Títol: JClic per OLPC

Volum: 1

Alumne: Marc Rodríguez Tristany

Director/Ponent: María José Casany

Departament: LSI

Data: 8/01/2009

ÍNDEX DE CONTINGUTS

I. INTRODUCCIÓ

OLPC (One Laptop Per Child) és un projecte que està desenvolupant un ordinador portàtil de baix cost, amb la finalitat que aquest pugui fer arribar el món de la informàtica arreu.

El que pretén, a grans trets, és que els nens i nenes dels països més pobres, que mai han arribat a tocar, o fins i tot, saber com és un ordinador, puguin fer-ho, i al mateix temps, aprendre a partir d'aquest.

OLPC és un petit *laptop* que aprofita el món de l'Open Source per tal de poder abaratir al màxim el preu d'aquest ordinador, i així fer que els governs, escoles, d'aquests països, puguin assumir-ne el cost. El fet que sigui Open Source permet que qualsevol desenvolupador pugui implementar les seves aplicacions i col·laborar així a millorar la qualitat d'aquest. El llenguatge d'alt nivell que suporta és Python.

Consta d'un sistema operatiu Linux Fedora (anomenat Sugar) limitat, amb una interfície molt simple, i molt intuïtiva. El *laptop* també disposa accés per xarxa sense fils, i ja de sèrie, diferents activitats i jocs per als nens.

JClic és un conjunt d'aplicacions informàtiques de codi obert encarades a estudiants, des dels més menuts fins a batxillerat, d'índole merament educativa, creades pel departament d'ensenyament de la Generalitat de Catalunya, per tal que

aquests puguin aprendre diferents conceptes jugant amb l'ordinador, a base de trencaclosques, sopes de lletres,etc.

Aquest aplicatiu està format per diferents mòduls desenvolupats en l'entorn Java i es poden presentar de dos maneres diferents, a partir d'Internet, mitjançant un applet o instal.lant-los i executant-los com una aplicació més del nostre sistema operatiu (tant pot ser Windows, Linux o Mac OS).
Bàsicament, els mòduls principals de JClic són l'Author, on els mateixos professors dissenyen el "joc", definint les característiques, imatges, sons, etc que aquest "joc" tindrà, i el Player, que és la plataforma que els executa i on l'alumne participa resolent els "jocs".

Així doncs, en el context en el que ens trobem, i partint de l'aplicatiu JClic i el *laptop* OLPC, podem definir l'objectiu del nostre **PFC**, que no és més que apropar-los, traduint JClic per que pugui executar-se a la plataforma del projecte OLPC.

Amb aquest projecte es preten proporcionar continguts al projecte OLPC. Una de les mancances principals del projecte OLPC és la falta de continguts educatius per les aplicacions que venen amb el *laptop*. A Catalunya, Espanya i altres països de Sud Amèrica existeixen molts projectes implementats ja en JClic. Per tant, amb aquest projecte es podràn executar els ja existents en la plataforma de OLPC.

No obstant, com que JClic és molt gran, ens hem centrat només en el Player (el que realment executa els jocs creats per els professors) i en 3 tipus de jocs (els trencaclosques d'intercanvi, els dobles i els de forats).

II. EL PROJECTE OLPC

A. INICIS

Els orígens del OLPC es remunten molt més enllà del moment en que s'engendra aquesta màquina. Ja fa més de 4 dècades, quan els ordinadors encara no eren un aparell d'ús diari per a les persones (inclosos els nens), l'home va començar a pensar que aquests aparells podien ser de gran utilitat per la educació i es va posar a treballar per assolir aquesta meta.

El primer exemple d'iniciativa d'utilitzacio dels ordinadors per l'educació dels nens va ser l'any 1967, quan un grup de persones van crear Logo, el primer llenguatge de programació desenvolupat específicament per els nens i dissenyat per Danny Bobrow, Wally Feurzeig y Seymour Papert.

A partir d'aquí van començar a aparèixer diferents projectes, aplicacions i màquines pensades perquè els nens aprenguessin jugant. Les primeres potències mundials començaven a incorporar amb força els computadors a la vida quotidiana i deixaven enrere els països més pobres, que no tenien els mitjans necessaris per accedir a les noves tecnologies de la informació i la comunicació (TIC). Aquesta divisió entre països rics i països pobres pel què fa a l'accés a les TIC reb el nom d'escletxa digital (en anglés Digital Divide). Davant dels problemes que tenien els països més pobres del planeta per accedir a les TIC, van començar a sorgir

projectes de cooperació, com pot ser el projecte que al 1982 va distribuir varis mils de l'antic Apple II a estudiants de Dakar. Entre les persones que van col·laborar en aquest projecte trobem al senyor Negroponte, que com veurem més endavant, ha sigut el creador de l'OLPC.

A l'any 2002, Negroponte, després d'un projecte on regala varis ordinadors a una petita població de Cambodja, pensa que es podria intentar fabricar un *laptop* de petites dimensions i baix preu (100 $ inicialment), per tal de fer arribar les TIC arreu del món. Negroponte, pensant en concret en la gent menys afortunada, que no tenia oportunitat d'accedir a les TIC, comença a pensar en un projecte de gran envergadura.

A partir d'aquest moment, es posa a treballar per tal d'aconseguir el seu objectiu, i en aquell mateix any, les organitzacions AMD, News Corp., Google i Red Hat (creador del Sugar) es manifesten a favor del projecte. Diversos països com Brasil, Tailandia, etc, es mostren molt interessats.

En els següents mesos, altres companyies i països s'agregaren al projecte, fins que a finals del 2006, surten els primers OLPC's (de l'anglès One Laptop Per Child) de les oficines de Massachussets.

A partir d'aquest pas de gegant, la màquina coneguda arreu del món com OLPC s'ha anat actualitzant, refinant i complementant (fet que ha encarit

considerablement el cost, que ja costa uns 200 $) fins al dia d'avui, on ja en molts països els nens disfruten aprenent informàtica amb el petit ordinador.

B. VISIO DE L'APRENENTATGE QUE FOMENTA EL PROJECTE OLPC

En aquest apartat intentarem enumerar quins són els principis pedagògics en els que es basa el projecte OLPC, explicant-los breument:

En primer lloc cal tenir en compte que el projecte OLPC no és un projecte de creació d'un nou ordinador de baix cost, sinó que és un projecte educatiu. L'ordinador o *laptop* és la base tecnològica per portar a terme un projecte educatiu que es basa en els principis pedagògics que s'expliquen més endavant.

En el projecte OLPC té molta importància el treball col·laboratiu. L'XO (és el nom que reb l'ordinador o *laptop*) està pensat perquè els nens puguin treballar de forma cooperativa els uns amb els altres. El fet que els nens treballin conjuntament fa que s'hagin de relacionar entre ells i que comparteixin experiències. També permet compartir coneixements entre ells ajudant a fer que l'aprenentatge sigui més ràpid i divertit.

Un altre principi important del projecte OLPC és la importància d'aprendre jugant. Encara que en la educació formal els jocs encara són considerats com una activitat poc sèria, poden arribar a ser de gran utilitat en l'aprenentatge dels nens. Els jocs

representen un gran estímul pels nens. Mentres juguen, i encara que ells no se'n adonin, estan aprenent al mateix temps.

Amb l'aprenentatge basat en el joc es pot motivar molt més fàcilment els nens. Els jocs ofereixen un ambient molt proper al alumne que fa que es predisposi més al aprenentatge. L'OLPC es basa en l'aprenentatge basat en el joc.

Una de les comunitats més importants que treballen en aquest aspecte és SIG-GLUE (*Special Interest Group for Game-based Learning in Universities and Life Long Learning*).

Una educació de qualitat i un sistema d'aprenentatge per tothom són bàsics per aconseguir una societat justa, equitable, econòmica i socialment viable. Això està relacionat amb els "Objectius del Mil.leni de les Nacions Unides" on es diu que tothom ha de tenir dret a una educació bàsica. El projecte, en aquest aspecte, és una bona iniciativa, ja que el que busca és arribar als racons més hinòspits del planeta, per tal que tot nen pugui arribar a conéixer el món de la informàtica. Aporta un granet d'arena per tal de poder assolir aquests objectius.

L'accés als ordinadors portables a les escoles pot portar beneficis importants a nivell nacional en el procés d'aprenentatge. Mentres no baixi el preu dels ordinadors portables aquest potencial d'aprenentatge queda reduït a una minoria selecta de persones.

Si tots els nens del món tenen ordinadors, com pretén el projecte OLPC, podran utilitzar-los com una eina de suport a l'aprenentatge i ajudar a trencar l'escletxa digital (Digital Divide).

C. L'ORDINADOR XO

L'XO és la màquina o ordinador que ha sigut creada pel projecte OLPC, per ajudar a lluitar contra la fractura digital al món.

En aquest apartat es descriuen les característiques més importants de l'ordinador XO.

1. Característiques

L'XO és una eina, que més que caracteritzar-se per la seva potència, rapidesa, o prestacions, es caracteritza per la facilitat d'interacció que té amb l'usuari (pensem que està fet exclusivament per als nens), pel disseny atractiu, però sobretot per la seva facilitat d'ús.

A continuació s'expliquen les característiques que té l'XO, dividides en característiques hardware, software, d'interfície i de disseny.

2. Hardware

L'XO va se creada, com hem dit, entre l'any 2002 i 2006, per un grup d'empreses que es van interessar i involucrar en el projecte sense ànim de lucre. Amb la força d'aquestes companyies grans, van poder fer realitat el projecte, aconseguint un ordinador suficientment equilibrat i potent, per poder oferir el seu servei fonamental, jugar i aprendre.

Com que un dels punts claus en el desenvolupament de l'XO era construir una màquina amb un preu final baix, no es va intentar dissenyar un ordinador amb totes les prestacions que la tecnologia actual permet. Més aviat, es va pensar en dissenyar un ordinador amb les capacitats necessàries perquè els nens dels països pobres la poguessin utilitzar per jugar i aprendre. Com que aquesta màquina havia de ser utilitzada en entorns on les condicions ambientals podien ser extremes, es va dissenyar una màquina versàtil, àgil, robusta i durable (tant energèticament com de durabilitat de la màquina parlant), on els nens poguessin disfrutar i aprendre sense cap mena d'impediment.

L'XO és una màquina pensada per als nens. Per aquest motiu ha de ser petita, poc pesada, robusta, que entri bé per els ulls, i cridanera. En la foto següent podeu veure-ho.

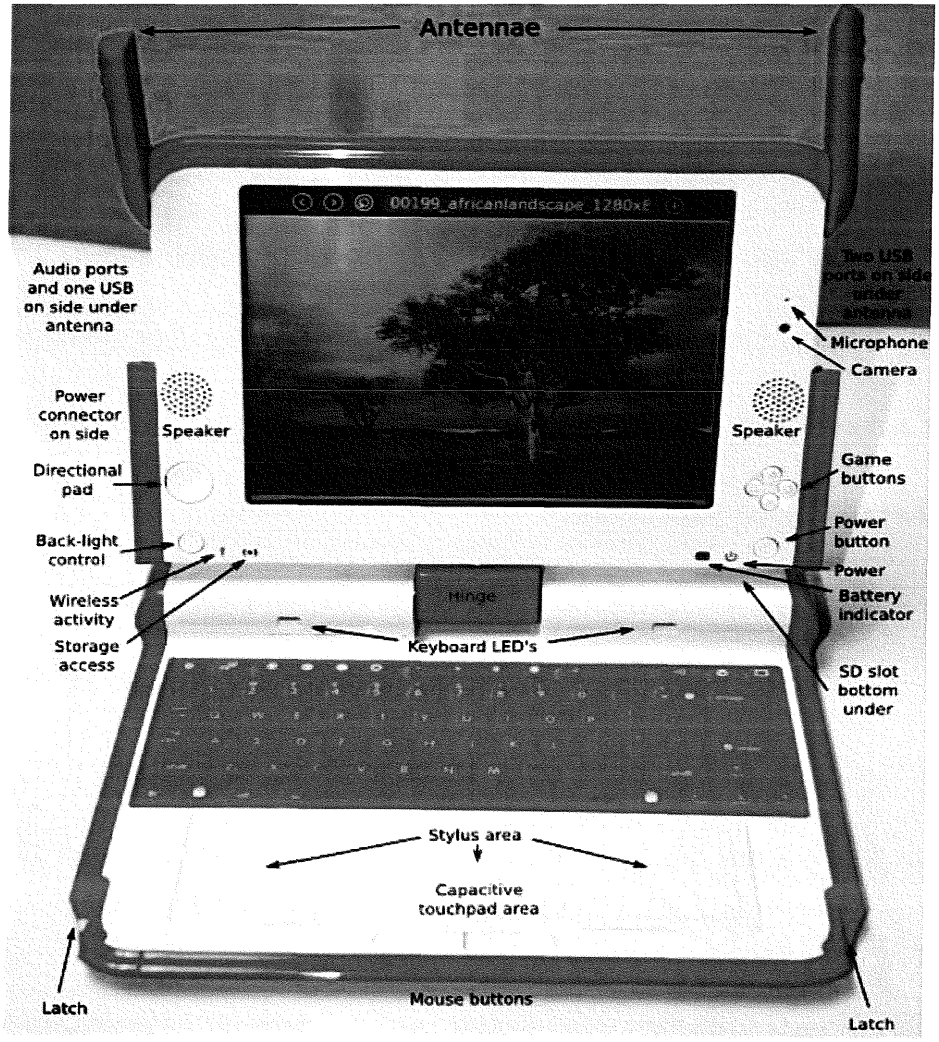


Figura 1: XO

Parlant d'aspectes més tècnics, la màquina mesura 242mmx228mmx32mm i pesa 1,5 Kg i la carcassa permet moure la pantalla al gust de l'usuari.

Mirant dins la seva pell, trobem una CPU AMD Geode a 433 MHZ compatible amb les instruccions de l'Athlon i el controlador gràfic està integrat dins la CPU, amb una arquitectura de memòria unificada. Té una memòria RAM de 256 MB DDR3 i té una de 1024 MB de tipus flash.

Té una pantalla TFT de 7 polzades i mitja i la resolució és de 1200x900.

Com a perifèrics dels quals disposa, podem destacar el teclat amb membrana de goma, el touchpad, que suporta un mode d'entrada d'escriptura, l'àudio, una targeta AD1888, targeta inalàmbrica Marvell compatible amb els estàndards 802.11b/g amb doble antena, com podem veure a la foto, i una càmera de vídeo.

També disposa de micròfon, sortida d'àudio, alimentació amb bateria (dura un mínim de 2000 cicles de càrrega/descàrrega) o a la corrent, 3 connectors USB i una ranura per targeta MMC/SD.

El que tingui una durabilitat tan alta és degut al poc escalfament que té.

3. Software

L'XO utilitza únicament software lliure. El fet que sigui així ha permès, principalment, abaratir de forma important el cost de la màquina perquè no cal obtenir ni pagar llicències de software específiques. A més, qualsevol programador por realitzar les seves pròpies aplicacions i executar-les en l'XO. El fet d'utilitzar programari lliure, ha fet possible que un gran nombre de persones s'hagin involucrat en aquest projecte per així millorar-lo i arribar a codificar aplicacions que siguin beneficioses per als nens que utilitzen l'ordinador. Tant mestres, alumnes com programadors tenen la llibertat d'inventar, reutilitzar el software i continguts d'aquestes aplicacions.

L'XO consta així, d'un Sistema Operatiu (SO) Linux Fedora versió Core 6 adaptat a les capacitats del hardware que incorpora.

Aquest SO suporta diferents entorns de programació:

Python, amb el qual està construïda la interfície gràfica i els diferents models d'activitats.

JavaScript per tal de poder executar scripts al navegador.

Csound, un ambient de so i àudio programable.

Squeak, una petita versió de Smalltalk.

Logo, un llenguatge d'alt nivell de molt fàcil aprenentatge, raó per la qual s'utilitza per treballar amb nens.

També disposa d'algun nivell de suport per Java i Flash.

Les aplicacions natives de les què disposa, enumerades de forma ràpida, son l'Xulrunner, que és el browser d'Internet, Evince, un simple visualitzador de documents, el processador de textos AbiWord, un lector de RSS, un client de correu electrònic, un client de xat, VoIP, un diari, una wiki, un reproductor i editor multimèdia, una eina de composició musical, eines per treballar amb imatges i gràfics, jocs, una shell i un debbuger.

Les biblioteques i plug-ins que utilitza inclouen Xul, GTK+, Matchbox, Pango, ATK, Cairo, X Window, Avahi i gstreamer.

4. Interfícies

Parlem ara una mica de la interfície d'usuari de l'XO. Aquesta ha de ser, com podem comprendre, fàcil d'utilitzar i feta a mesura de l'usuari que l'ha d'utilitzar i al treball que ha de realitzar, l'aprenentatge. Per tant, s'ha creat una interfície que captura gràficament el seu món de companys, amics, mestres, col·laboradors, fent més ènfasi en les connexions dins d'una mateixa comunitat d'usuaris, entre les persones i les seves activitats.

El que s'ha fet per tal que això sigui possible, és en primer lloc, anomenar a tot el que per nosaltres són aplicacions informàtiques, activitats. Això representa una qualitat intrínseca de la experiència de l'aprenentatge que s'espera que tinguin els nens quan utilitzin la màquina.

Una altra qualitat a destacar, importantíssima i que facilita l'aprenentatge col·lectiu comentat anteriorment, és el fet que el portàtil utilitza una xarxa de malla que interconnecta totes les màquines XO que abasta.

Aquesta xarxa permet que totes les activitats que l'usuari utilitzi passen a convertir-se en activitats en xarxa i que per tant poden ser compartides per diversos nens, que poden jugar de forma cooperativa.

Un dels altres aspectes fonamentals que disposa la interfície Sugar és el Journal (diari), que consisteix bàsicament en un compendi de totes les activitats executa-

des i realitzades en la màquina durant l'estona, hora, dia en el qual l'hem estat utilitzant.

Aquest Journal realitza una organització cronològica de registre de les coses fetes per l'usuari, però també les realitzades implícitament en la seva participació en les activitats. Això li permet veure de forma íntegra, tot el que ha anat realitzant i poder tornar a una activitat a la que havia estat, en el moment en el que la va tancar. Aquí podem veure alguns screenshots de la interfície gràfica de l'XO.

En aquesta primera imatge de sota veiem la pantalla principal del Sugar, on podem veure el nivell de bateria, les aplicacions obertes o els documents que té el clipboard. El ninotet que veiem al mig de la pantalla simbolitza l'usuari que utilitza l'ordinador i és així com el veuran els altres usuaris de la comunitat a la que està.

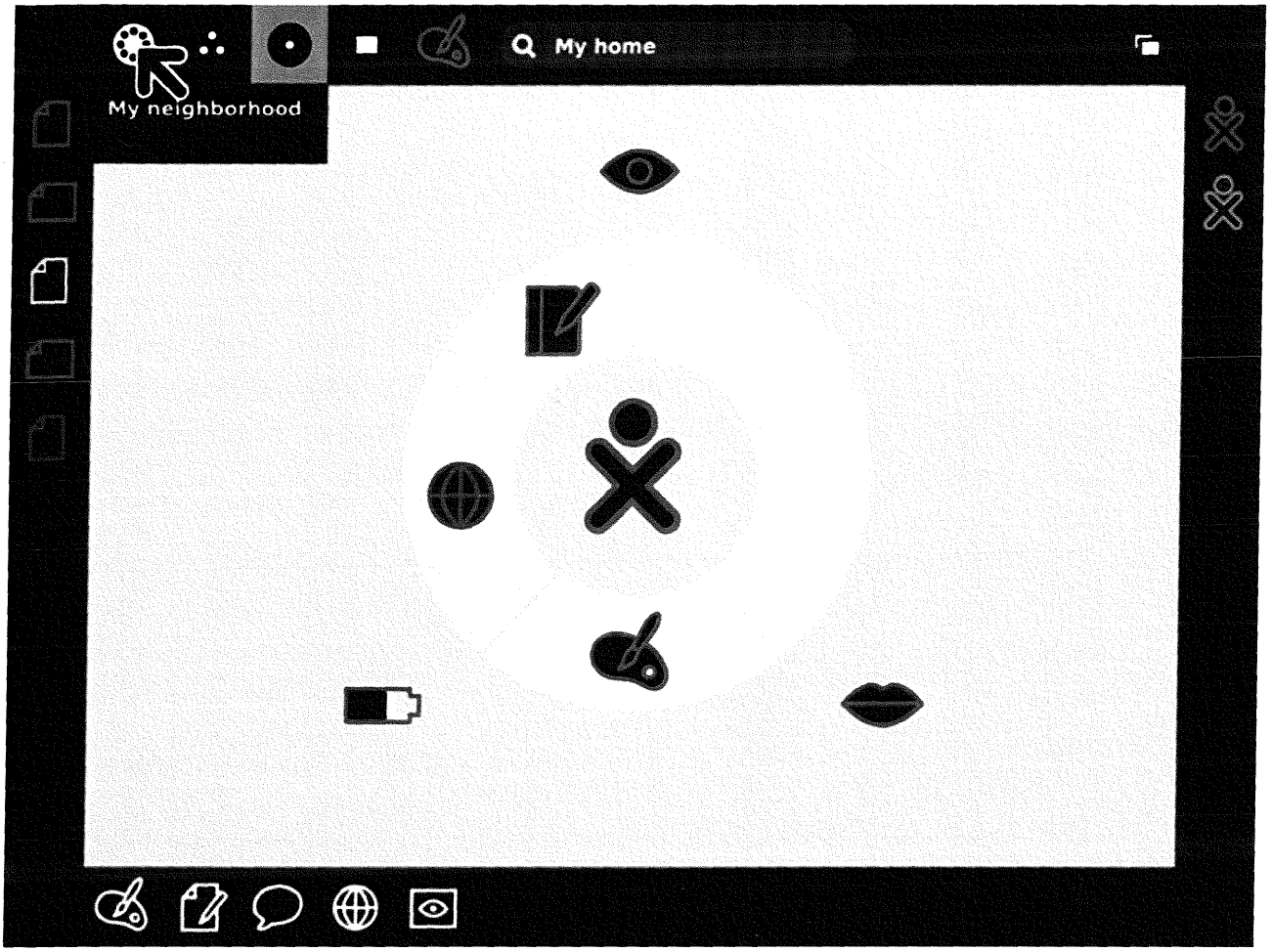


Figura 2: Vista de la interficie de Sugar

En aquesta segona imatge podem veure la comunitat d'usuaris a la que està inclòs l'usuari que utilitza la màquina. Podem així veure un conjunt d'usuaris, els quals uns comparteixen la utilitat de dibuix, els altres el browser.

Amb un simple clic es pot tant xatejar amb la gent de la mateixa comunitat, compartir activitats, i així, aprendre en col·lectiu.

