gener de 2011].

[45] GCC, the GNU Compiler Collection. http://gcc.gnu.org/. [Online; Visitada per darrera vegada el 9 de gener de 2011].

[46] Geotagging — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Geotagging&oldid=404369550. [Online; Visitada per darrera vegada el 3 de gener de 2011].

[47] Global Positioning System — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Global_Positioning_System&oldid=405299320. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[48] GNU Binutils. http://www.gnu.org/software/binutils/. [Online; Visitada per darrera vegada el 9 de gener de 2011].

[49] GNU C Library. http://www.gnu.org/software/libc/. [Online; Visitada per darrera vegada el 9 de gener de 2011].

[50] GNU General Public License. http://www.gnu.org/licenses/gpl.html. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[51] GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[52] Google Summer of Code. http://code.google.com/intl/es-ES/soc/. [Online; Visitada per darrera vegada el 12 de gener de 2011].

[53] GP2X Caanoo — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/GP2X_Caanoo. [Online; Visitada per darrera vegada el 13 de gener de 2011].

[54] Greenpois0n. http://greenpois0n.com/. [Online; Visitada per darrera vegada el 12 de gener de 2011].

[55] Grep. http://www.gnu.org/software/grep/. [Online; Visitada per darrera vegada el 5 de gener de 2011].

[56] How To Setup Cross Compile Environment. http://biffengineering.com/wiki/index.php?title=HowToSetupCrossCompileEnvironment. [Online; Visitada per darrera vegada el 8 de gener de 2011].

[57] IEEE 802.11. http://standards.ieee.org/getieee802/download/802.11-2007.pdf. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[58] Integrated development environment — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Integrated_development_environment&oldid=401386832. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[59] IOS (Apple) — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=IOS_(Apple)&oldid=404098080. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[60] iOS Dev Center. http://developer.apple.com/devcenter/ios/index.action. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[61] iOS Dev Center Support. http://developer.apple.com/support/ios/ios-dev-center.html. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[62] iOS Developer Program. http://developer.apple.com/programs/ios/. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[63] iOS SDK 4.2. http://adcdownload.apple.com/ios/ios_sdk_4.2__final/xcode_3.2.5_and_ios_sdk_4.2_final.dmg. [Online; Visitada per darrera vegada el 31 de desembre de 2010].

[64] IPhone — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=IPhone&oldid=405779852. [Online; Visitada per darrera vegada el 4 de gener de 2011].

[65] iPhone browser. http://code.google.com/p/iphonebrowser/. [Online; Visitada per darrera vegada el 7 de gener de 2011].

[66] IPhone Dev Team — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=IPhone_Dev_Team&oldid=399916240. [Online; Visitada per darrera vegada el 28 de desembre de 2010].

[67] IPhone Dev Team blog. http://blog.iphone-dev.org/. [Online; Visitada per darrera vegada el 28 de desembre de 2010].

[68] iphone-dev Toolchain. http://code.google.com/p/iphone-dev/. [Online; Visitada per darrera vegada el 31 de desembre de 2010].

[69] iPhone firmware 3.0 download. http://appldnld.apple.com.edgesuite.net/content.info.apple.com/iPhone/061-6582.20090617.LlI87/iPhone2,1_3.0_7A341_Restore.ipsw. [Online; Visitada per darrera vegada el 30 de desembre de 2010].

[70] IPhone models — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=IPhone&oldid=405779852#Models. [Online; Visitada per darrera vegada el 4 de gener de 2011].

[71] iphonedevonlinux Toolchain 3.0. http://code.google.com/p/iphonedevonlinux/wiki/Installation. [Online; Visitada per darrera vegada el 29 de desembre de 2010].

[72] Issue 185. http://code.google.com/p/iphone-dev/issues/detail?id=185. [Online; Visitada per darrera vegada el 10 de gener de 2011].

[73] iTunes download. http://www.apple.com/es/itunes/download/. [Online; Visitada per darrera vegada el 28 de desembre de 2010].

[74] Lacking the bridge between SDL-Window and SDL-Surface. hhttp://forums.libsdl.org/viewtopic.php?t=6816&sid=385c2fd612cc1c4e7bf42d9a975b31ad. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[75] LaTeX – A document preparation system. http://www.latex-project.org/. [Online; Visitada per darrera vegada el 10 de gener de 2011].

[76] libxml2. http://xmlsoft.org/. [Online; Visitada per darrera vegada el 5 de gener de 2011].

[77] Libxml2 — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Libxml2&oldid=397898647. [Online; Visitada per darrera vegada el 4 de gener de 2011].

[78] Maemo. http://maemo.org/. [Online; Visitada per darrera vegada el 6 de gener de 2011].

[79] Massively multiplayer online role-playing game — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Massively_multiplayer_online_role-playing_game&oldid=405706141. [Online; Visitada per darrera vegada el 4 de gener de 2011].

[80] MBX Graphics IP Core Family. http://www.imgtec.com/powervr/mbx.asp. [Online; Visitada per darrera vegada el 3 de gener de 2011].

[81] Mobile terminal. http://code.google.com/p/mobileterminal/. [Online; Visitada per darrera vegada el 5 de gener de 2011].

[82] Multi-touch — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Multi-touch&oldid=405255839. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[83] Newlib C library. http://sourceware.org/newlib/. [Online; Visitada per darrera vegada el 9 de gener de 2011].

[84] Nokia. http://www.nokia.com/. [Online; Visitada per darrera vegada el 8 de gener de 2011].

[85] Objective-C — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Objective-C&oldid=403868189. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[86] OpenGL. http://www.opengl.org/. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[87] OpenGL ES. http://www.khronos.org/opengles/. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[88] OpenSSH — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=OpenSSH&oldid=402456932. [Online; Visitada per darrera vegada el 5 de gener de 2011].