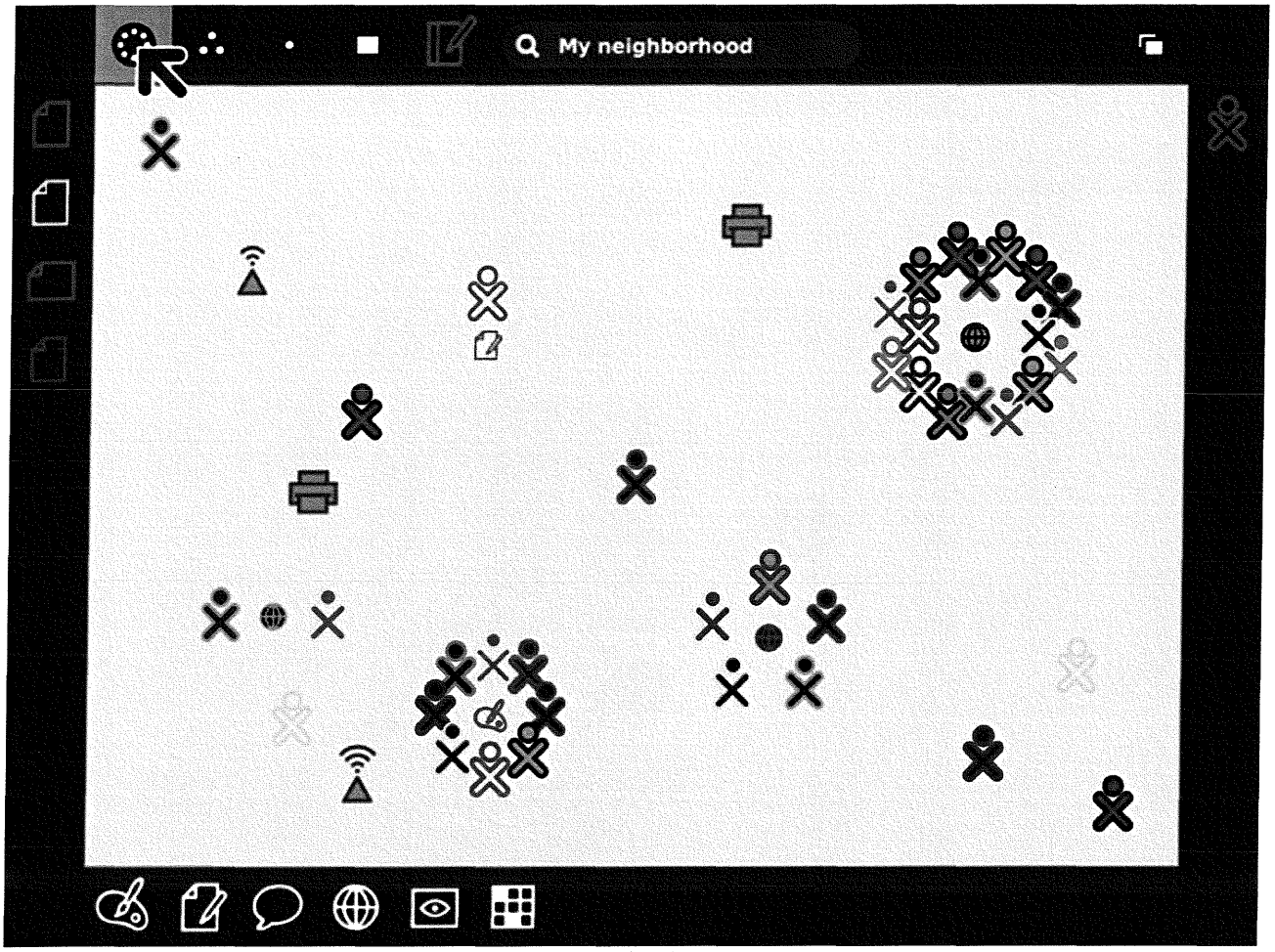


Figura 3: Vista de la interficie de Sugar (Xarxa de malla)

En aquesta última imatge podem veure el Journal. Aquí és on podem identificar tot el que ha realitzat l'usuari de la màquina, i en quin moment ho ha fet.

Podem veure, per exemple, que ha utilitzat el browser, ha fet fotos amb la càmera i ha xatejat amb un altre usuari.

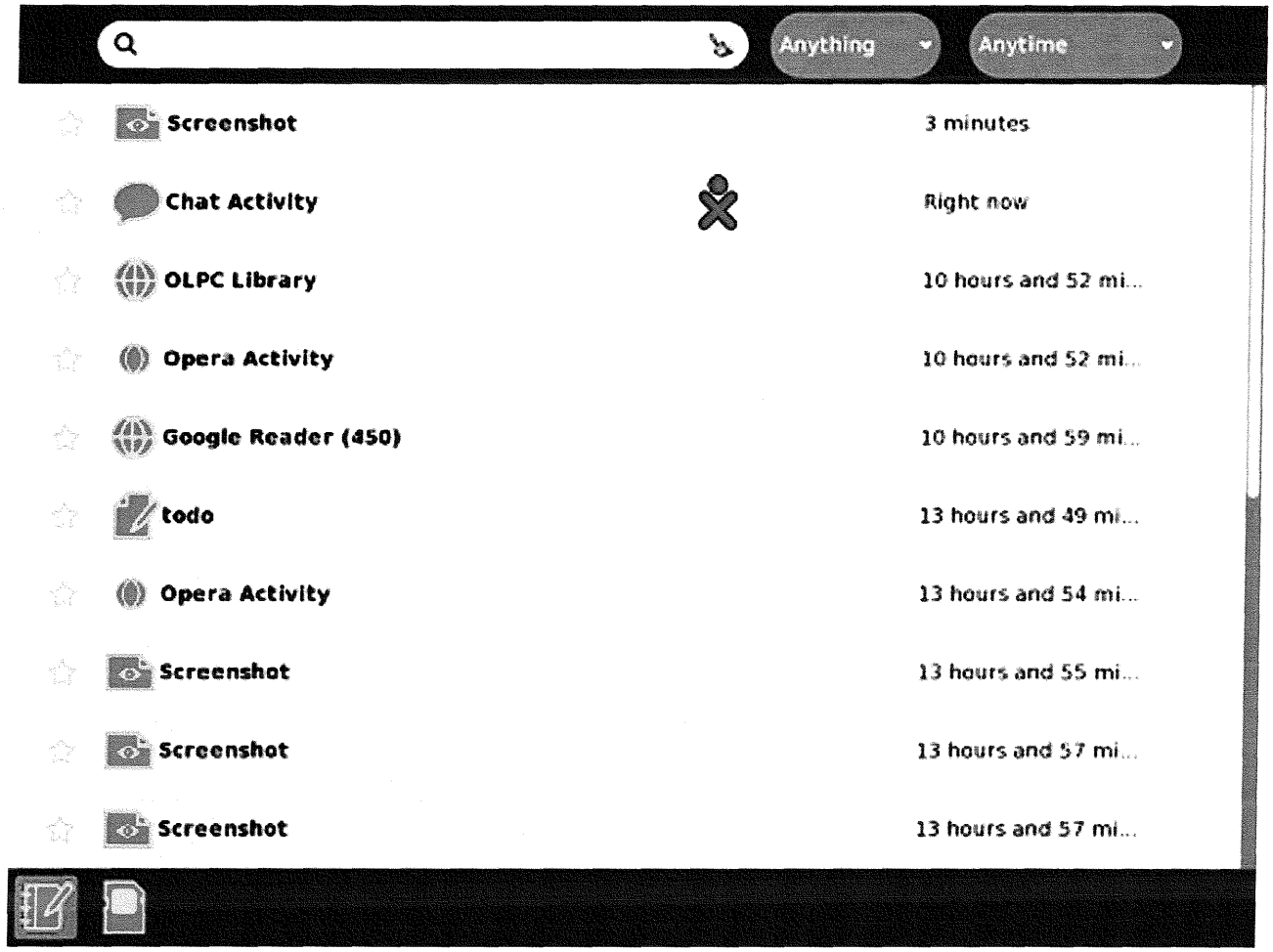


Figura 4: Journal

5. Disseny

Finalment, i per acabar amb el tema del projecte OLPC, cal parlar del disseny de la màquina. El disseny de l'XO és fonamental, donat que la primera impressió que tenim de les coses és la que entra primer per els nostres ulls, per tant el disseny que tenen, la manera de presentar-les al que ha de ser l'usuari final, en aquest cas els nens, ha de ser vistós.

Per tant, la màquina ha de tenir un aspecte juvenil i amb colors molt vius.

Durant la gestació de la màquina, aquesta va passar per molts canvis fins arribar al que finalment té, el que hem vist en la foto anterior.

Més endavant parlarem d'aspectes més tècnics, com son la configuració del Sistema Operatiu per al seu correcte funcionament, com són comandes bàsiques del shell, la configuració del llenguatge, del teclat, del llenguatge de programació, la manera de crear i codificar activitats, el funcionament del Journal, el funcionament de la xarxa de malla, entre moltes altres coses.

III. JCLIC

A. DEFINICIÓ

JClic és un conjunt d'aplicacions creades per el Departament d'Ensenyament de la Generalitat de Catalunya sota la Llicència Pública General GNU (GPL), que serveixen per a realitzar diversos tipus d'activitats educatives i escolars, tot jugant amb l'ordinador.

JClic suporta diversos tipus d'activitats, que poden anar des de trencaclosques, associacions, exercicis amb textos, mots encreuats, sopes de lletres, entre d'altres.

Els professors, amb ajuda d'una de les eines de JClic, l'Author, són els encarregats de crear projectes pels seus alumnes. Aquests projectes consten d'un conjunt d'activitats. Una activitat pot ser un trencaclosques o uns mots encreuats, entre molts d'altres. Cada activitat és com un joc que és utilitzat per l'alumne per aprendre diverses matèries com geografía, idiomes etc. Per exemple, un projecte realitzat per un professor podria ser titulat "El dofí", i aquest podria contenir un conjunt d'activitats com poden ser un trencaclosques, una sopa de lletres, etc, encarades a ensenyar que és, que menja, com és el dofí. A sota podem veure una imatge d'una activitat de les milers que existeixen per la xarxa.

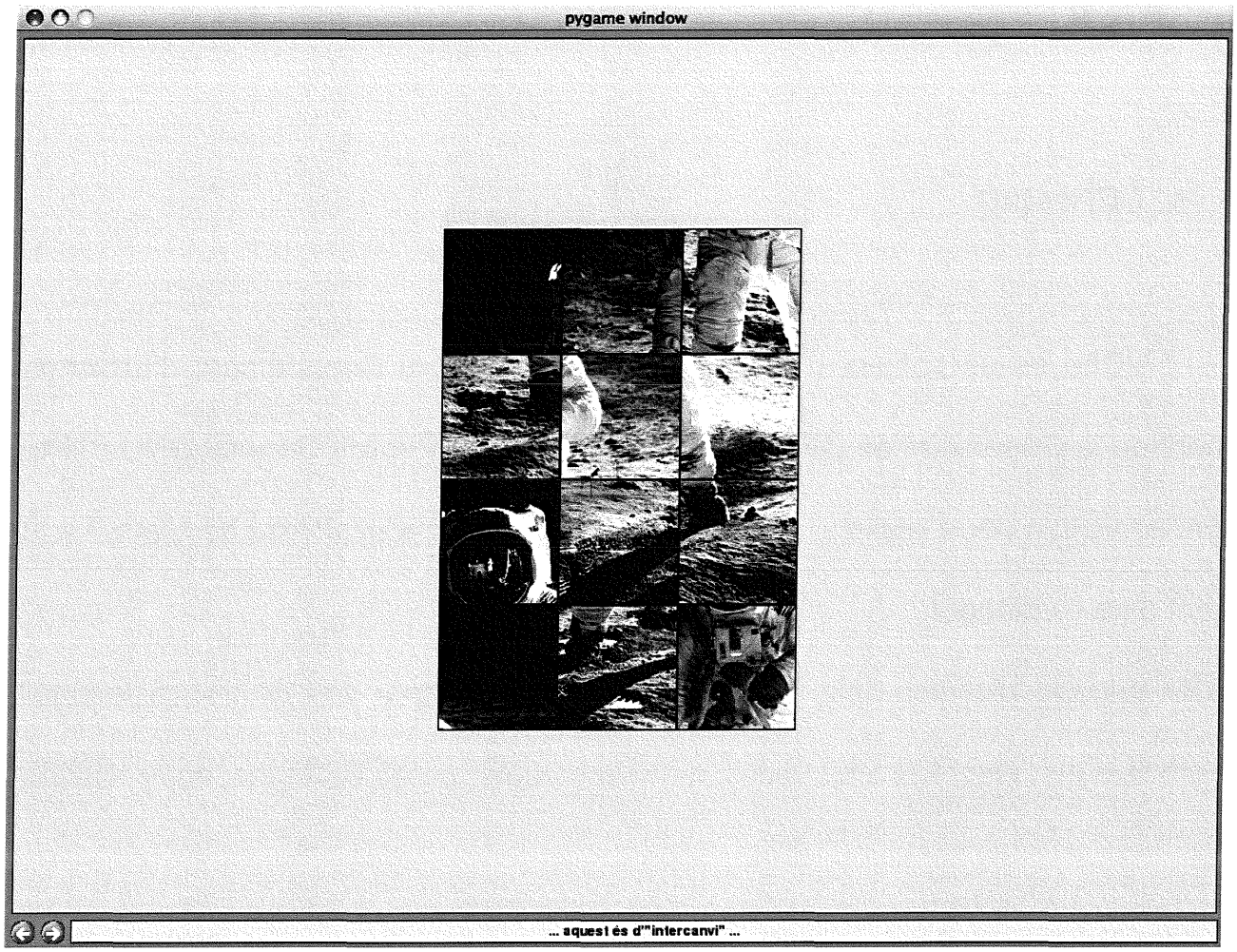


Figura 5: L'astronauta és una de les activitats de Jclic.

Els professors s'encarreguen de decidir l'ordre com apareixen les activitats per pantalla.

JClic té un antecessor, que és el Clic, creat el 1992 amb la finalitat de fer arribar la informàtica a totes les escoles catalanes. El projecte va funcionar i va començar a ser emprat per molts educadors, tant de Catalunya primer, com d'Espanya o fins i tot a l'estranger després. Va tenir una gran acceptació arreu.

Uns anys després va sortir una actualització del Clic, el Clic 3.0, i finalment, el JClic que és el que s'utilitza actualment.

B. CARACTERíSTIQUES

JClic és un projecte OpenSource (codi obert) que està desenvolupat en la plataforma Java i es pot utilitzar tant en Windows, MAC OS X, Linux o Solaris i com hem dit abans, està format per diferents components o mòduls que definim a sota:

JClic applet: El mòdul que permet executar l'aplicatiu en mode web. Únicament és necessari un browser, que tant pot ser Internet Explorer, Firefox, Safari, etc.

JClic player: Mòdul que s'ha d'instal·lar com una aplicació més del nostre ordinador. És el més important de l'aplicatiu, ja que permet executar els projectes creats per els professors i jugar als nens.

Els fitxers que contenen els projectes juntament amb les activitats que s'executen en el Player són de tipus .jclic, que internament és bàsicament un fitxer xml, amb diferents tags que defineixen la descripció completa del projecte, les activitats, les seqüències, els elements multimèdia (imatges i sons) i els diferents elements que gestionen l'activitat, entre d'altres.

El que fa el Player és identificar tots aquests elements i disposar-los a la pantalla de la forma que veiem en la imatge de sota.

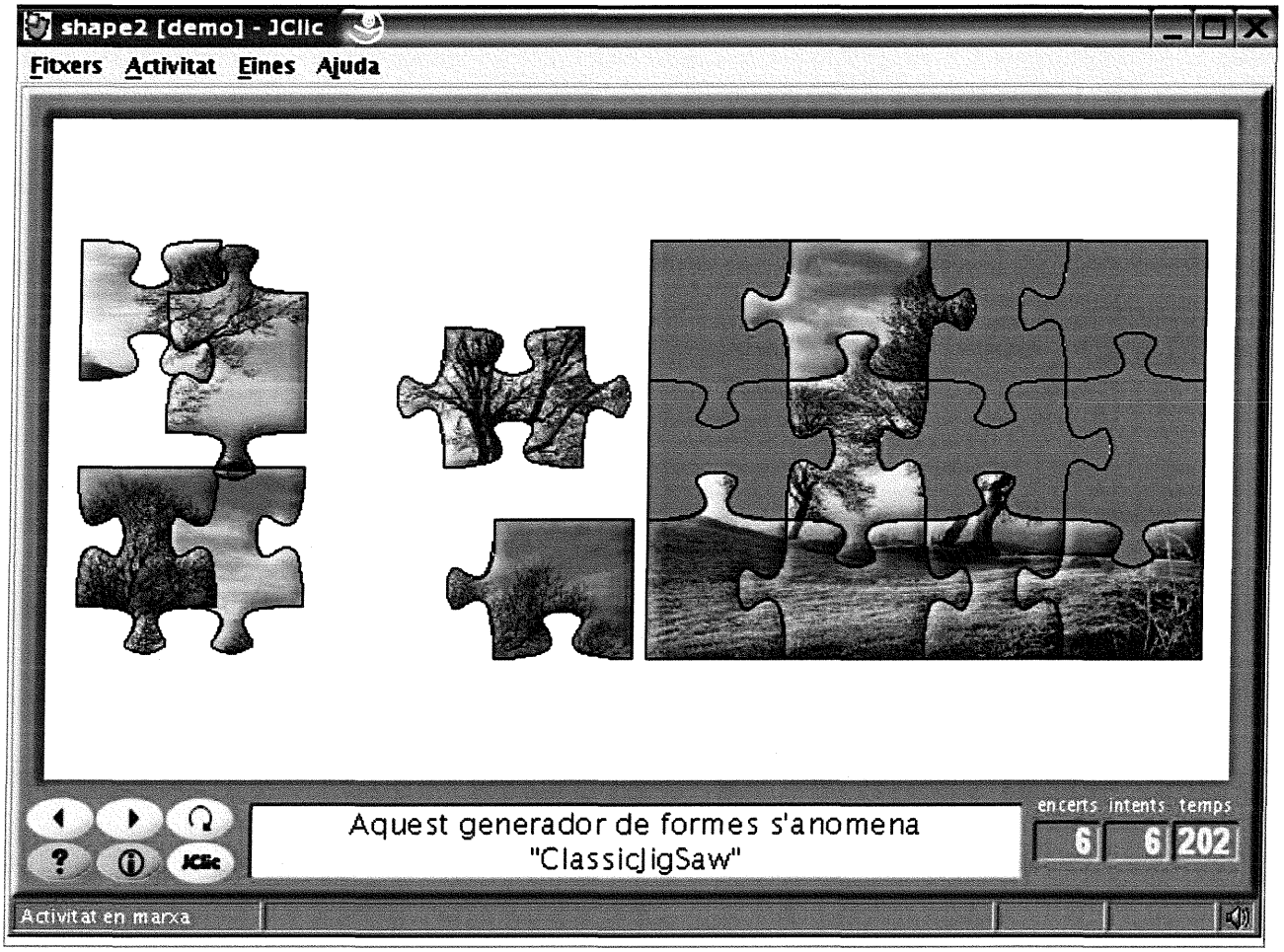


Figura 6: Player de JClic

JClic Author: és l'eina imprescindible per als professors. La que permet crear, editar i publicar les activitats mitjançant una interfície molt senzilla, visual i intuïtiva, com podem veure en la imatge de sota.

El que fa l'Author és, un cop dissenyada l'activitat amb aquest component, crea l'arxiu .jclic que necessita el Player per executar-la.

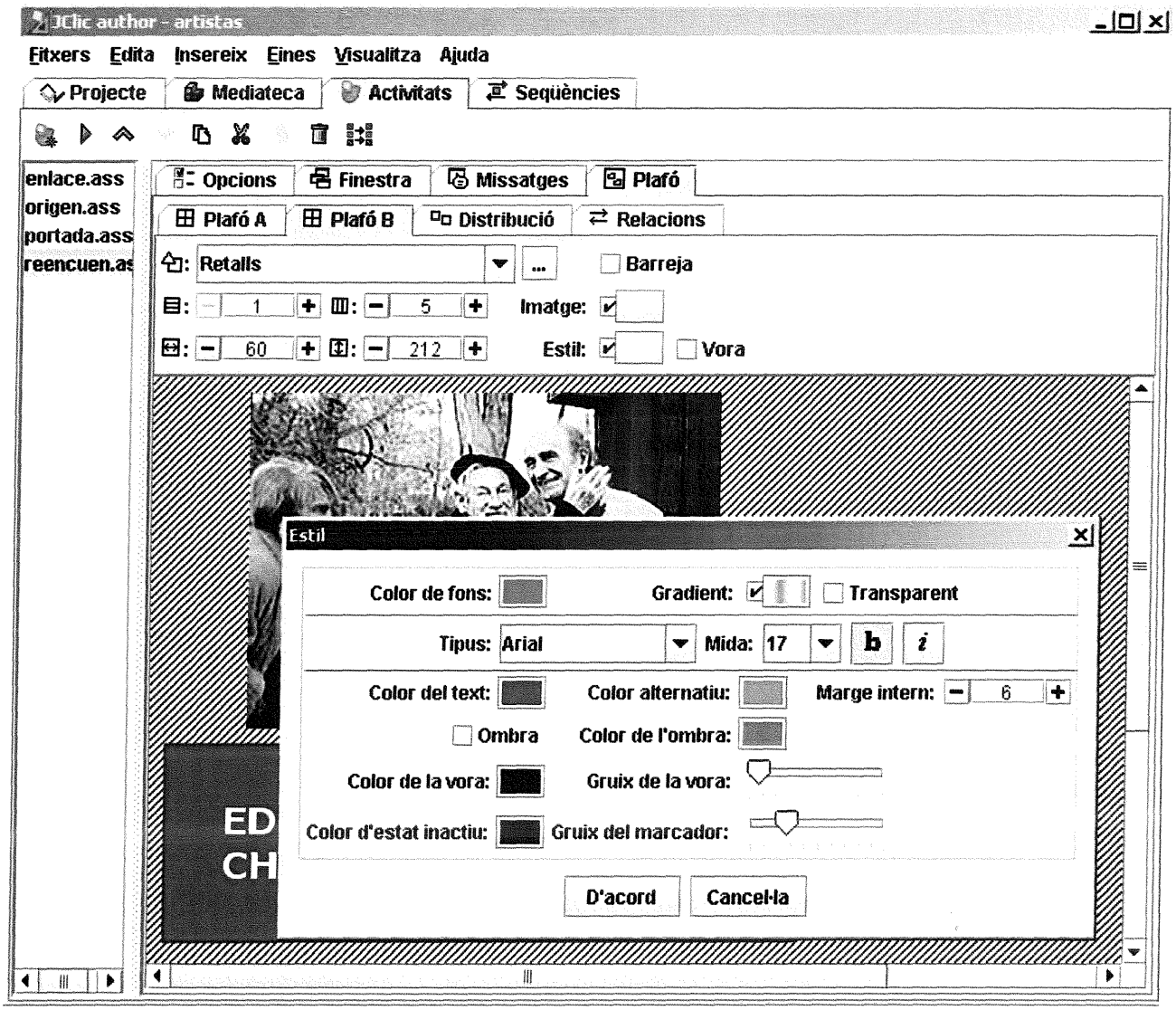


Figura 7: JClic Author

JClic Reports: és el component de recull de dades i generació d'informes (logs) sobre els resultats de les activitats fetes per els alumnes. Reports es basa en un esquema client-servidor. El servidor pot ser qualsevol ordinador d'una xarxa i els clients poden ser de dos tipus: les aplicacions JClic, que envien al servidor les puntuacions aconseguides i els navegadors web, des d'on es poden consultar els resultats.

C. ARXIUS JCLIC

Totes les dades de les activitats d'un projecte Jclic s'expressen i s'emmagatzemen utilitzant el llenguatge de marques XML.

A continuació s'explica l'estructura principal dels arxius .jclic que defineix les característiques del projecte.

L'element arrel dels arxius Jclic és <JClicProject> i conté 4 elements principals:

<Settings>: conté el tema, creadors, dates, localització, revisions, etc, del projecte

<Activities>: conté elements de tipus <activity>, que són les diferents activitats que té el projecte, amb les seves característiques i funcionament.

<Sequence>: descriu l'ordre amb el que s'han d'executar i presentar les activitats. També explicita el comportament dels botons d'avançar i retrocedir d'activitat.

<MediaBag>: és una relació dels diferents elements multimèdia necessaris per poder presentar l'activitat de forma correcta.

Els arxius JClic també es poden presentar comprimits en un arxiu .zip, que conté aquests més els elements multimèdia necessaris per poder executar-se.

IV. ENTORN DE DESENVOLUPAMENT

Per a poder aconseguir el nostre objectiu de traduir el Player de JClic per a la màquina XO necessitem, per una banda, un conjunt d'eines essencials per tal de poder codificar-lo en una màquina no XO, i per l'altra, una configuració bàsica del Sugar per a poder treballar de forma correcta, i varies aplicacions que ens ajudaran a fer més fàcil l'adaptació del Player, com explicarem més endavant.

En aquest apartat explicarem de forma detallada, com hem dit, les eines que se'ns oferien per a treballar, les utilitzades, els plug-ins utilitzats per a poder codificar en Python, les configuracions bàsiques per a què tot funcioni en les diferents màquines, problemes que hi han hagut, entre d'altres.

Començarem primer per descriure la configuració de les màquines amb les que hem treballat per a fer tota la implementació i seguirem per la de la màquina XO. De totes formes aquest segon cas el podriem dividir en dos, ja que en primera instància, vem virtualitzar el SO de l'XO a les nostres màquines per a tal d'aprendre el seu funcionament i poder fer les primeres proves, i finalment la màquina XO real, on hem acabat d'adaptar l'aplicatiu.

A. L'ENTORN A LES NOSTRES MÀQUINES

El primer pas que s'havia de realitzar per a poder començar la traducció de l'aplicatiu JClic era buscar un entorn de desenvolupament (IDE) idoni per a les nostres màquines.

Les màquines de que hem disposat durant la realització del projecte han estat dos portàtils Apple, un Macbook i un Macbook Pro, així que vem haver de buscar una IDE que funcionés de forma correcta dins el Mac OS X Leopard i que permetés la implementació en Python.

En un primer moment, es va pensar que seria més complicat trobar un aplicatiu adequat en Mac que en Windows, però de seguit vem veure que el Sistema Operatiu d'Apple ja disposa de molts i variats entorns dels quals vem poder escollir.

Després de buscar per la xarxa diferents opcions, es va decidir optar o per NetBeans o Eclipse, tots dos Open Source, encara que aquests, per defecte, no disposen de Python com a llenguatge d'alt nivell per a codificar.

El fet que s'escollís entre aquests dos en comptes d'un d'especialitzat per a Python, és que els primers són els dos més coneguts i potents que existeixen i que disposen de moltes opcions i diferents mòduls per tal de poder compilar, executar, debugar, crear executables, afegir-hi una gran varietat de plug-ins, entre moltes altres. En canvi, algun IDE trobat exclusivament per a Python, no disposava de tantes opcions com NetBeans o Eclipse.

Abans de fer una breu explicació dels dos entorns cal comentar que el Sistema Operatiu de Mac Leopard ja disposa d'una versió de Python, la 2.4, instalada, però mirant els repositoris, vem veure que ja existia una versió 2.5, així que la vem actualitzar. Hem treballat així, amb la versió 2.5 del Python. Més endavant explicarem els plug-ins utilitzats (en especial el Pygame).

A sota podem veure la captura de pantalla on es pot veure la versió de Python utilitzada.

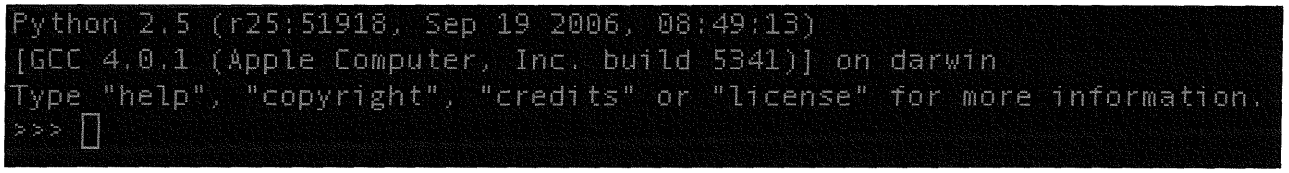


Figura 8: Versió de Python en Mac OS X Leopard

Leopard també disposa, de forma nativa, juntament amb el Python 2.4, d'una petita IDE per a codificar en aquest llenguatge, però de seguida va ser descartada, ja que es tracta d'un software bastant rudimentari i poc àgil d'utilitzar.

Passem ara fer una breu descripció dels dos entorns de desenvolupament que vem escollir i després explicarem quin dels dos vàrem triar i el per què.

1. Netbeans 6

NetBeans és una plataforma per el desenvolupament d'aplicacions d'escriptori usant Java com a llenguatge d'alt nivell per defecte i a un entorn de desenvolupament integrat (IDE). És un entorn Open Source.

La plataforma NetBeans permet que les aplicacions siguin desenvolupades a partir d'un conjunt de components software anomenats mòduls.

És una base modular i extensible utilitzada com una estructura d'integració per crear les aplicacions. El que permet aquesta plataforma és extendre NetBeans de forma que aquestes s'integrin fàcilment i que puguin també utilitzar-se per desenvolupar aplicacions que no siguin només Java, sinó que puguin acomodar altres tipus de llenguatges d'alt nivell, com pot ser el Python, mitjançant la utilització de diferents plugins.

Aquesta plataforma ofereix serveis comuns a les aplicacions d'escriptori, permetent al desenvolupador enfocar-se en la lògica específica de la seva aplicació.

Entre les característiques de la plataforma hi ha:

- Administració de les interfícies d'usuari
- Administració de les configuracions d'usuari
- Administració de l'emmagatzematge
- Administració de finestres
- Framework basat en assistents

En la imatge de sota podem veure un snapshot de NetBeans, amb els diferents mòduls dels que disposa, com l'explorer, la finestra principal de codificació, el compilador, el debugger, entre d'altres.

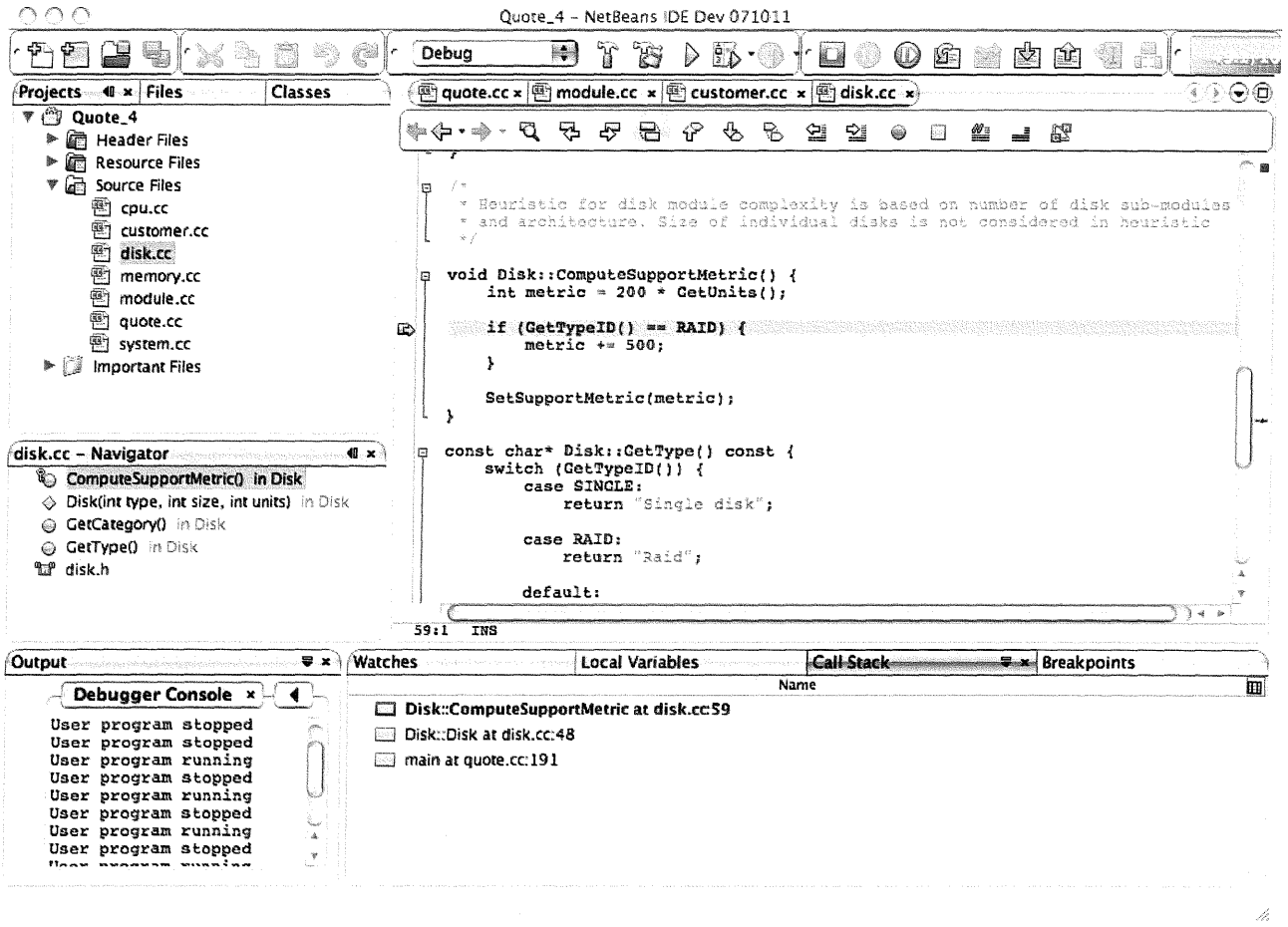


Figura 9: NetBeans

2. Eclipse 3.3

Com en el cas de NetBeans, Eclipse és un IDE per el desenvolupament d'aplicacions d'escriptori basat en Java de codi obert multiplataforma.

Eclipse també es una comunitat d'usuaris, extenent constantment les àrees d'aplicació cobertes.

També permet, com en el cas de NetBeans, la integració de diferents plug-ins per tal d'extendre les seves funcionalitats i permetre la implementació d'aplicacions codificades en un llenguatge diferent de Java, a part d'altres millores.

A sota podem veure la pantalla principal d'Eclipse, amb un disseny molt semblant a la de NetBeans.

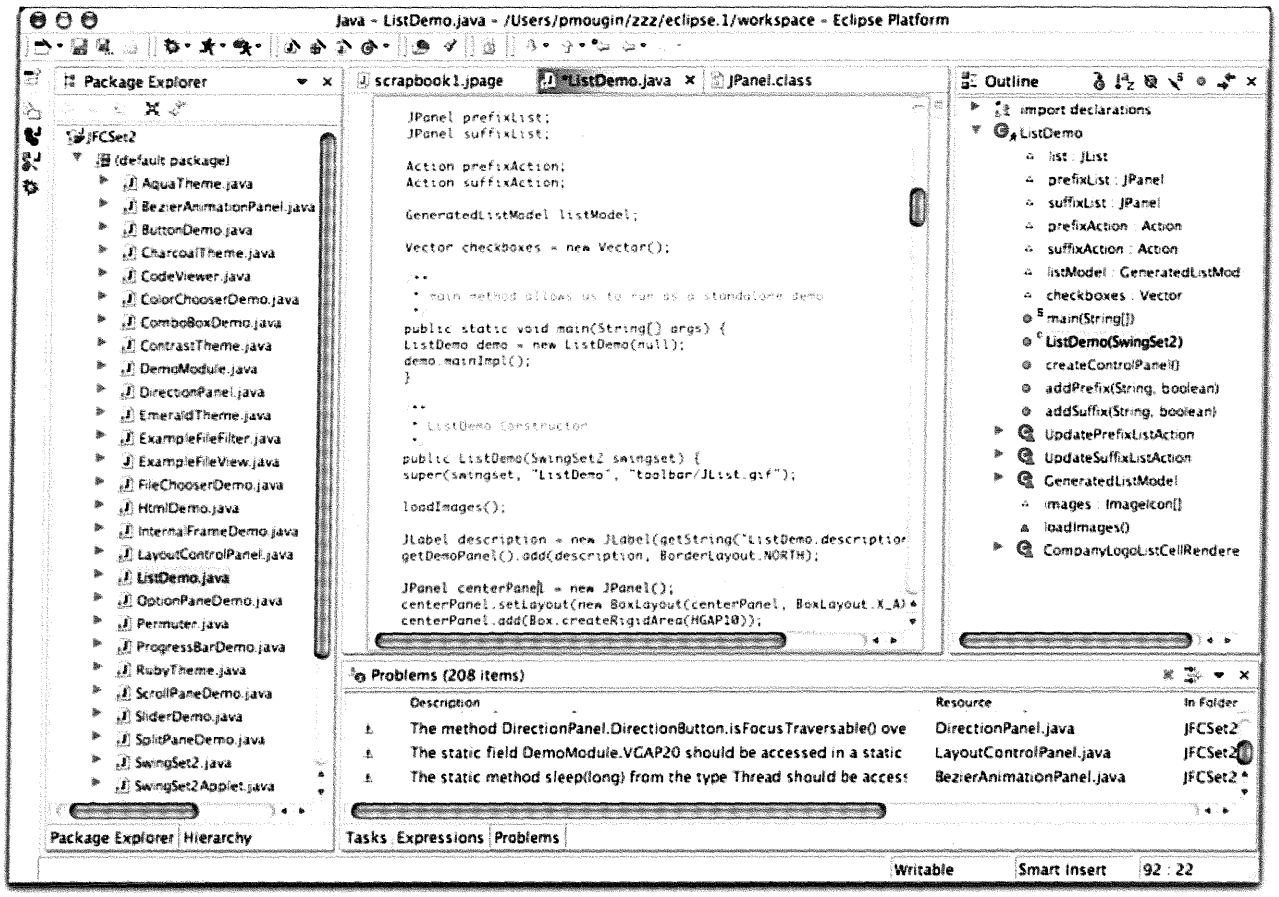


Figura 10: Eclipse

Després d'instal·lar els dos aplicatius, i configurar-los per a què poguessin compilar i codificar en Python, amb els seus respectius plug-ins (Pydev en l'Eclipse i en NetBeans), que explicarem a continuació, la primera cosa que es va fer és una prova, implementant un petit programet tipus "Hello World" en Python.

Com que la diferència entre els dos IDE's per a nosaltres no era molt significant, el que ens va fer decidir per un d'ells, és que en un primer moment, l'Eclipse ja va compilar i exectuar de forma correcta el programa i el NetBeans ens va donar diferents errors només compilar.

Al ser dos aplicatius de més o menys la mateixa potència i de no voler perdre molt de temps en instal·lar els components bàsics, vàrem triar el què en un primer moment va executar la petita apliació.

Així doncs, vàrem escollir Eclipse per a la traducció de JClic.

Tot seguit explicarem els plugins instal·lats tant en Mac OS X com a Eclipse per a poder implementar el nostre projecte en Python.

3. Configuració de la màquina i Eclipse

Un cop escollit l'entorn de desenvolupament que utilitzariem per desenvolupar el nostre projecte, vem haver d'adequar la màquina a les necessitats que teniem.

Primer de tot es van haver d'instal·lar un seguit de llibreries al Python 2.5 que prèviament haviem configurat a la màquina.

La llibreria principal que necessitàvem era Pygame, la llibreria de Python que permet implementar jocs i que treballa amb gràfics majoritàriament. En un capítol posterior l'explicarem en detall.

També necessitàvem un parssejador d'XML en Python, ja que havíem de tractar els arxius .jclic i la seva informació, XML's en definitiva, per això necessitàvem una llibreria que ens ho permetés fer. Aquesta és PyXML, ja que conté DOM, un parssejador XML.

Per a poder configurar Python i integrar-hi Pygame i PyXML vem haver d'instal·lar un seguit de plug-ins i llibreries que detallem a sota:

L'element imprescindible, el **Python 2.5**, que anteriorment havíem configurat en la nostra màquina, a partir de la versió 2.4 nativa que portava el nostre ordinador.

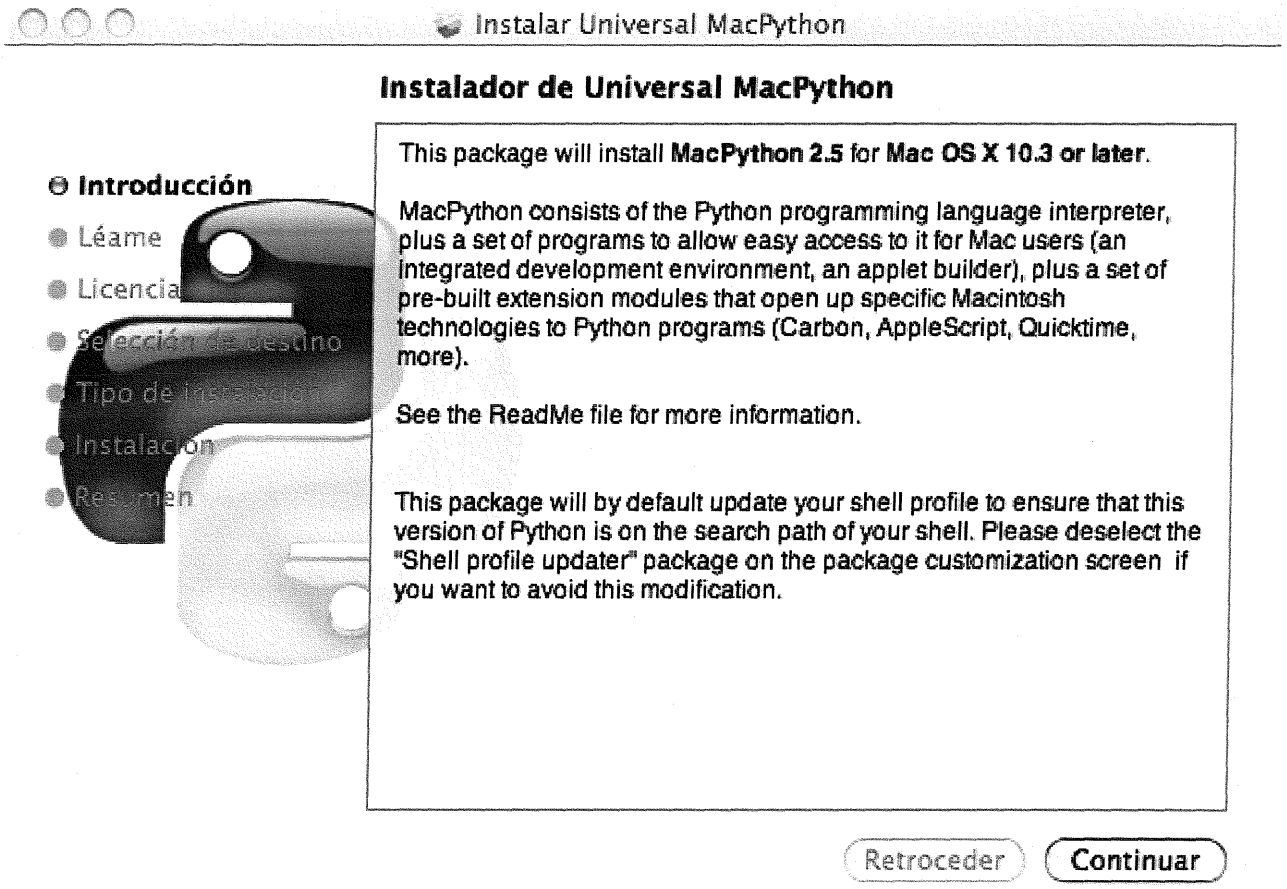


Figura 11: Instal·lador de Python 2.5

Les llibreries que anomenem a continuació són addicionals al Python 2.5 que cal instal·lar per poder desenvolupar aplicacions.

PIL 1.16 - Python Image Library. Aquesta llibreria afegeix a Python la capacitat de processament d'imatges i gràfics, de molts tipus de formats de fitxers.

NumPy 1.0.3.1 és un package fonamental en Python, necessari per al càlcul científic. Conté, entre d'altres, un objecte vector N-dimensional, funcions lineals algebràiques, transformacions de Fourier i permet un treball sofisticat amb nombres aleatòris.

Aquesta llibreria té 2 subllibreries que també vem instal·lar, que afegeixen petites millores i funcionalitats. Aquestes son la **Numeric 24.2** i la **NumArray 1.5.2**.

PyObjC 1.4 proveeix un bridge (pont) entre els llenguatges Python i C. Aquest pont és bidireccional, permetent al programador de Python utilitzar tot el potencial que el C permet i a l'inrevés, de manera totalment transparent.

wxPython 2.9.3.0 és una eina de creació d'interfícies gràfiques d'usuari (GUI) per a Python. Permet als programadors crear programes amb una interfície gràfica funcional i robusta. Com totes les llibreries anteriorment citades, és Open Source.

Aquesta llibreria és imprescindible per a poder implementar menús, finestres, botons, etc.

Pygame 1.7.1 és la llibreria més important per al nostre projecte, ja que és la què permet a Python la implementació de jocs a partir d'un conjunt de mòduls de Python. És la llibreria que permet tractar amb gràfics, figures, events de teclat, de

mouse, elements multimèdia, etc. En un capítol posterior parlarem d'aquesta llibreria amb més precisió i explicarem els diferents mòduls dels què disposa.

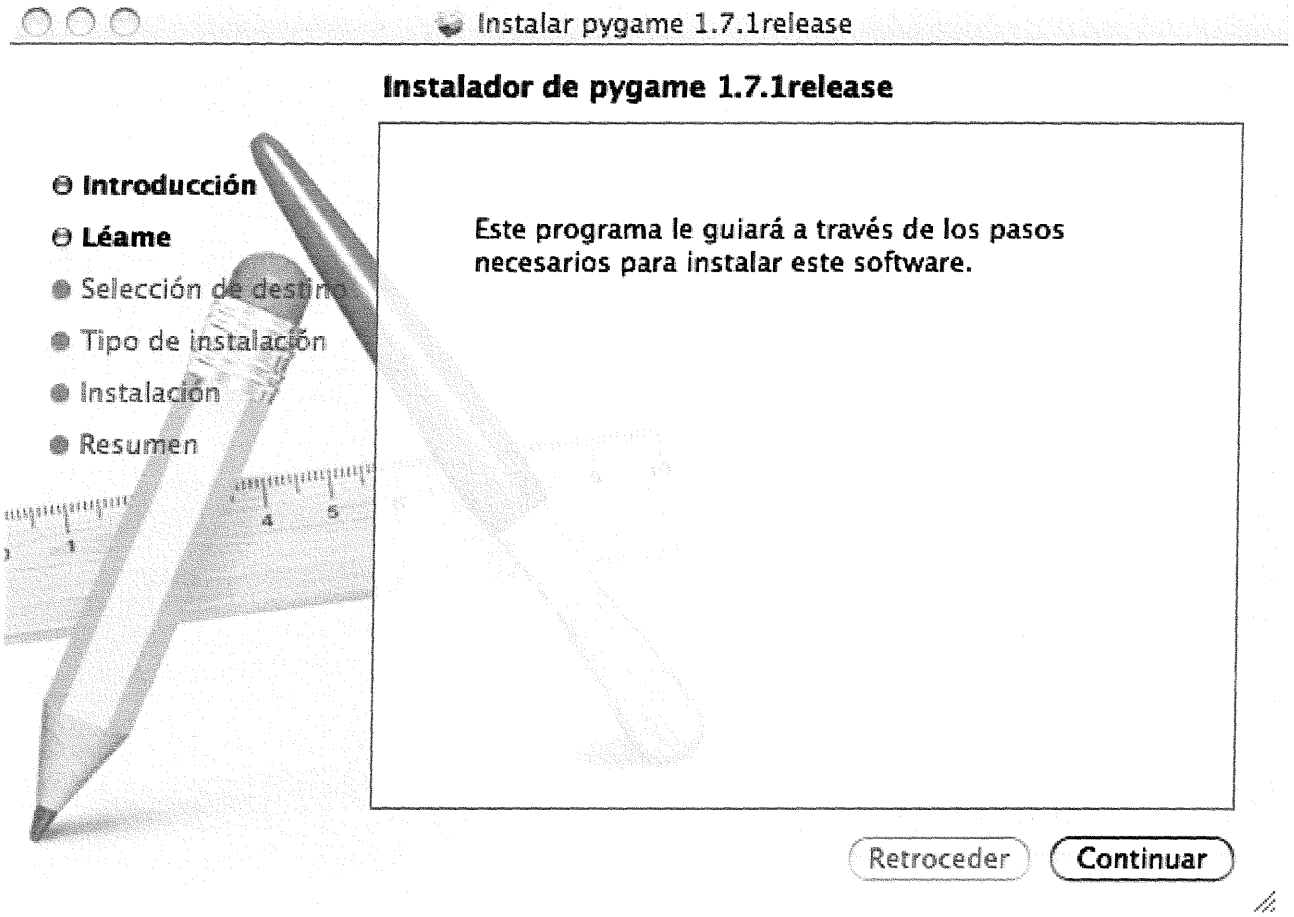


Figura 12: Pantalla d'instal·lació de Pygame

PyXML: És la llibreria de Python que permet tractar amb arxius XML, fent el parsseig i el posterior tractament de la informació. Aquesta eina té diferents mòduls, on els més importants són el DOM i el SAX.