[89] Operating system — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Operating_system&oldid=404487753. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[90] Pong - Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Pong&oldid=403308092. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[91] Port numbers. http://www.iana.org/assignments/port-numbers. [Online; Visitada per darrera vegada el 7 de gener de 2011].

[92] Property list — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Property_list&oldid=398330973. [Online; Visitada per darrera vegada l'1 de gener de 2011].

[93] Property List Programming Guide. http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/PropertyLists/Introduction/Introduction.html. [Online; Visitada per darrera vegada el 5 de gener de 2011].

[94] PwnageTool — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=PwnageTool&oldid=403375347. [Online; Visitada per darrera vegada el 28 de desembre de 2010].

[95] PwnageTool 3.0 download. http://thepiratebay.org/user/iphonedev/. [Online; Visitada per darrera vegada el 28 de desembre de 2010].

[96] Python plistlib. http://docs.python.org/dev/library/plistlib.html. [Online; Visitada per darrera vegada el 5 de gener de 2011].

[97] Quemu. http://wiki.qemu.org/Main_Page. [Online; Visitada per darrera vegada el 8 de gener de 2011].

[98] Scratchbox. http://www.scratchbox.org/. [Online; Visitada per darrera vegada el 6 de gener de 2011].

[99] SDL 1.3 download. http://www.libsdl.org/hg.php. [Online; Visitada per darrera vegada el 12 de gener de 2011].

[100] SDL 1.3 snapshot. http://www.libsdl.org/tmp/SDL-1.3.tar.gz. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[101] SDL-CreateWindow vs. SDL-SetVideoMode. http://forums.libsdl.org/viewtopic.php?t=6312&sid=d5c9ae80f640f5cf77bfdf08b5245867. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[102] SDL mixer. http://www.libsdl.org/projects/SDL_mixer/. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[103] SDL-net 1.2. http://www.libsdl.org/projects/SDL_net/release/SDL_net-1.2.7.tar.gz. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[104] SDL-net demos. http://jonatkins.org/SDL_net/SDL_net_demos.tar.gz. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[105] SDL on iPhone : Landscape Mode. http://jrs0ul.com/en/reply/?msg=24. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[106] sdl-static. http://code.google.com/p/sdl-static/. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[107] Secure Shell — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Secure_Shell&oldid=400207261. [Online; Visitada per darrera vegada el 31 de desembre de 2010].

[108] SHA-1 — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=SHA-1&oldid=404589595. [Online; Visitada per darrera vegada el 7 de gener de 2011].

[109] Simple DirectMedia Layer. http://www.libsdl.org/. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[110] Simple DirectMedia Layer - Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Simple_DirectMedia_Layer&oldid=399486134. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[111] Smartphone — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Smartphone&oldid=404125868. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[112] Sn0wbreeze. http://ih8sn0w.com/index.php/welcome.snow. [Online; Visitada per darrera vegada el 12 de gener de 2011].

[113] Software cracking — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Software_cracking&oldid=401495161. [Online; Visitada per darrera vegada el 28 de desembre de 2010].

[114] Software development kit — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Software_development_kit&oldid=402153142. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[115] SpringBoard — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=SpringBoard&oldid=403375685. [Online; Visitada per darrera vegada l'1 de gener de 2011].

[116] Subscriber Identity Module — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Subscriber_Identity_Module&oldid=404815076. [Online; Visitada per darrera vegada el 31 de desembre de 2010].

[117] Superuser — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Superuser&oldid=404747951. [Online; Visitada per darrera vegada el 4 de gener de 2011].

[118] TeX — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=TeX&oldid=405527657. [Online; Visitada per darrera vegada el 10 de gener de 2011].

[119] The Application Runtime Environment. http://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/RuntimeEnvironment/RuntimeEnvironment.html. [Online; Visitada per darrera vegada el 14 de gener de 2011].

[120] Toolchain — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Toolchain&oldid=384182958. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[121] Touchscreen — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Touchscreen&oldid=405179342#Capacitive. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[122] Underclocking — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Underclocking&oldid=399108006. [Online; Visitada per darrera vegada el 3 de gener de 2011].

[123] Wi-Fi — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Wi-Fi&oldid=405513816. [Online; Visitada per darrera vegada el 2 de gener de 2011].

[124] Wii - Nintendo. http://www.nintendo.es/NOE/es_ES/wii_54.html. [Online; Visitada per darrera vegada el 13 de gener de 2011].

[125] Wikibooks - Manual de LaTeX. http://es.wikibooks.org/w/index.php?title=Manual_de_LaTeX&oldid=142864. [Online; Visitada per darrera vegada el 10 de gener de 2011].

[126] Wolfson codecs. http://www.wolfsonmicro.com/products/codecs/. [Online; Visitada per darrera vegada el 3 de gener de 2011].

[127] Xcode. http://developer.apple.com/technologies/tools/xcode.html. [Online; Visitada per darrera vegada el 27 de desembre de 2010].

[128] Xcode Build Setting Reference. http://developer.apple.com/library/ios/#documentation/DeveloperTools/Reference/XcodeBuildSettingRef/0-Introduction/introduction.html. [Online; Visitada per darrera vegada el 15 de gener de 2011].

[129] XML. http://www.w3.org/XML/. [Online; Visitada per darrera vegada el 4 de gener de 2011].

[130] Gabriel Valiente Feruglio. *Composició de textos científics amb LATEX*. Edicions UPC, 1998.

[131] Antonio Gabriel Escañuela Rodríguez. Videojocs lliures: Interacció multiplataforma, 2011. Facultat d'Informàtica de Barcelona (UPC).

[132] Zdziarski, Jonathan A. *iPhone Open Application development*. Farnham O'Reilly, 2008.