Un cop instal·lats tot aquest conjunt de llibreries i mòduls es va passar a integrar a la nostra màquina l'Eclipse i a configurar el plug-in pertinent per tal de dotar-lo del mòdul per a la creció de programes en Python.

El plug-in necessari per aconseguir-ho és **PyDev 1.3.24**. Aquest permet als programadors usar Eclipse per a la implementació de Python i Jython, transformant Eclipse en un potent IDE de Python. Conté moltes funcionalitats, com la completació de codi, analitzador de sintàxis, refactorització, debugging i molts d'altres.

Per a poder adaptar PyDev a Eclipse s'han de seguir els següents passos:

Un cop instalat Eclipse, executar-lo i anar a la pestanya Help i clicar al menú "Find and Install", com es pot veure a la imatge de sota.

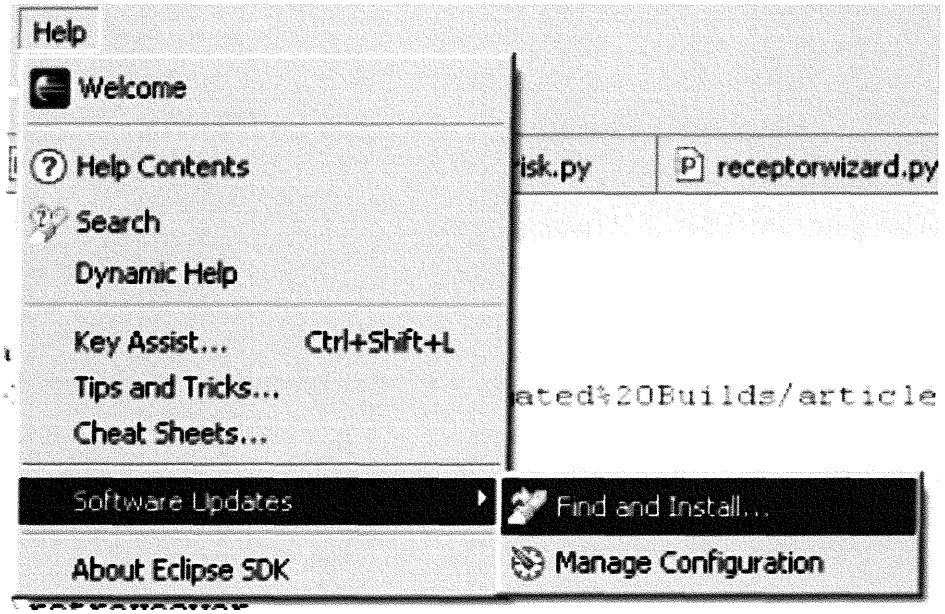


Figura 13: Instal·lació PyDev

Seguidament, seleccionar “Search for new features for install”.

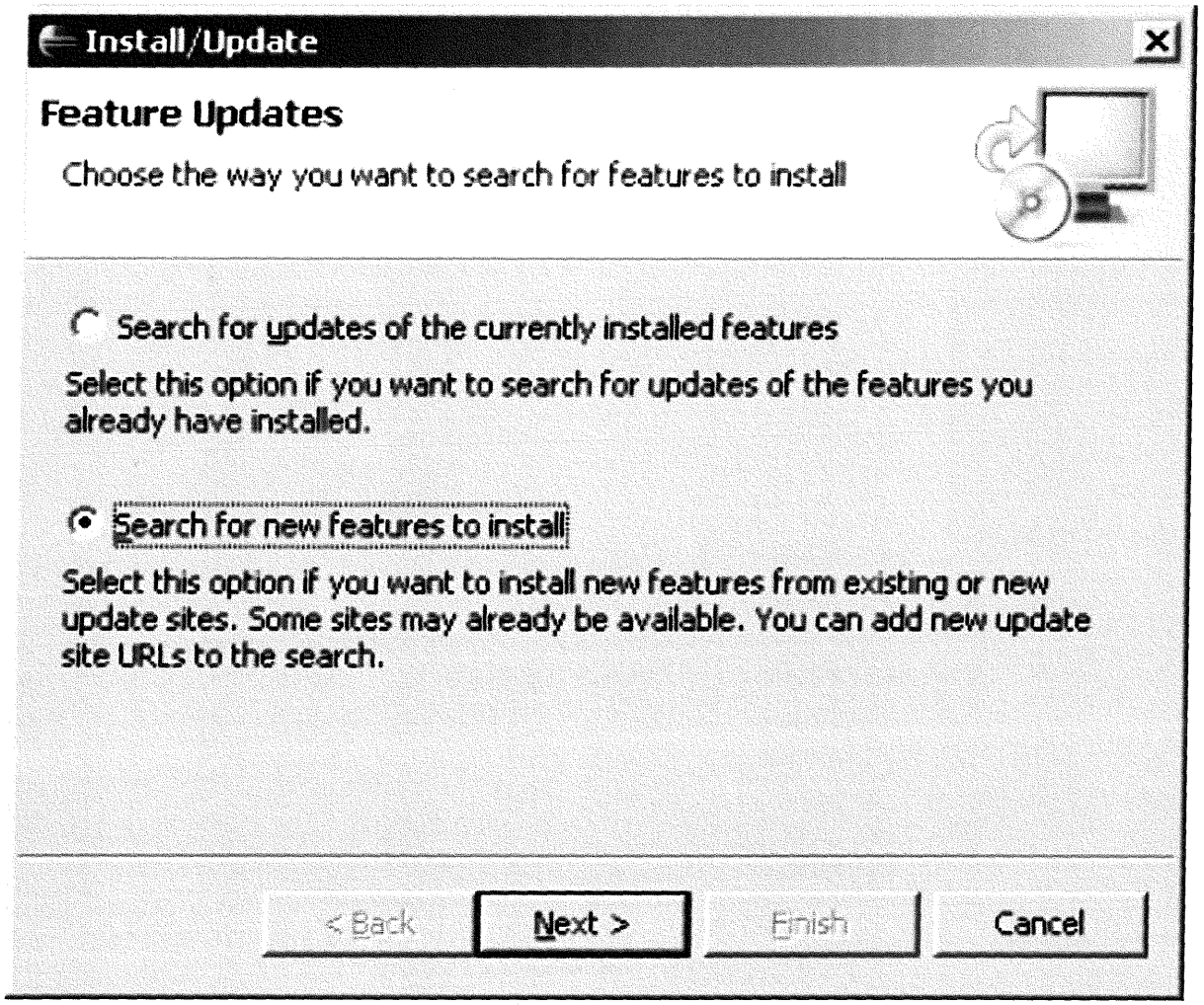


Figura 14: Instal·lació PyDev

En la següent pantalla, clicar a “new remote site”.

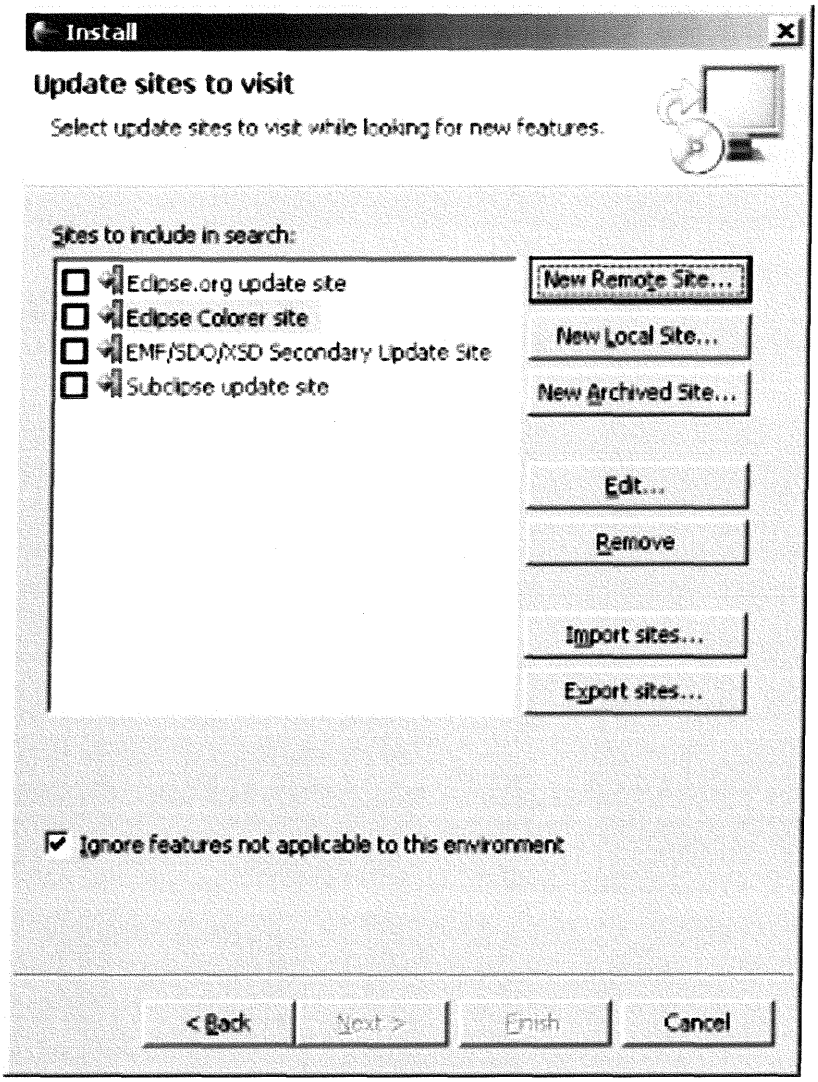


Figura 15: Instal·lació PyDev

En aquest moment hem d'omplenar el lloc d'actualització "Pydev Extensions" amb la seva URL pertinent.

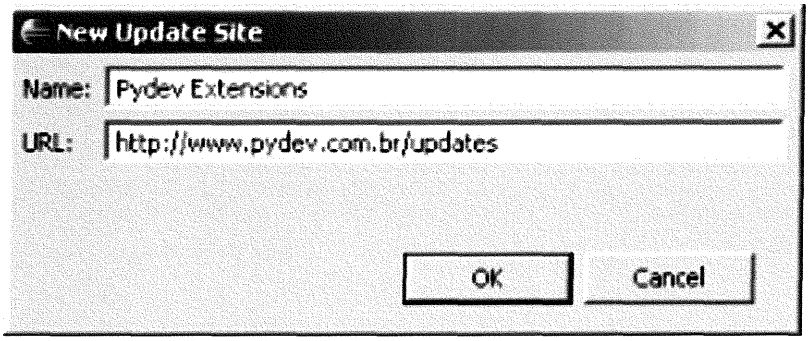


Figura 16: Instal·lació PyDev

Clicar “Finish”. S’hauria de veure una pantalla com la següent.

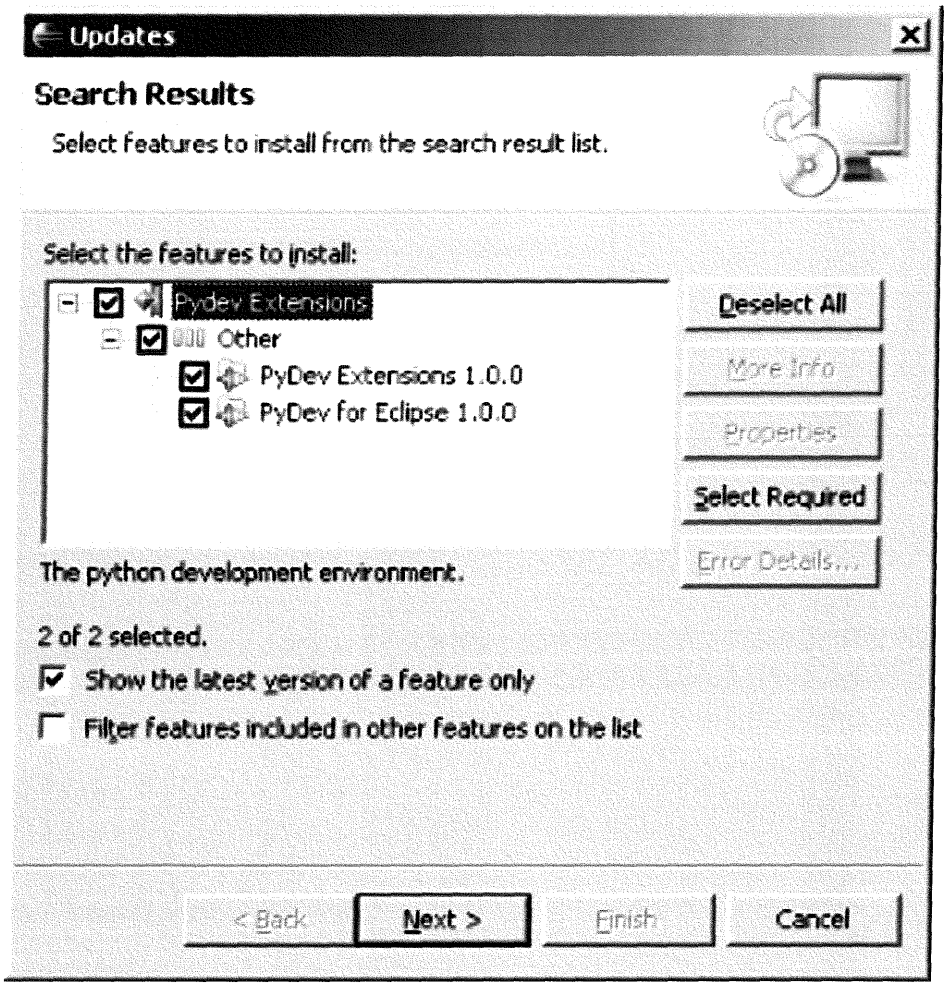


Figura 17: Instal·lació PyDev

S’han de seleccionar totes les opcions i clicar “Next”.

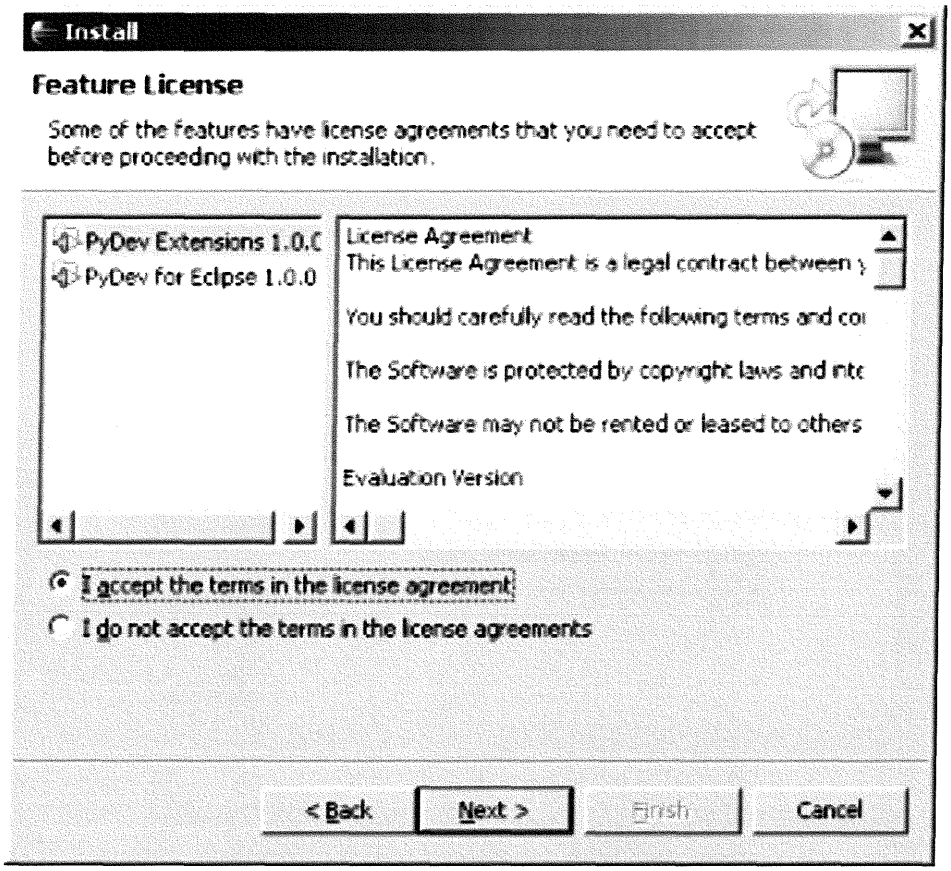


Figura 18: Instal·lacio PyDev

Arribats en aquest moment, s'ha d'acceptar la llicència, clicar "Next" i en la següent pantalla, clicar "Finish".

En aquest moment ja tenim instalat PyDev en Eclipse, i ja podem crear el primer projecte per a implementar en Python a Eclipse, però encara ens falta fer una cosa per què tot funcioni correctament, i és actualitzar Eclipse per tal que els plugins instal·lats a la màquina funcionin també en Eclipse, com són el cas del PyGame, PyXML i tots els altres.

El que hem de fer és anar al menú principal i clicar "Preferences". Un cop allà ens apareixerà una opció anomenada "PyDev". Cliquem en aquesta opció i allà, a la opció "Interpreter - Python".

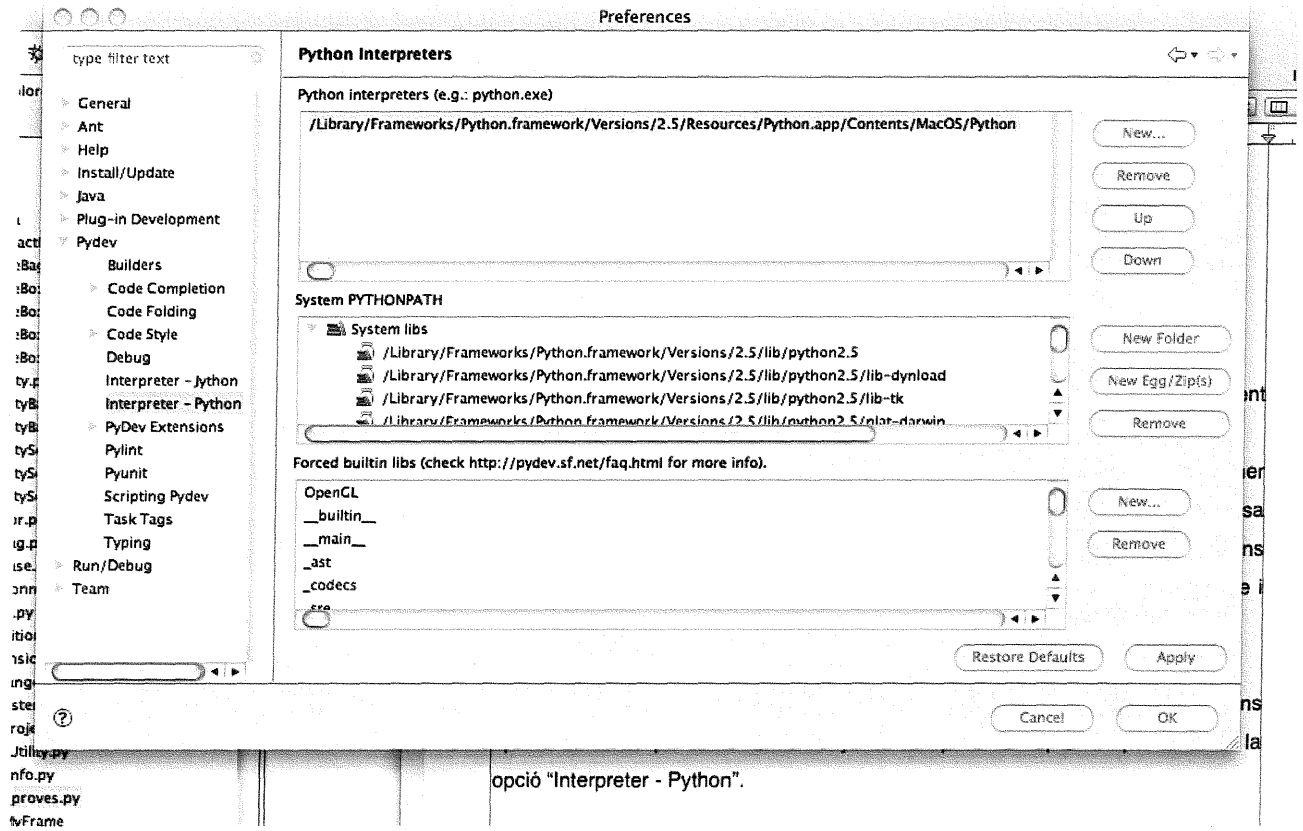


Figura 19: Configuració PyDev

En aquest moment, hem de borrar l'intèrpret que té per defecte amb el botó "Remove" i seguidament clicar a "New". Aquí hem de buscar en el nostre sistema l'arxiu de l'intèrpret de Python i agregar-lo.

Un cop fet això, clicar "Apply" i ens configurarà Eclipse amb Python i totes les llibreries i mòduls instal·lats.

Finalment cliquem "OK", reiniciem Eclipse i ja podem utilitzar-lo per a crear el nostre primer projecte en Python.

4. Crear el primer projecte en Python amb Eclipse

Per a crear el nostre primer projecte en Python amb Eclipse, el primer que hem de fer és exectuar Eclipse.

Un cop executat, hem d'anar a File - New Project i clicar "PyDev Project".

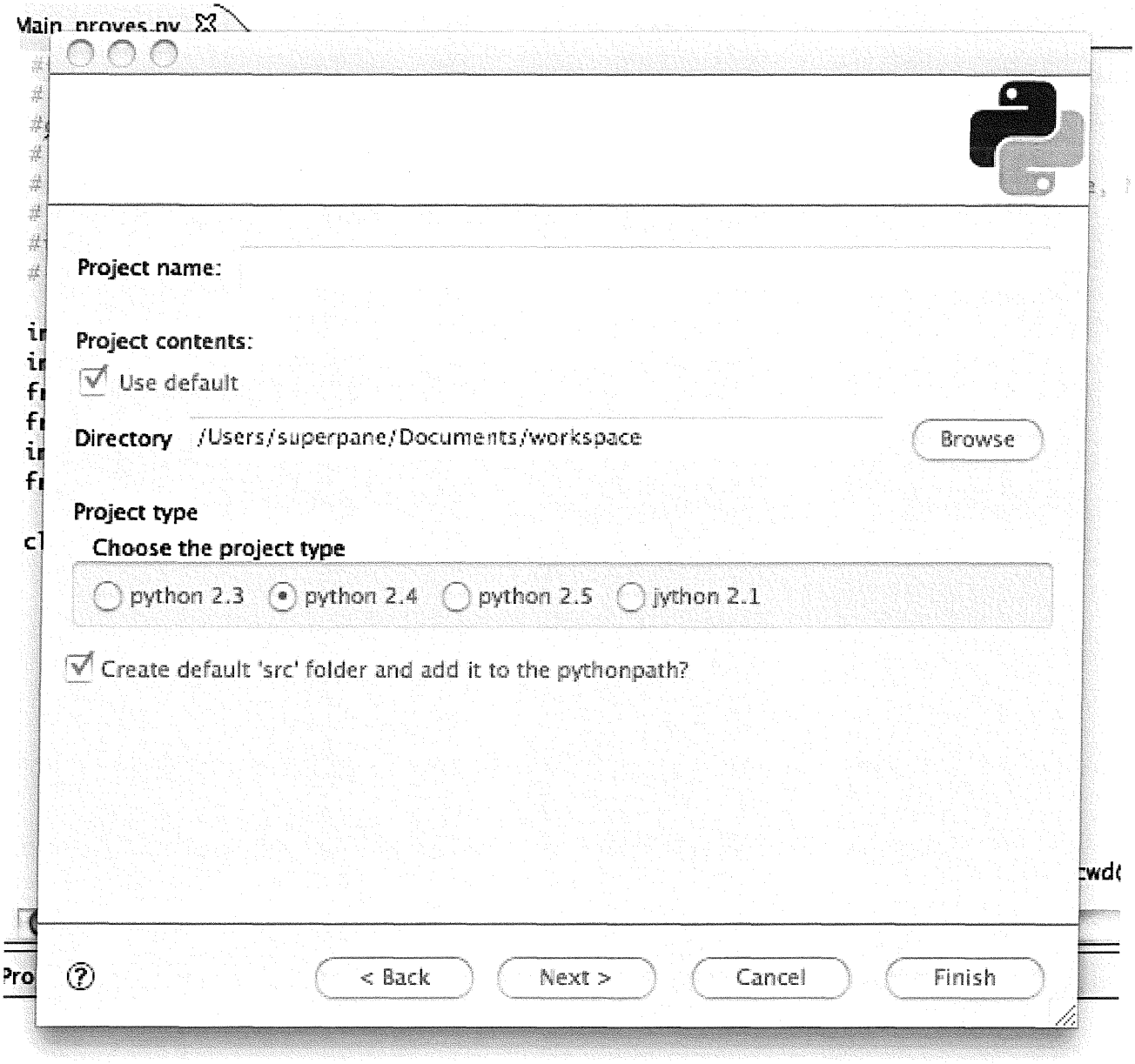


Figura 20: Primer projecte amb PyDev

En aquest punt, hem d'esciure a "Project Name" el nom que volem donar al nostre projecte, en "Project Type" escollir Python 2.5 i finalment clicar "Finish".

El projecte ja està creat, i ja podem començar a codificar. La pantalla que tindrem al crear el projecte (prova1) serà la següent:

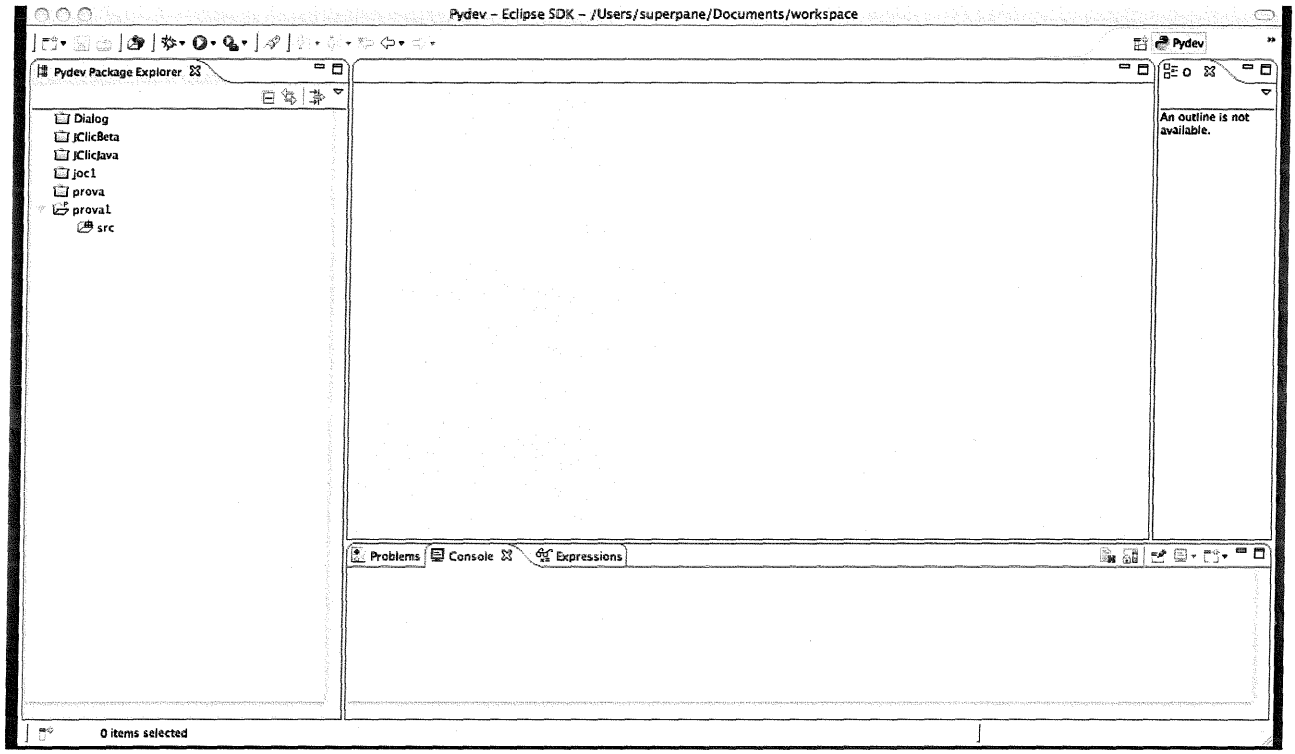


Figura 21: Primer projecte en PyDev

Un cop tenim el projecte creat, ja podem afegir classes al projecte, que seran arxius amb extensió .py, que s'agregaran dins la carpeta *src* del nostre projecte.

Per afegir una classe al nostre projecte, hem de clicar amb el botó dret del mouse a sobre la carpeta *src*, i anar a New - File.

Es poden crear els fitxers que es volen. També es poden crear subcarpetes, subprojectes, etc.

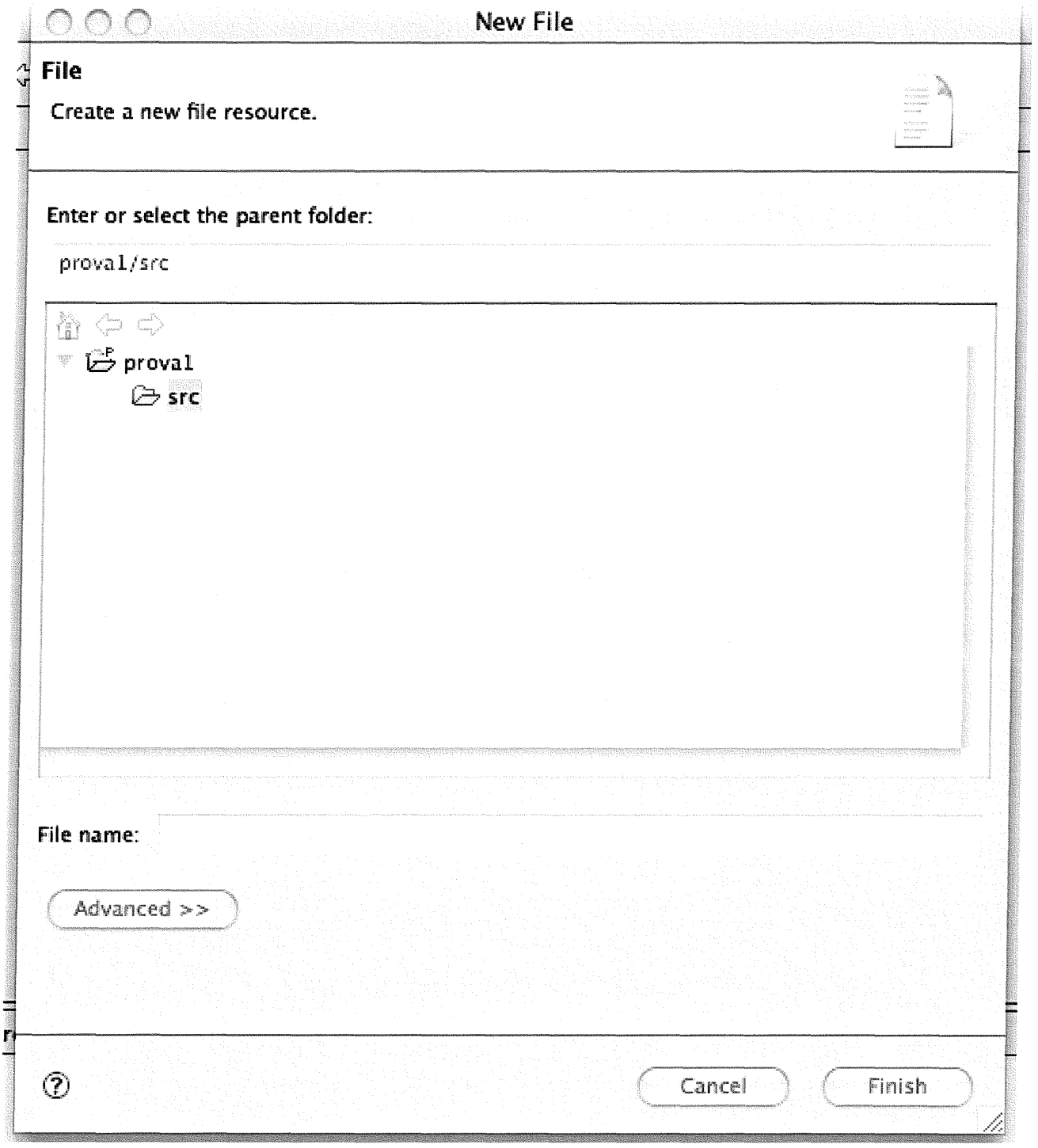


Figura 22: Creació d'una nova classe

Aquí hem de posar el nom que tindrà la classe a "File name". Ja la tenim creada i podem començar a codificar.

Quan ja hem codificat la aplicació que volem, es pot passar a compilar o debugar, segons les nostres necessitats.

Això es fa anant a la opció del menú Run i dins d'aquesta, clicar Run o bé Debug.

Si hem decidit executar l'aplicació i tot ha anat bé, s'executa de forma correcta. Si hi ha algun error, el compilador ens ho fa saber a la part inferior de la pantalla.

B. PYGAME

Al ser la llibreria més important i la que més hem utilitzat en el nostre projecte durant tot el procès d'implementació, explicarem d'una forma més detallada **Pygame** i les seves característiques i peculiaritats.

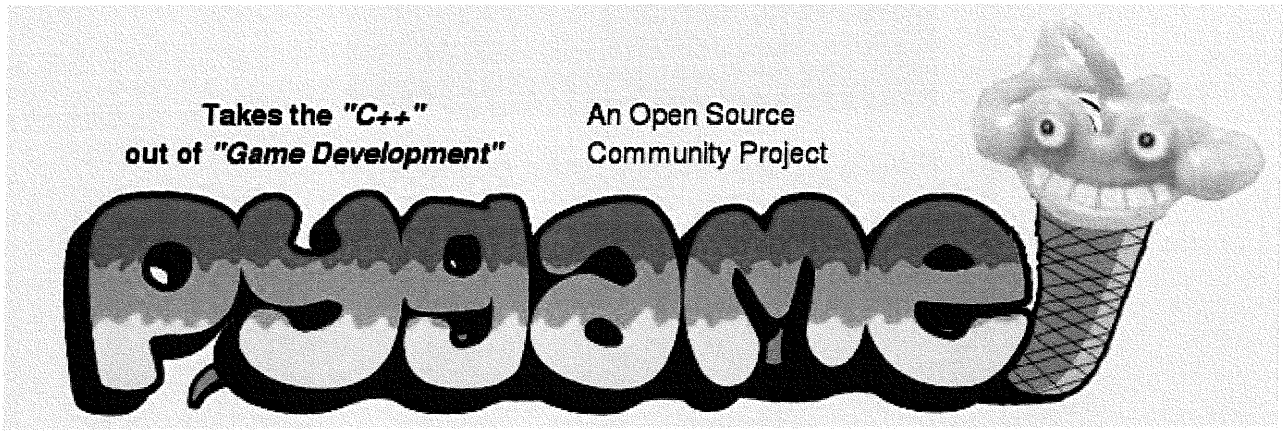


Figura 23: Logo de Pygame

Pygame, com hem dit anteriorment, és una llibreria de Python que permet la implementació de jocs a partir d'un conjunt de mòduls, que tracten tant gràfics, com figures, events de tot tipus, elements multimèdia, etc.

És la llibreria perfecta per al desenvolupament de jocs de poca envergadura. És capaç d'executar-los a 30 fps, o sigui, que no és molt potent, però ens és molt útil en la implementació de petits jocs, així que s'adequa perfectament a les nostres necessitats, ja que els jocs de JClic no son especialment complicats i amb necessitat de molta potència gràfica.

En les imatges de sota podem veure dos exemples de jocs creat amb Pygame.

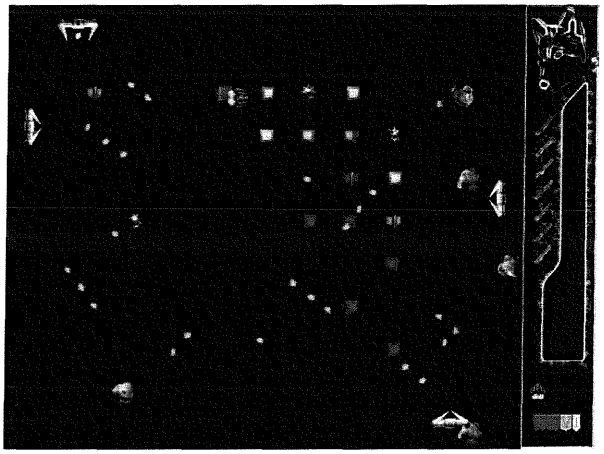


Figura 24: Joc implementat en Pygame

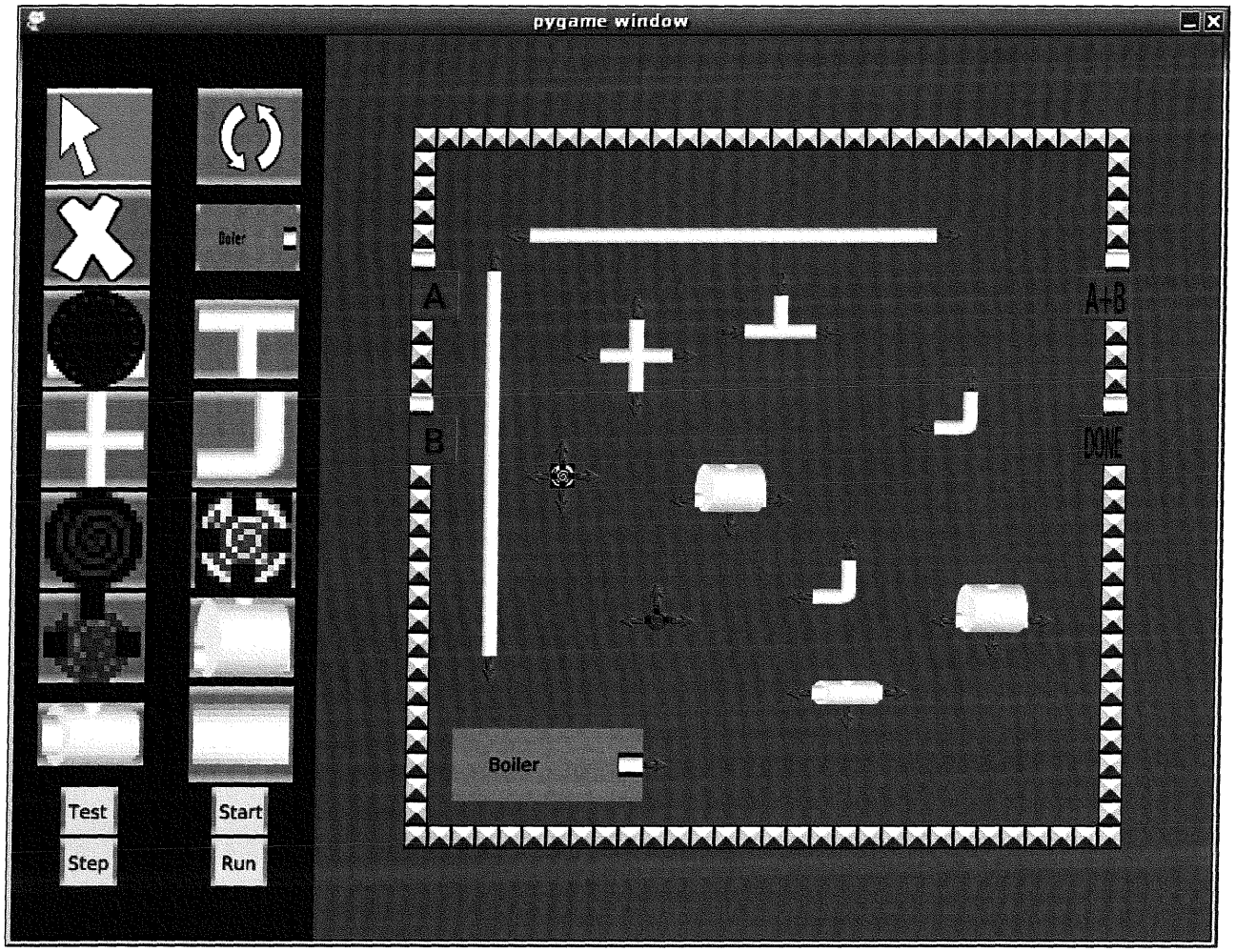


Figura 25: Joc implementat en Pygame

La llibreria Pygame està formada per diferents mòduls, que identifiquen els diferents elements que pot tenir un joc, com són els sons, imatges, superfícies, events, fonts, etc, i les seves operacions pertinents. Ara farem una breu explicació de cada una.

1. Mòduls de Pygame

a) Pygame

És el mòdul pare, el que engloba tots els altres, que seran fills d'aquest, i que conté les funcions principals de la llibreria, com poden ser la d'inicialitzar tots els submòduls importants (pygame.init) i la de desactivar-les (pygame.quit). També conté les variables globals de la llibreria, ubicades a pygame.locals.

b) Display

És el mòdul que ens permet controlar com es mostra la finestra del joc dins la pantalla de l'ordinador. Pygame permet visualitzar les aplicacions en una finestra o en pantalla completa.

Aquest mòdul disposa de diferents operacions per tal de poder tractar la visualització de la superfície que ocupa l'aplicació, com poden ser la d'inicialitzar la superfície, d'actualitzar el contingut d'aquesta, entre d'altres.

c) Event

Aquest mòdul és el que tracta els diferents events que tindrà l'aplicació o joc creat, tant els de teclat, com els de mouse, joystick o els botons de sortir de l'aplicació, minimitzar i maximitzar.

En Pygame, tots el que succeeix són missatges d'events que són encuats, i les operacions de què disposa aquest mòdul s'encarreguen de gestionar aquesta cua.

Hi ha diverses formes d'accedir a aquesta cua, on tots els events tenen un identificador.

Aquest és un mòdul que no s'utilitza massa, ja que per aconseguir l'estat de diversos dispositius, en comptes de fer servir aquesta cua, es pot accedir a la seva entrada mitjançant els propis mòduls que aquests tenen, com son els de mouse, key i joystick. Aquests tracten els events per separat de cada un dels perifèrics citats.

d) Font

Aquest mòdul permet renderitzar les fonts TrueType dins d'un objecte Superfície.

e) Image

Mòdul que conté funcions per a poder carregar i desar imatges transferint superfícies com formats que puguin ser utilitzats per altres mòduls.

Pygame només permet carregar per defecte, arxius d'imatge BMP, però podem arribar a carregar quasi tots els formats d'imatge existents, amb rutines preparades per aquesta finalitat.

f) Rect

És el mòdul que permet emmagatzemar i manipular àrees de tipus rectangular.

Un objecte Rect pot ser creat per 4 valors, que són les coordenades esquerra o superior, l'amplada i l'alçada.

g) Surface

Mòdul utilitzat per representar qualsevol imatge o figura. Surface fixa una resolució i un format dels diferents píxels que la contenen.

Quan es crea una superfície, per defecte és negra, i se li han d'explicitar unes mides. Si aquestes no se li passen, la superfície es crearà com millor cregui el display.

Existeixen altres mòduls, com el Mixer, Sprite, Movie, Music, etc, però en el nostre projecte no han estat utilitzats.

C. L'ENTORN A SUGAR

Com hem dit anteriorment, Sugar és l'entorn gràfic del Sistema Operatiu de l'XO, i per al qual hem realitzat el nostre projecte. És la Sistema que executarà la nostra aplicació i on els nens disfrutaran dels jocs de JClic.

Per tal de poder adaptar el Player de JClic codificada en una màquina Mac a Sugar hem hagut de configurar primer una màquina virtual a Mac OS X i virtualitzar-lo i així poder fer diferents proves, i un cop hem tingut les màquines XO reals, configurar-les i adaptar-hi finalment el programa.

Anem a explicar doncs com hem configurat una màquina virtual a Mac OS X, com li hem instal·lat Sugar i com hem configurat el SO per tal que la nostra aplicació s'hi pugui adaptar.

1. Màquina Virtual

Per tal de poder executar el Sistema Operatiu Sugar en el nostre ordinador necessitem un programa especial per a tal finalitat. Aquest és un sistema de virtualització per software.

Un sistema de virtualització per software és un programa que simula un sistema físic, com poden ser un ordinador o qualsevol tipus de hardware amb unes característiques hardware determinades. Quan s'executa el simulador, proporciona

un ambient d'execució similar a un ordinador físic, amb CPU, BIOS, targeta gràfica, RAM, targeta de xarxa i tots els altres components d'un PC.

Existeixen molts tipus de software d'aquestes característiques, però per a les màquines Mac, els dos que despunten per sobre dels altres són VMware, Virtual PC i Parallels.

Les prestacions dels tres són molt similars, però al tenir una llicència a mà i trobar molta més documentació de com instal·lar-lo i poder-li afegir el Sugar, per a VMware, ens vem decidir naturalment per aquest.

VMWare és, com hem dit, un sistema de virtualització per software que permet executar Sistemes Operatius dins del mateix hardware de la nostra màquina de manera simultània, permetent així el major aprofitament de recursos.

Per distingir com funciona VMware a diferència d'altres sistemes de virtualització per software, el compararem amb Virtual PC.

Mentres Virtual PC emula una plataforma x86, VMware la virtualitza, de forma que la major part de les instruccions en VMware s'executen directament sobre el hardware físic, mentre en el cas de Virtual PC es tradueixen en crides al Sistema Operatiu que s'executa en el sistema físic.

Figura 26: VMware

Un cop escollit el sistema de virtualització i instal·lat a la nostra màquina (VMware Fusion 1.1), vem haver de configurar una imatge del Sistema Operatiu Sugar, en la seva versió 656, estable per a la seva virtualització en VMware.

Un cop descarregada la imatge, en format tar.gz, la vem descomprimir i col·locar a la carpeta on van les imatges dels sistemes a virtualitzar. Només això per poder arrencar per primera vegada Sugar en la màquina Mac.

Un cop fet això, executem VMware Fusion i ja podem escollir el Sistema Operatiu que es vulgui.

En la imatge de sota podem veure la pantalla on escollim el que volem executar.

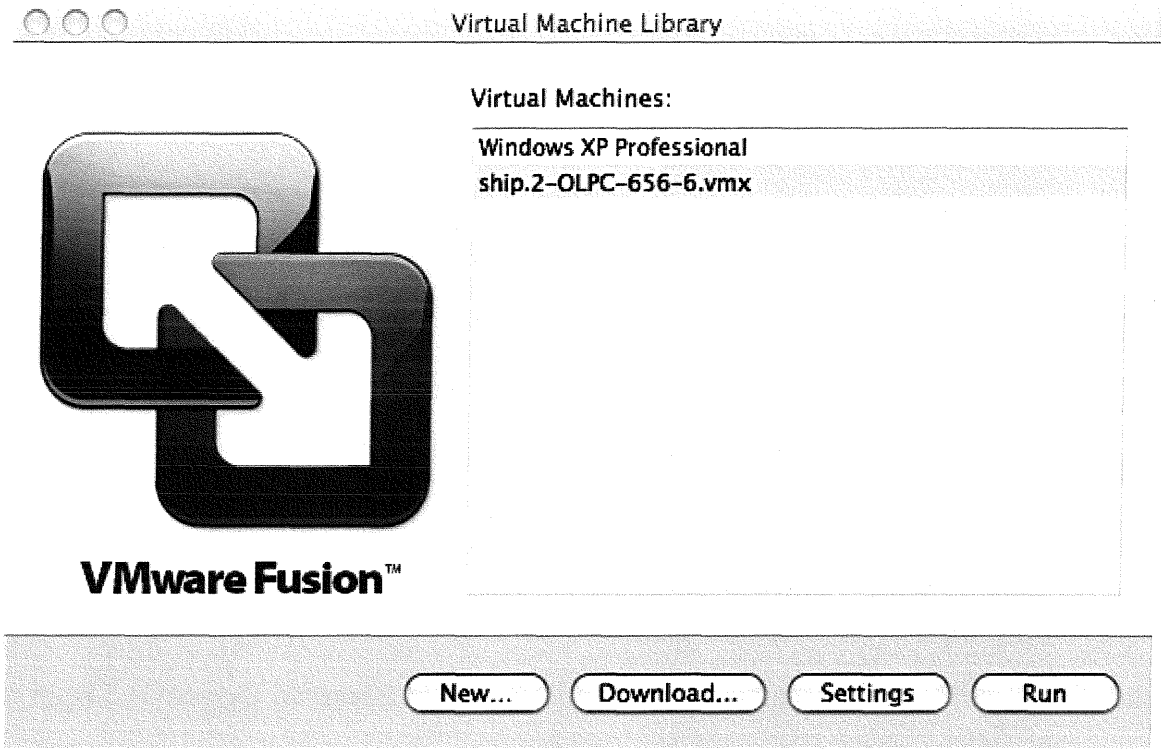


Figura 27: VMware

Un cop seleccionat el sistema que volem executar (en aquest cas Sugar), aquest es posarà a carregar, com podem veure en la imatge de sota.

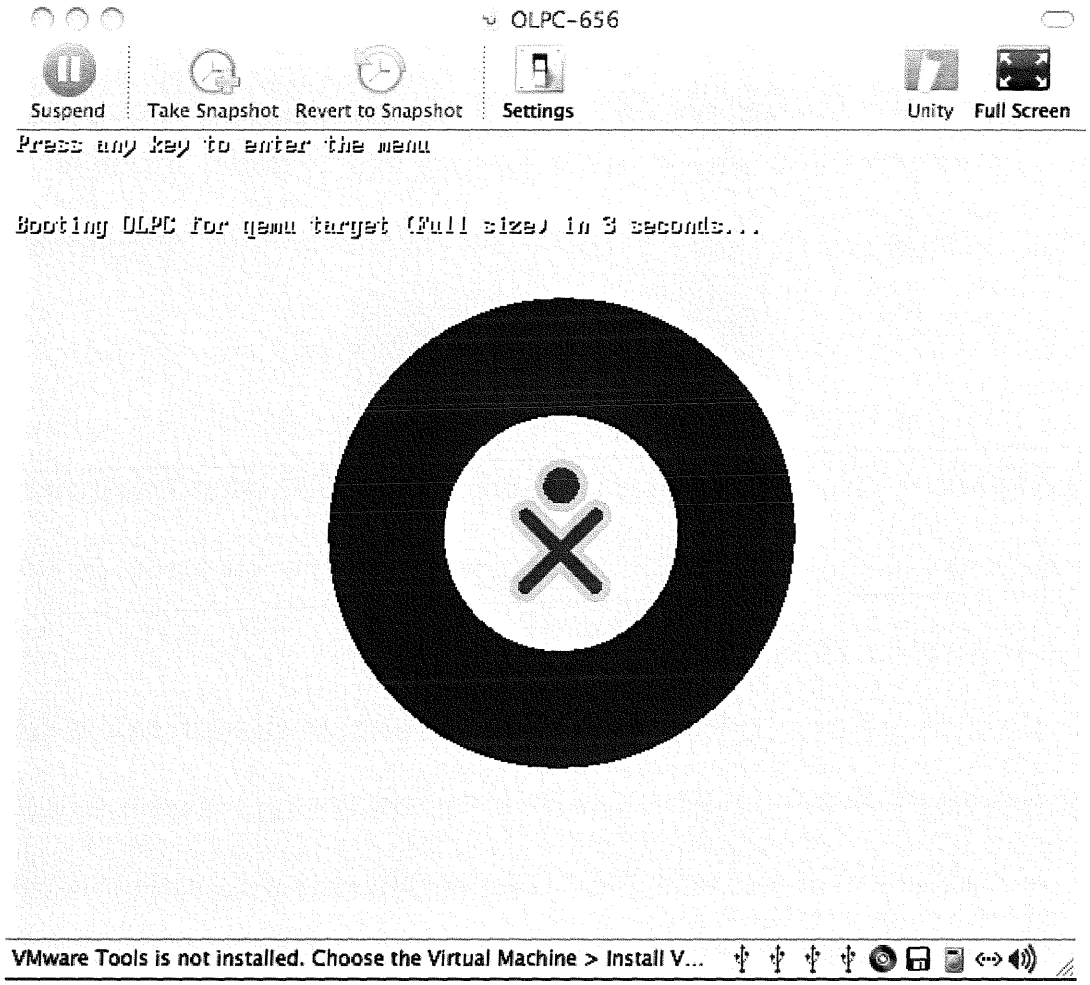


Figura 28: Càrrega Sugar

Ja tenim doncs, el Sugar funcionant en el nostre ordinador. Anem ara a veure la configuració que l'hi hem realitzat per tal que tot funcioni correctament, i la nostra aplicació pugui executar-s'hi correctament.

2. Configuració de Sugar

Un cop ja hem entrat dins Sugar, ens demanarà crear un usuari, la primera vegada que entrem. El creem i finalment ens apareix la pantalla de l'escriptori. A partir d'aquí ja podem començar a utilitzar el sistema i configurar les

coses més importants per el seu bon funcionament, com poden ser el teclat, el tipus de les fonts, començar a agafar tacte amb les comandes de la consola, ja que com hem dit anteriorment, és un Linux Fedora, per a poder instal·lar paquets necessaris, actualitzar els que ja tenim, els shortcuts més importants, entre d'altres.

Per a configurar el teclat perquè sigui del tipus QWERTY, com l'espanyol, tant en consola com per a l'entorn gràfic, a la imatge que tenim instalada (la 656) hem d'entrar a la consola i fer el següent:

- Anar a /etc/sysconfig, obrir l'arxiu **keyboard**, que en aquest cas ja existeix, i afegir KEYBOARDTYPE="pc" i canviar KEYTABLE ="es" XKB_LAYOUT="es". Amb això ja tenim el teclat configurat, però perquè els canvis tinguin efecte, hem de reiniciar el sistema.

Parlant del tema de les mides de les fonts, que en altres imatges anteriors del Sistema Operatiu es veien massa petites, en la 656 ja no el tenim, així que no hem hagut de tocar res de la configuració.

Un altre tema que vem haver de configurar va ser el tema del servidor de col.laboració, que no és més que el què permet a l'XO poder crear una comunitat virtual en forma de malla i permetre als nens jugar i aprendre junts.
Per tal que aquest servidor funcioni de forma correcta en la nostra imatge 656 s'ha de canviar el nom del servidor a olpc per olpc.collabora.co.uk. Per canviar-lo,

executem la comanda *vim /home/olpc/.sugar/default/config* i canviem el nom de la variable server que hi ha per *olpc.collabora.co.uk*.

Parlem ara una mica del tema de la consola i de diferents coses que ens hem anat trobant i que creiem que son interessants per a ser citades:

- Per a treballar com a root (superusuari) i tenir tots els permisos possibles, poder mirar els processos que s'estan executant, configurar la xarxa, etc, necessitem posar la comanda *su -*.
- Per a actualitzar els paquets que té instalats Sugar hem de teclejar la comanda *yum update*, en mode superusuari.
- Per instal·lar alguna aplicació nova al Sistema Operatiu necessitem teclejar *yum install nom_aplicació*. Aquesta comanda ens ha estat molt útil ja que hem hagut d'instal·lar varis programes. Primer de tot vam haver d'instal·lar un client d'ftp, per a poder-nos passar arxius del nostre disc dur físic, com per exemple el codi implementat en Mac, els diferents arxius .jclic, entre d'altres, al disc dur virtual de virtual de Sugar, mitjançant SSH (Secure SHell). El client que vàrem escollir va ser SFTP (Secure File Transfer Protocol), que treballa sobre SSH. Déspres d'instal·lar aquest programa i durant la implementació vàrem haver d'instal·lar també una llibreria gràfica per a Python, el wxPython, ja que Sugar no la porta per defecte i que hem hagut d'utilitzar per a realitzar la nostra aplicació.

Per acabar de comentar el tema del Sugar citarem algun shortcut important de que disposa el Sistema Operatiu Sugar.

- Per anar al mode consola: ctrl+alt+F1
- Per sortir del mode consola: ctrl+alt+F1 altra vegada
- Tancar una aplicació: ctrl+q
- Canviar d'aplicació/finestra: alt+Tab

Un cop arribats aquí, ja vem poder realitzar les proves pertinents en Sugar a la nostra aplicació, tant executant-la, com empaquetant-la en una activitat OLPC i instal·lar-la en el sistema, a part de la creació de dos activitats inicials que hem fet per a l'adaptació al Python i a la màquina XO.

V. PLANIFICACIÓ

En aquest apartat parlarem de la planificació seguida durant tot el projecte, citant les diferents etapes que aquest ha tingut, i mostrant, a partir d'un diagrama de Gant, el procés que ha seguit per arribar fins al dia d'avui i fer-se realitat.
En aquest diagrama de Gant podrem veure la repartició de feines duta a terme i la duració d'aquestes des de Febrer del 2008 fins a principis de Gener del 2009. Parteix del moment en que escollim el projecte i comencem l'aprenentage de Python fins al moment de finalitzar la memòria.

Per altra banda, també en aquest apartat calcularem el cost del projecte, a partir de les hores treballades, tant dels 2 integrants del projecte, com de la directora.

A. TASQUES DEL PROJECTE I DIAGRAMA DE GANT

A sota es pot veure la divisió en tasques i subtasques que ha tingut el projecte, des del primer dia fins al dia d'entrega de la memòria.

Instal·lació de l'entorn

Instal·lació de l'entorn de programació al ordinador

Instal·lació de l'emulador de Sugar

Aprenentatge de Python i Pygame

Realització dels primers exemples en Pygame

Exemple dels hobbies

Exemple dels conills

Estudi de com crear una activitat en Sugar

Traducció de JClic

Anàlisi de requeriments

Estudi del codi original

Implementació

Testeig

Testeig en entorn pc

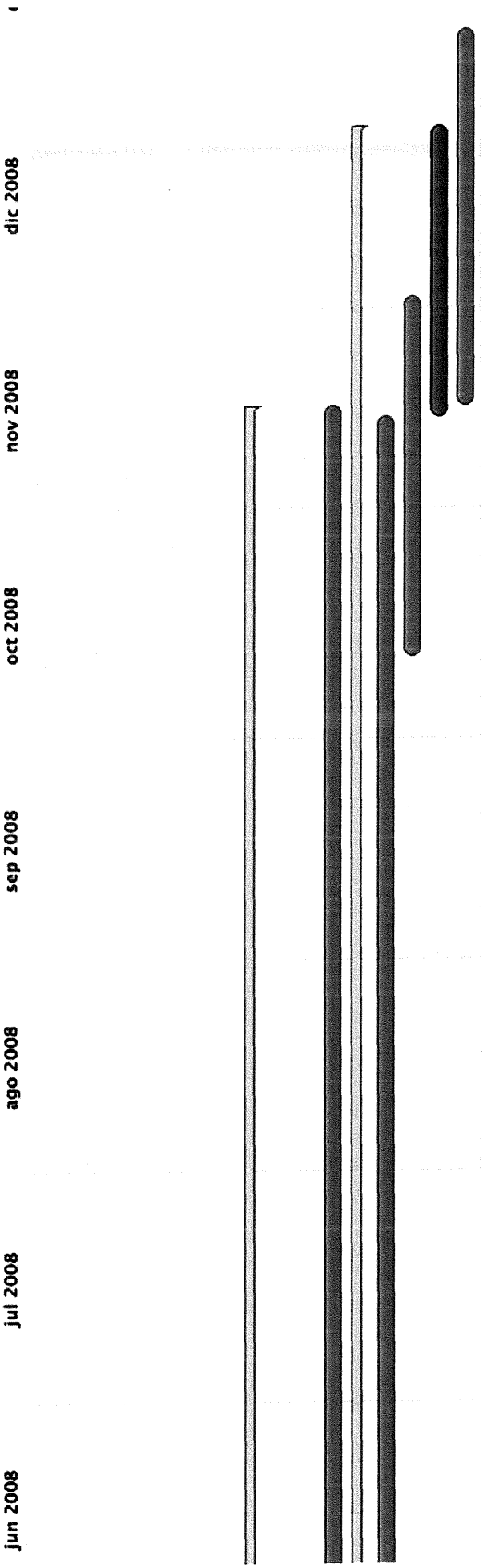
Testeig en entorn emulat

Testeig en màquina XO real

Escriptura de la memòria

A les dos pàgines següents podem veure el diagrama de Gant desglosat en les diferents tasques citades anteriorment.

En groc podem veure la feina feta per l'Aleix, en blau marí la feta per mi, Marc Rodríguez, i en vermell la feina conjunta. En blau cel podem veure la duració total d'una tasca, contant-hi totes les subtasques.


jun 2008
jul 2008
ago 2008
sep 2008
oct 2008
nov 2008
dic 2008

Tarea	Esfuerzo
1) Instal·lació de l'entorn	4s 2d
1.1) Instal·lació de l'entorn de programació a l'ordinador	2s 1d
1.2) Instal·lació de l'emulador de Sugar	2s 1d
2) Aprenentage de Python i Pygame	4s 1d
3) Realitzacio dels primers exemples en Pygame	> 4s
3.1) Exemple dels Hobbies	2s
3.2) Exemple dels conills	> 2s
4) Estudi de com crear una activitat en Sugar	3s
5) Traducció de JClic	34s 4d
5.1) Anàlisi de requeriments	2d
5.2) Estudi de codi original	3s 2d
5.3) Implementació	31s
6) Testeig	36s
6.1) Testeig en entorn pc	23s
6.2) Testeig en entorn emulat	7s 1d
6.3) Testeig en màquina XO real	5s 4d
7) Escriptura de la memòria	> 7s 2,5d

B. COSTOS DEL PROJECTE

Com hem dit anteriorment, calcularem en aquest punt el cost del projecte, tenint en compte les hores treballades i que els integrants d'aquest han sigut tres persones, dos programadors i una directora de projecte.

El que hem de fer per a calcular el valor total en preu és comptabilitzar les hores treballades, tant dels programadors com del director, que podem fer a partir del diagrama de Gant, i mitjançant el preu/hora mig de cadascun dels dos rols.

Per a fer nombres rodons, suposarem que el preu per hora que s'ha de pagar al programador és de 30 €, i per al director de projecte, 60.

Una altra suposició és que del total de les hores treballades, calcularem un 10% d'aquestes, i seran les que ha treballat el director.

Anem doncs a realitzar els càlculs pertinents per a calcular el cost total de JClic per OLPC.

Calculant que hem treballat 4 hores de mitja en tota la duració del projecte, podem comptar les hores totals treballades per a cada un de nosaltres.

La duració del projecte ha sigut de 6 mesos, exactament 48 setmanes. Hem contat que hem treballat 5 dies a la setmana.

Tenim doncs:

Marc: 50 dies feina "individual" + 451/2 dies de feina "conjunta" = 275 dies

Aleix: 71 dies feina "individual" + 451/2 dies de feina "conjunta" = 296 dies

Això seria contant els caps de setmana, per tant hem de restar d'aquestes dues xifres, els dies de cap de setmana, considerant, com hem dit, que hem treballat 5 dies a la setmana.

Marc:

275 / 7 = 39 setmanes * 2 dies de cap setmana / setmana = 78 dies

275 dies - 78 dies = 197 dies treballats

Aleix:

296 / 7 = 42 setmanes * 2 dies de cap setmana / setmana = 84 dies

296 dies - 84 dies = 212 dies treballats

Ara que ja sabem els dies treballats "reals", ja podem calcular el nombre de hores emprades:

Marc: 197 dies * 4h/dia = 788 hores

Aleix: 212 dies * 4h/dia = 848 hores

En total: 788 hores + 848 hores = 1636

Per a calcular l'aportació de la directora del projecte hem considerat un 10% de les hores dedicades pels programadors, és a dir:

10% de 1636 = 163 hores dedicades per la professora.

Tenint aquestes dades, ja podem calcular el cost total, tenint en compte el preu indicat anteriorment per als programadors i directors, 30 i 60 € respectivament:

Cost Marc = 788 hores * 30 euros/hora = 23640 euros

Cost Aleix = 848 hores * 30 euros/hora = 25440 euros

Cost Directora = 163 hores * 60 euros/hora = 9780 euros

Cost total = 23640 euros + 25440 euros + 9780 euros = 58860 euros

Cal dir que el cost total del projecte és només el preu dels treballadors, ja que no s'ha tingut cap més cost adicional que aquest, degut a que les màquines XO utilitzades per les proves han sigut prestades per a tal efecte.

VI. JOCS PREVIS

Abans de començar a implementar la nostar aplicació, i per adaptar-nos a l'entorn de desenvolupament i al llenguatge d'alt nivel escollits (Netbeans i Python juntament amb Pygame) per a desenvolupar-la, es va decidir codificar dos jocs, pertinents a JClic.

El fet d'implementar aquestes dos aplicacions ens va permetre aprendre la sintàxis bàsica del Python, juntament amb les diferents operacions de què disposa Pygame. Vàrem triar dos jocs que continguessin una mica de tot, events de teclat, events de mouse, diferents superfícies, elements multimèdia...

També ens va permetre adaptar-nos a Eclipse i a la seva manera de funcionar, com codificar-hi, com compilar, debugar (element indispensable per a trobar errors), entre moltes d'altres opcions.

Un cop codificats els dos jocs, els vem adaptar a Sugar perquè funcionessin correctament i crear l'activitat pertinent per a cada un d'ells.

Els jocs implementats són el Hobbies i El Conill. Són dos activitats encarades per a nens petits i molt senzilles.

En el primer tenim dos recuadres. En un hi ha imatges de diferents activitats o hobbies, i en l'altre hi ha el nom d'aquestes activitats. El joc tracta de relacionar el nom amb la seva imatge clicant primer en una imatge o nom i seguidament es crea una fletxa que s'ha de relacionar amb el seu pertinent. Si encertem la relació imatge-nom, aquests es desactiven. Un cop s'han relacionat totes les imatges amb el seu nom pertinent, el joc s'ha acabat i ens informa de que ja hem finalitzat.

A sota podeu veure alguna captura de pantalla de Hobbies.

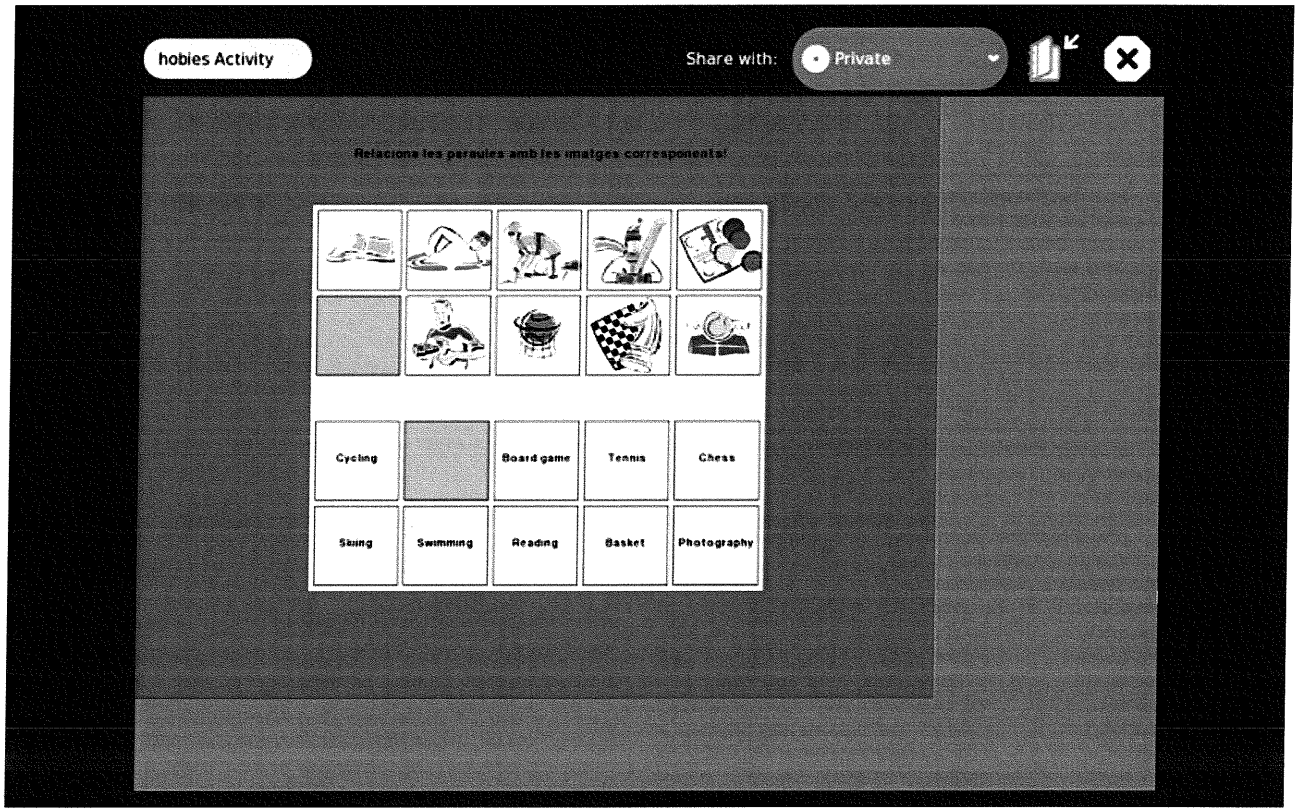


Figura 29: Hobbies executant-se en Sugar

En aquesta primera imatge podem veure el cos principal de la primera aplicació, on podem diferenciar el recuadre amb les imatges i el dels noms. Podem veure també que una de les parelles ja està desactivada, en gris, això vol dir que ja hem trobat una de les relacions.

Per a poder relacionar una imatge amb un dels noms s'ha de seleccionar o una imatge o un nom i seguidament fer seguir la fletxa que ens apareix amb l'altre recuadre. Si amb la fletxa seleccionem algun element del mateix recuadre de l'element que hem seleccionat primer, la fletxa desapareix i haurem realitzar la selecció de nou.

En la foto de sota, podem veure el moment en que es finalitza l'activitat, on veiem tots els elements desactivats.

Com podem veure en les dos imatges, l'aplicació s'executa com una activitat de Sugar, instalada en el Sistema Operatiu.

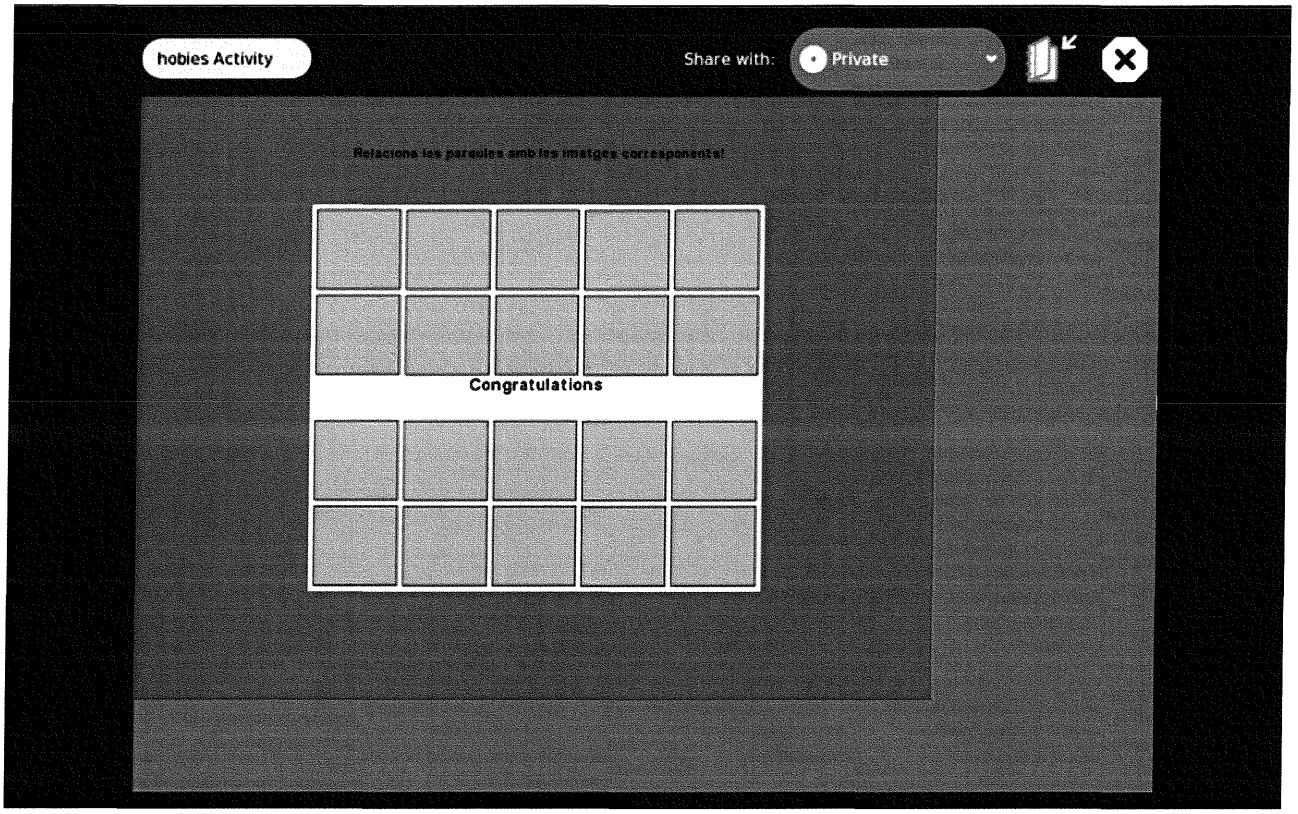


Figura 30: Hobbies acabat

El segon joc, Els Conills, és també una aplicació pensada per a nens petits i tracta de coneixer què i com és un conill.

Aquest és una mica més complex ja que disposa primer d'una pantalla inicial on es presenta el joc. Consta d'una primera activitat que correspon a un text, que és una petita definició genèrica del conill, on l'usuari ha de clicar als llocs on hi ha la paraula conill. La paraula que l'usuari clica es selecciona i un cop creu que ha

acabat, l'aplicació ens retorna el resultat. Si la selecció és correcta, el joc s'acaba i podem passar a la segona activitat. En canvi, si la selecció no ha sigut correcta (falten paraules per seleccionar o hi ha algun error i s'ha seleccionat una paraula que no és conill), no ens deixa tirar endavant i ens mostra que hi ha algun error.

Un cop realitzada l'activitat correctament, la segona activitat es tracta d'una sopa de lletres, on s'han de trobar les diferents parts d'un conill. Un cop trobada una paraula dins la sopa, ens apareix al costat una imatge de la part del conill que hem trobat.

A sota podem veure alguna captura de pantalla del segon joc.

Figura 31: El Conill executant-se en Sugar

En aquesta primera imatge veiem la pantalla de presentació del joc El Conill, amb un botó per a començar a jugar. Com podem veure, l'aplicació s'està executant com una activitat de Sugar, igual que Hobbies.

En la imatge de sota podem veure la sopa de lletres que té l'aplicació. Com hem dit i podem veure, un cop trobada una de les paraules que conté, ens apareix la imatge de la part del cos del conill que hem trobat.

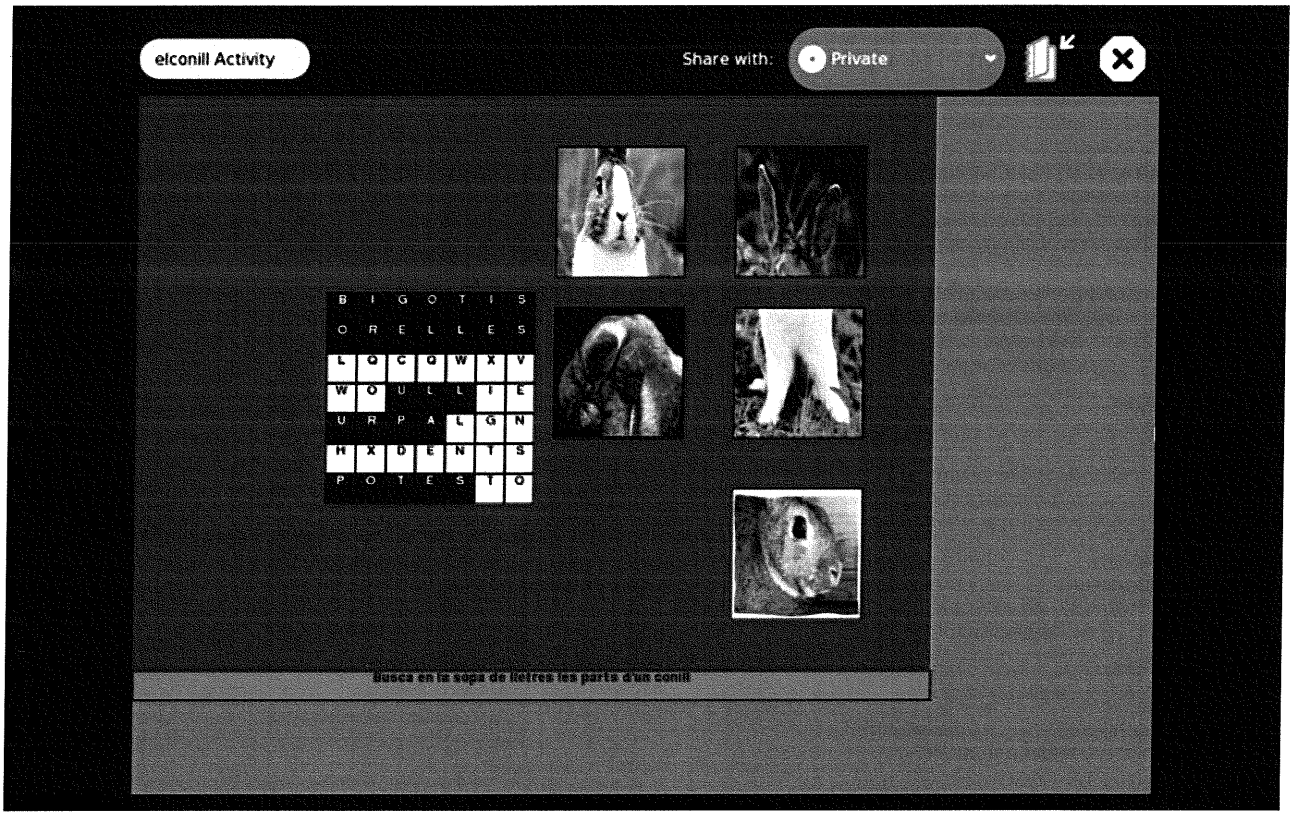


Figura 32: Sopa de lletres d'El Conill

A l'implementar aquestes dos aplicacions vàrem poder aprendre el funcionament de Python, però sobretot, les operacions bàsiques de Pygame necessàries per la traducció del JClic, el seu patró de funcionament, com tractar els diferents events,

les diferents superfícies, polígons, elements multimèdia i sobretot les diferències entre treballar en Python i la llibreria Pygame i Java i les seves llibreries gràfiques.

En aquest punt vàrem poder veure també que ens trobariem diferents problemes en la traducció, ja que amb el que amb Java i les seves llibreries gràfiques fem amb poques línees, amb Python i Pygame s'ha de fer amb moltes més, ja que treballa a més baix nivell. Això és degut a que Java és un llenguatge molt més utilitzat que Python i amb molts més recursos i llibreries per a realitzar aplicacions gràfiques.

Per acabar amb aquest tema, només ensenyar una captura de pantalla de com una de les aplicacions implementades està instalada com una activitat de Sugar.
La manera com crear una activitat i instal·lar-la al sistema, s'explicarà en un apartat posterior.

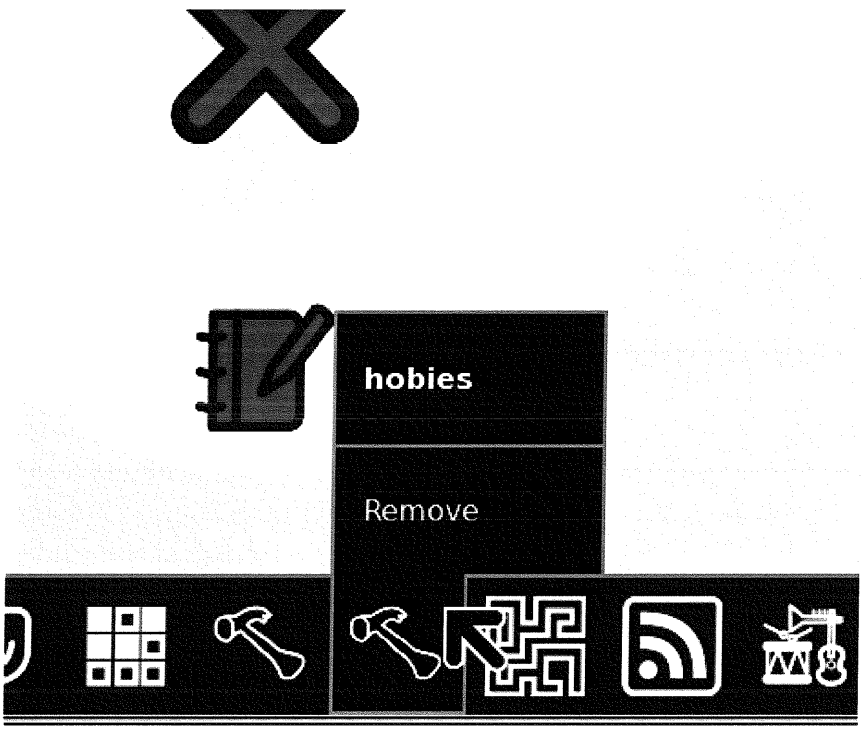


Figura 33: Hobbies instal·lat a Sugar, amb la seva icona

VII.ANÀLISIS DE REQUERIMENTS DE JCLIC

L'ànalisis de requeriments ens servirà per tenir una idea general de les funcionalitats que tindrà el nostre sistema software, i així poder tenir constància de com aquest es comporta respecte a les diferents situacions a les que es pot trobar.

Aquest és el pas previ a la identificació dels diferents casos d'ús del sistema, el que facilita aquesta tasca de forma considerable.

L'ànalisis de requeriments es divideix en els requeriments funcionals i no funcionals del sistema. Anem a veure doncs cada un d'ells.

A. FUNCIONALS

Els requeriments funcionals d'un sistema software defineixen el seu comportament intern, és a dir, les seves funcionalitats específiques.

En el nostre cas, on la funcionalitat bàsica del sistema software és simplement executar el programa i jugar amb les activitats JClic carregades, només tenim un requeriment funcional. Aquest és, com hem dit, el de poder obrir l'aplicació i jugar amb les activitats que la mateixa aplicació carrega, sigui per defecte, o descarregada d'internet.

El motiu que el nostre sistema només tingui un requeriment funcional és el fet que un cop l'executem, ja podem començar a jugar amb l'activitat que se'ns obre al

principi. A partir de que l'aplicació comença a executar-se, l'única cosa que podem fer és jugar amb les activitats carregades, tantes vegades com volguem, però és la única cosa que podem fer, i que ens interessa que faci.

El fet que només tinguem només un requeriment funcional ens permet aconseguir un requeriment no funcional que creiem que ha de tenir el nostre sistema, que no és més que la facilitat d'ús, un dels més importants, en el nostre cas.

B. NO FUNCIONALS

Un requeriment no funcional especifica propietats que ha de tenir el sistema però que no queda detallat en els requeriments funcionals, o sigui que permeten jutjar la operació d'un sistema en lloc dels seu comportament específic.

Ara farem una enumeració dels principals requeriments no funcionals que el nostre sistema ha d'incloure:

- **Portabilitat**: el nostre projecte ha de ser capaç i ho és d'executar-se en diferents plataformes, el codi font del software ha de ser capaç de poder-se reutilitzar en comptes de crear-se'n un de nou, quan el sistema passa d'una plataforma a una altra. A major portabilitat menor és la dependència del software respecte la plataforma. Aquest requeriment no funcional és necessari en el nostre cas, ja que la implementació no l'hem fet en el mateix XO, el que ha sigut imprescindible fer un codi que funcionés de forma automàtica en la màquina OLPC i així no haver de codificar res o gairebé res al fer les proves en l'XO.

- **Escalabilitat**: és la propietat que ens interessa obtenir per tal d'estar preparats perquè el nostre sistema pugui creixer sense perdre qualitat en els serveis que hem d'oferir, o sigui, ser capaços i tenir el nostre sistema preparat per tal de poder millorar-lo o adaptar-lo a les circumstàncies del moment.

- **Mantenibilitat**: el codi ha d'estar estructurat d'una manera consistent i predictible, i el sistema ha de ser implementat de tal manera que un canvi, per petit que sigui en el codi, no afecti i obligui a la generació de una nova versió d'una classe, d'una operació, o sigui, que al fer un petit canvi, no afecti a l'estructura principal del sistema.

- **Cost**: la capacitat d'obtenir i implementar el sistema software amb un pressupost el més baix possible. En el nostre cas, aquest aspecte és fonamental, ja que JClic per OLPC està creat per una màquina està pensada per a arribar als llocs amb menys possibilitats, i amb software lliure, per tant, hem d'ajustar-nos al màxim per a poder abaratir-lo.

- **Facilitat d'ús**: el nostre sistema ha de ser fàcil de ser utilitzat sense un aprenentatge previ, ja que va encarat a nens que mai o gairebé mai han utilitzat un ordinador. Com hem explicat anteriorment en els requeriments funcionals, això ho aconseguim amb escreix, ja que l'únic que pot fer l'usuari, o sigui, el nen, és senzillament jugar.

- **Estabilitat**: es diu que un sistema software és estable quan el seu nivell de fallades disminueix per sota d'un determinat llindar, que varia depenent de la estabilitat que es requereixi. En el nostre cas, no és un dels aspectes més fonamentals del nostre sistema, però hem d'intentar que el nombre de fallades sigui el menor possible, encara que l'XO, i més amb les versions del sistema operatiu que existeixen actualment, no sigui molt estable.

VIII.ESPECIFICACIÓ

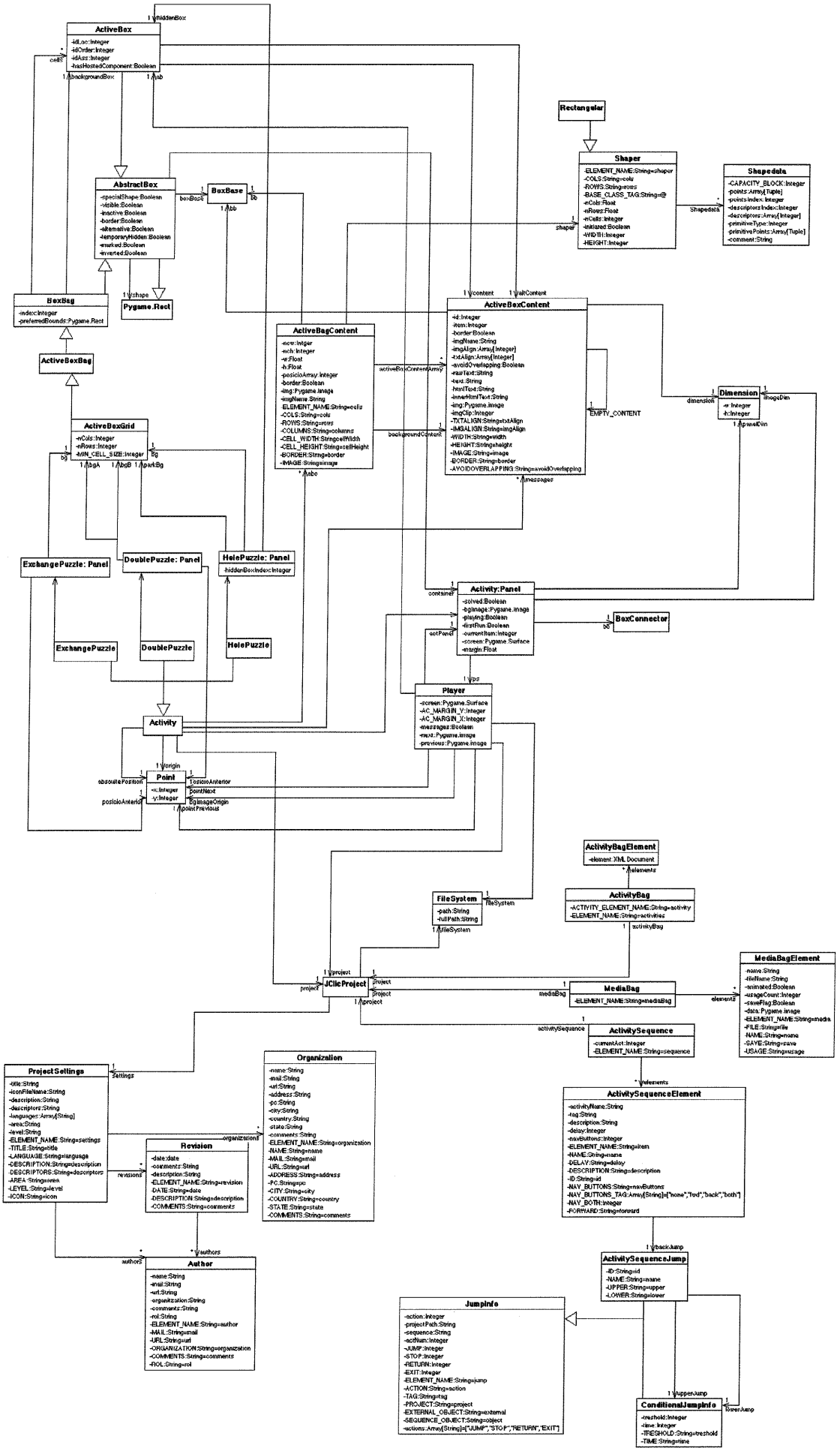
A. DIAGRAMA DE CLASSES

Un diagrama de classes és un tipus de diagràma estàtic que descriu l'estructura d'un sistema. Aquests són utilitzats durant el procès d'ànalisis i disseny d'aquests sistemes, en el nostre cas, JClic, on es crea un disseny conceptual de la informació que s'ha de manegar, i els components que s'encarregaran del funcionament i la relació entre un i altre.

Així doncs, en aquest apartat presentem el diagrama de classes del nostre projecte, amb totes les classes utilitzades i les seves relacions, i també els atributs de cada una d'elles. Cal dir que alguna de les classes no disposa dels atributs, ja que en té masses, i no cabia a la imatge.

Per fer-lo una mica més entendible, cal dir que al mig de la imatge de la pàgina 87 hi podem trobar les classes més importants i principals del projecte, com poden ser l'Activity, el Player i el JClicProject, que són les encarregades de realitzar les tasques més generals i que criden a les classes més especialtzades.

A la part de baix de l'imatge trobem les classes que engloben la tasca del parseig XML, i a la part superior, totes les classes encarregades d'implementar la interfície gràfica, com són els Panels i els Boxes.


ActiveBox
AbstractBox
BoxBase
BoxBag
ActiveBoxBag
Pygame.Rect
Rectangular
Shaper
Shapedata
ActiveBoxContent
ActiveBagContent
Dimension
ActiveBoxGrid
ExchangePuzzle: Panel
DoublePuzzle: Panel
HolePuzzle: Panel
Activity:Panel
BoxConnector
ExchangePuzzle
DoublePuzzle
HolePuzzle
Player
Activity
Point
ActivityBagElement
FileSystem
ActivityBag
JClicProject
MediaBag
MediaBagElement
ActivitySequence
ProjectSettings
Organization
Revision
ActivitySequenceElement
Author
ActivitySequenceJump
JumpInfo
ConditionalJumpInfo

B. CASOS D'ÚS

En enginyeria del software, un cas d'ús és una tècnica per a la captura de requeriments potencials del sistema software que s'està dissenyant. Cada cas d'ús proporciona un o més escenaris que indiquen com hauria d'interactuar el sistema amb l'usuari per aconseguir un objectiu específic. És, doncs, una seqüència d'interaccions que es creen entre un sistema i els seus actors en resposta a un event que inicia un actor principal sobre el propi sistema software.

En el nostre cas, el diagrama de cas d'ús es sintetitza en cinc interaccions entre l'usuari final, que no és més que el nen que utilitzarà JClic en l'XO, i el sistema software, JClic. Aquest només disposa d'un tipus d'actor, que serà el que interaccionarà amb el sistema.

No existeix cap altre tipus d'usuari, com podria passar en un altre sistema, que segons qui el manipuli té unes funcionalitats que un altre no tindria, i a l'inrevés, com per exemple en el cas d'un sistema software d'un caixer automàtic.

En aquest cas existeixen dos tipus d'actors. El client del banc o caixa que utilitza el caixer és un usuari potencial, que pot treure diners, fer un ingrès, canviar el password de la targeta, etc, però també n'existeix un altre, que seria el treballador del banc o caixa, que pot realitzar operacions que el client no pot fer, com per exemple, consultar dades de clients, saber quants diners ha extret, afegir bitllets al caixer, entre molts d'altres.

Com hem dit doncs, el nostre sistema només disposa d'un actor, l'usuari que utilitzarà JClic, i diposa de cinc operacions que pot realitzar. Aquestes són:

- Obrir el programa.
- Tancar el programa.
- Avançar a la següent activitat.
- Retrocedir a l'activitat anterior.
- Jugar a l'activitat actual a la que estem.

A sota podem veure el diagrama de casos d'ús del nostre sistema software i la especificació de cada un d'ells.

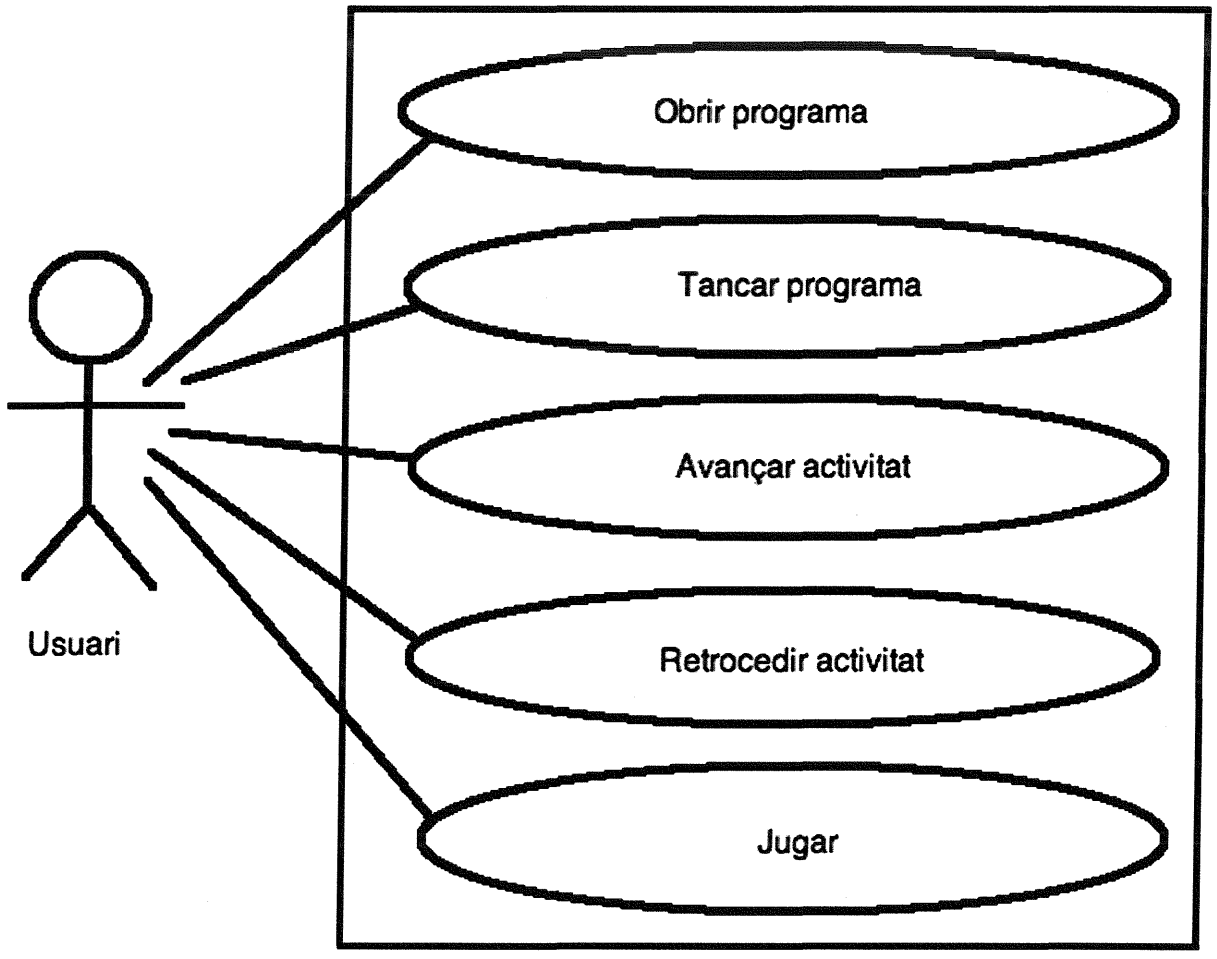


Figura 34: Cas d'ús JClic per OLPC

Cas d'ús Obrir programa:

Actors: Usuari de l'XO

Propòsit: Obrir JClic per a jugar amb alguna activitat

Resum: El programa s'executarà

Curs típic d'esdeveniments:

El cas d'ús s'inicia quan l'usuari es disposa a obrir el programa.

L'usuari clica la icona de JClic.

El sistema executa la classe principal del sistema i l'aplicació s'obre.

Cas d'ús Tancar programa:

Actors: Usuari de l'XO

Propòsit: Tancar JClic i deixar de jugar.

Resum: El programa es tancarà i sortirà, sense fer cap comprovació

Curs típic d'esdeveniments:

El cas d'ús s'inicia quan l'usuari es disposa a tancar el programa.

L'usuari clica Sortir o prem la tecla Q.

El sistema comprova que hem clicat el botó o la tecla correcta i es tanca.

Cas d'ús Avançar activitat:

Actors: Usuari de l'XO

Propòsit: Avançar a la següent activitat que conté el projecte JClic.

Resum: El programa avançarà a la següent activitat que conté el projecte obert.

Curs típic d'esdeveniments:

El cas d'ús s'inicia quan l'usuari es disposa a avançar d'activitat.

L'usuari clica el botó amb la fletxa d'avançar.

El sistema comprova que hem clicat el botó d'avançar i passa a executar la següent activitat disponible.

Cas d'ús Retrocedir activitat:

Actors: Usuari de l'XO

Propòsit: Retrocedir a l'activitat anterior que conté el projecte JClic.

Resum: El programa retrocedirà a l'activitat anterior que conté el projecte obert.

Curs típic d'esdeveniments:

El cas d'ús s'inicia quan l'usuari es disposa a retrocedir d'activitat.

L'usuari clica el botó amb la fletxa de retrocedir.

El sistema comprova que hem clicat el botó de retrocedir i passa a executar la activitat anterior disponible.

Cas d'ús Jugar:

Actors: Usuari de l'XO

Propòsit: Jugar a l'activitat JClic actual.

Resum: El programa executa l'activitat actual del projecte obert a l'iniciar l'aplicació , on l'usuari es disposa a jugar.

Curs típic d'esdeveniments:

El cas d'ús s'inicia quan l'usuari es disposa a jugar a l'activitat executant-se en aquell moment.

L'usuari realitza les operacions necessaries per a poder jugar amb l'activitat.

El sistema comprova que les accions fetes per l'usuari son correctes segons l'activitat que s'està executant en aquell moment.

IX. DISSENY

A. PATRONS DE DISSENY

Els patrons de disseny són la base per la cerca de solucions als problemes que es troben en el desenvolupament de software. Per a què una solució sigui considerada un patró ha de tenir certes característiques. Una d'elles és que ha de comprobar la seva efectivitat resolent problemes similars en ocasions anteriors. Una altra és que ha de ser reusable, el que significa que és aplicable a diferents problemes en diferents circumstàncies.

Els patrons de disseny ajuden a descompondre el sistema en diferents subsistemes més petits i s'especifiquen regles, responsabilitats i guies per a relacionar aquests subsistemes.

En el nostre cas, els patrons de disseny utilitzats per a realitzar JClic per OLPC han sigut dos (els més rellevants), el Model-Vista-Controlador i el patró Herència. Anem a veure cada un d'ells.

Model-Vista-Controlador: Primer de tot, cal dir que aquest patró ens proporciona mantenibilitat i portabilitat.

El que fa el MVC (Model-Vista-Controlador) és separar-nos les dades, la interfície d'usuari i la lògica de control d'un sistema software en tres subsistemes diferenents, que son:

- **Model**: És la representació de la informació amb què el sistema opera i que assegura la integritat de les dades. Agrupa les entitats del domini de dades.

- **Vista**: Aquest és el subsistema que presenta el model en un format adequat per a poder interactuar amb l'usuari, o sigui, la interfície d'usuari.

- **Controlador**: Subsistema que respón als diferents events, com són les accions de l'usuari, que invoca canvis en el model, que afecta quasi sempre també en la vista.

A sota podem veure el diagrama de funcionament del patró MVC.

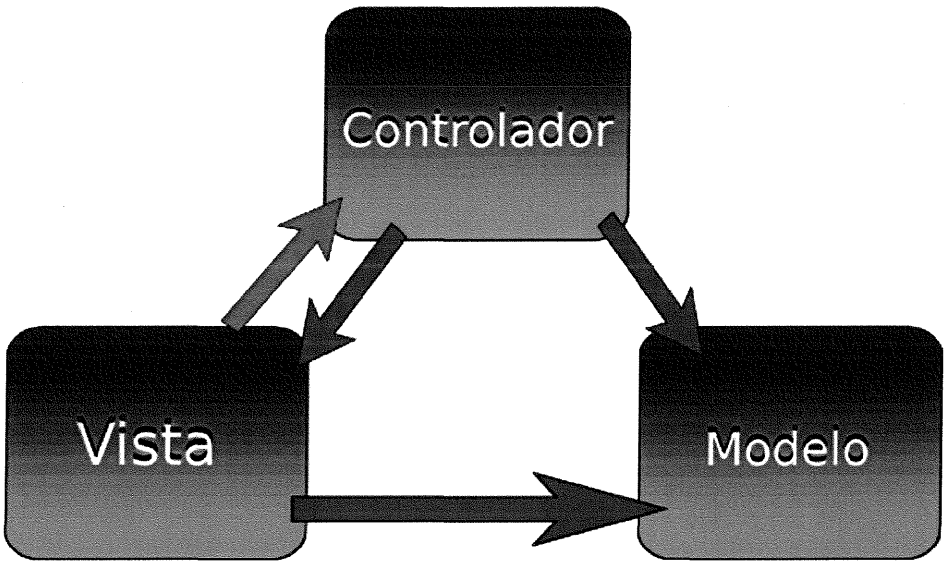


Figura 35: Patró Model-Vista-Controlador

Herència: Patró que permet que els objectes siguin creats a partir d'altres ja existents, obtenint característiques similars.

És la relació entre una classe general i una altra més específica, i ens permet crear aquestes classes més específiques, anomenades derivades o subclasses a partir

d'una classe base, anomenada també superclasse. Direm que una subclasse hereda d'una superclasse. Això ens permet compartir mètodes i atributs entre elles. A sota podem veure un exemple d'herència en el nostre sistema. Veiem com, per exemple, la classe Box Bag hereda de la classe AbstractBox, que al mateix temps hereda de la classe Rect de Pygame. Per tant, BoxBag extén les funcionalitats de AbstractBox, que al mateix temps extén les funcionalitats de la classe Rect de Pygame.

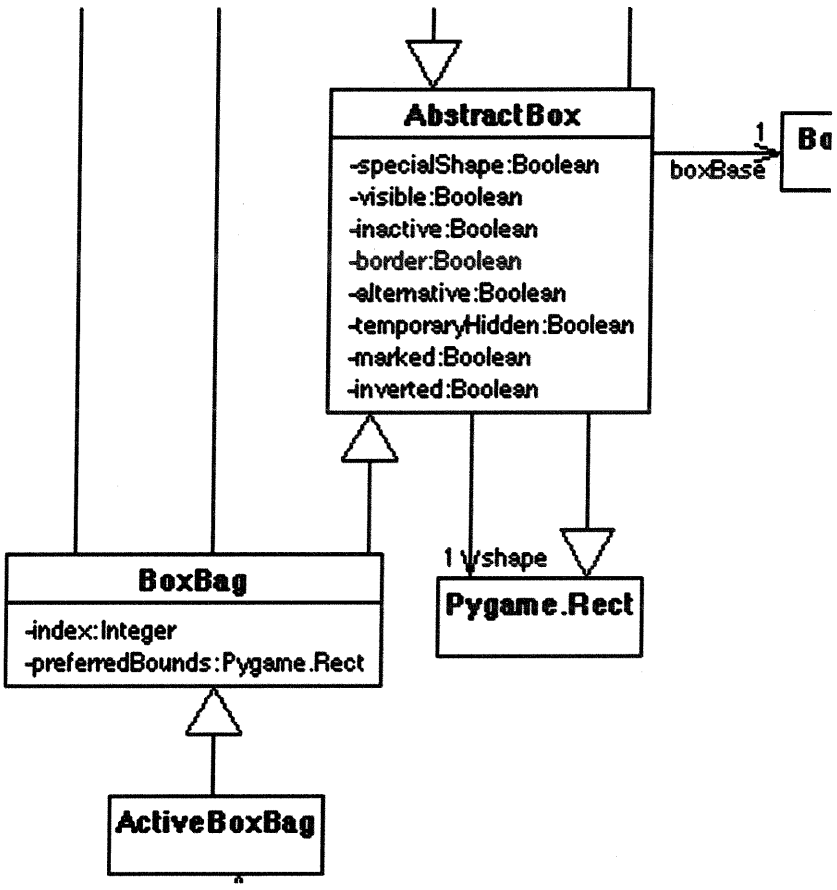


Figura 36: Exemple d'herència en JClic per OLPC

A sota tenim un altre exemple d'herència, de la classe Activity. Podem explicar aquesta herència com que existeixen molts tipus d'activitats JClic, i aquestes

comparteixen característiques, atributs i mètodes amb les altres. Totes aquestes característiques compartides estan definides en la classe Activity.

Però tenen d'altres mètodes, característiques o atributs que difereixen depenent de cada subtipus d'activitat, i que estan definits en cada subclasse de la superclasse Activity, com son ExchangePuzzle, DoublePuzzle i HolePuzzle, per exemple.

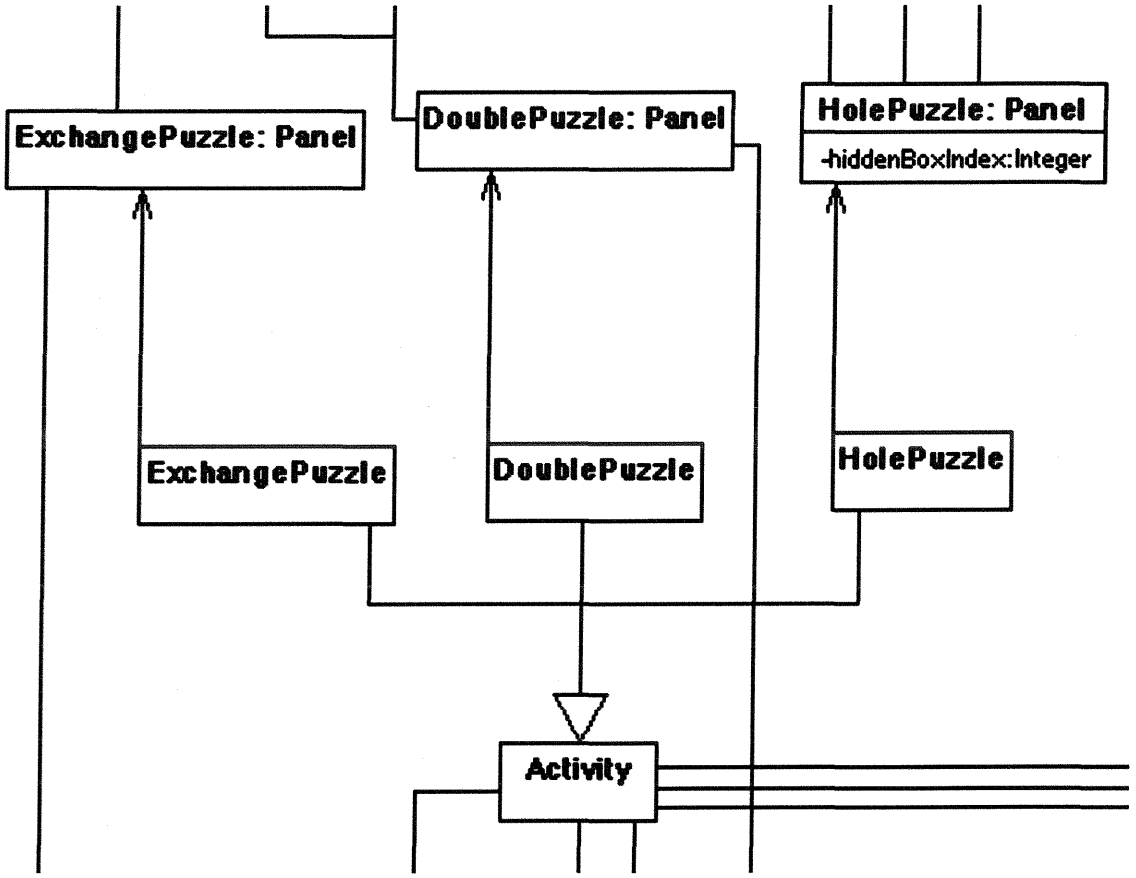


Figura 37: Herència d'Activity

X. IMPLEMENTACIÓ

En aquest apartat parlarem una mica de la implementació/traducció del JClic per OLPC i dels elements més importants dels què disposa i que cal remarcar.

El fet que féssim una traducció més o menys acurada de la versió del Player de JClic en Java ha significat que seguíssim bastant al peu de la lletra el diagrama de classes original i treballéssim amb l'estructura bàsica que té en Java. Això vol dir que algunes de les classes de menys importància han sigut traduïdes al peu de la lletra del Java al Python, lògicament amb els canvis que obliguen les diferències sintàctiques d'un llenguatge respecte de l'altre.
En canvi, les classes més importants, com la que executa el Player (classe principal), el parseig de l'XML i sobretot la interfície gràfica, ha hagut de ser modificada lleugerament de la versió Java, a causa de les limitacions de Python respecte Java, en tema de llibreries gràfiques, tractament d'arxius XML, entre d'altres.

Anem a parlar doncs, dels elements més importants de la implementació de JClic per OLPC, el parseig XML, com es guarden les dades i el funcionament bàsic de la interfície gràfica.

A. PARSEIG XML

Com hem explicat en un apartat anterior, la llibreria utilitzada per a realitzar aquesta tasca ha sigut PyXML de Python, la llibreria més potent per el tractament i parseig d'arxius XML que existeix en Python.

Aquesta llibreria ens ha servit per realitzar el parseig dels arxius .jclic (arxius XML en el fons) per tal de poder carregar i visualitzar les activitats que aquest conté.

La classe on comença el parseig dels arxius .jclic és la classe Activity, on es comencen a guardar en atributs globals on hi ha contingudes les característiques principals del projecte JClic que el sistema està carregant.

El parseig es realitza de forma recursiva, ja que els arxius .jclic contenen tags principals, que al mateix temps contenen tags que pertanyen al primer, com podem veure en la imatge de sota.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Nom del projecte JClic i la versió -->
<JClicProject name="climatic" version="0.1.3">
 <!-- Característiques del projecte JClic -->
 <settings>
  <!-- Títol del projecte -->
  <title>El canvi climàtic</title>
  <!-- Descripció i data de les revisions que ha tingut el projecte -->
  <revision description="Creació del paquet d'activitats en Clic 3.0" date="11/27/07" />
  <revision description="Conversió de Clic3 a JClic" date="11/28/07">
   <!-- Nom, mail, organització de l'autor de la revisió -->
   <author name="Equip Clic" mail="clic@xtec.net" organization="Àrea TIC - Departament d'Educació" />
  </revision>
```

Figura 38: Arxiu .jclic

En la imatge podem veure com comença un arxiu .jclic, amb el nom del projecte, i les seves característiques.

El que fa doncs, la classe Activity, amb l'ajuda de la classe JClicProject, és guardar tots aquests atributs, parsejant de dalt a baix.

Podem veure un tall del codi de parseig de la classe Activity per fer-nos una idea de la forma de fer-ho.

```
child = e.getElementsByTagName(self.DESCRIPTION)
if(child != []):
    self.description = getParagraphs(child[0])

#child = e.getChild(self.MESSAGES)
child= e.getElementsByTagName(self.MESSAGES)
if(child != []):
    #Iterator itr = child.getChildren(ActiveBoxContent.ELEMENT_NAME).iterator()
    child2 = child[0].getElementsByTagName(ActiveBoxContent.ELEMENT_NAME)
    for x in child2:
        #child2=((org.jdom.Element)itr.next())
        i = getStrIndexAttr(x.getAttribute(self.TYPE), self.MSG_TYPE, -1)
        if(i >= 0):
            self.messages[i]=getActiveBoxContent(x, self.project.mediaBag)

child = e.getElementsByTagName(self.SETTINGS)
if(child != []):
    self.margin = getIntAttr(child[0], self.MARGIN, self.margin)
    self.infoUrl = getStringAttr(child[0], self.INFO_URL, self.infoUrl, False)
```

Figura 39: Tall de codi del parseig a la classe Activity

Un cop ha guardades les característiques del projecte, passa a guardar la seqüència d'execució de les activitats que formen el projecte JClic a carregar, que ho delega a una altra classe, l'ActivitySequence, d'on en deriven d'altres.

```
<sequence>
 <item name="01 Les bèsties" />
 <item name="puz02.puz" />
 <item name="02 Definició canvi climàtic" />
 <item name="03 Les bèsties" />
 <item name="puzle dofins" />
 <item name="04 Les bèsties" />
</sequence>
```

Figura 40: Seqüència d'activitats

Ja tenim en aquest punt la seqüència d'execució guardada. Ara toca començar a guardar els atributs de cada activitat que conté el projecte, per separat, delegant la

feina a diferents classes segons el que s'ha de guardar, com poden ser l'ActivityBag, ActiveBox, ActiveBagContent i altres classe que deriven d'aquestes. Podem a sota, veure un tall de l'arxiu .jclic, on hi han les característiques de cada activitat.

```
<activities>
 <!-- Nom i tipus de l'activitat -->
 <activity class="@puzzles.ExchangePuzzle" name="02 Definició canvi climàtic" code="">
  <!-- Descrició de l'activitat -->
  <description />
  <!-- Missatges que l'activitat dona a l'usuari -->
  <messages>
   <!-- Boxes que s'utilitzen per mostrar els missatges, amb les seves característiques
   <cell type="initial">
    <style borderStroke="0.0" markerStroke="2.7">
     <color background="0xFFFFCC" />
    </style>
    <!-- Text que tindrà el missatge -->
    <p>Ordena la definició de canvi climàtic</p>
   </cell>
   <cell type="final">
    <style borderStroke="0.0" markerStroke="5.0">
     <color background="0xFFFFCC" />
    </style>
    <p>Molt bé.</p>
   </cell>
   <cell type="finalError">
    <style />
    <p>Aquest ordre no és el correcte</p>
   </cell>
  </messages>
  <!-- Característiques principals de l'activitat -->
  <settings margin="8" dragCells="true" report="true" reportActions="false">
```

Figura 41: Tall de característiques de cada activitat

Ja tenim, en aquest punt, parsejades les diferents activitats que formen part del projecte. Ja només queda guardar els diferents elements multimèdia de què disposa el projecte i que haurem de reproduir en el moment d'execució.

A sota veiem el tall de l'arxiu .jclic que conté la informació sobre els elements multimèdia.

```
<mediaBag>
 <!-- Noms del fitxers multimedia que s'utilitza -->
 <media name="baixa-cilindrada.gif" file="baixa-cilindrada.gif" />
 <media name="aigua-calenta.jpg" file="aigua-calenta.jpg" />
 <media name="bombeta.jpg" file="bombeta.jpg" />
 <media name="bosc.jpg" file="bosc.jpg" />
 <media name="bushidrogen.jpg" file="bushidrogen.jpg" />
 <media name="central_eolica.jpg" file="central_eolica.jpg" />
 <media name="central_nuclear.jpg" file="central_nuclear.jpg" />
 <media name="circuitsecundari-alternador-generador.jpg" file="circuitsecundari-alternador-generador.jpg" />
 <media name="cotxe-hibrid-toyota-prius.JPG" file="cotxe-hibrid-toyota-prius.JPG" />
 <media name="edifici_contencio.jpg" file="edifici_contencio.jpg" />
 <media name="estufa.jpg" file="estufa.jpg" />
 <media name="finestra.jpg" file="finestra.jpg" />
 <media name="nucli.jpg" file="nucli.jpg" />
```

Figura 42: Elements multimèdia del projecte JClic

El parseig dels elements multimèdia es realitza a partir de la classe MediaBag i les classe que deriven d'aquesta.

Un cop realitzat tot parseig del document .jclic, ja tenim tota la informació necessària per a poder-la presentar per pantalla i així poder implementar la interfície gràfica.

B. INTERFíCIE GRÀFICA

Un cop ja tenim parsejat l'XML, ja estem en disposició de mostrar l'acitivitat per pantalla. Per tal de poder fer això, primer cal comprendre com estan emmagatzemades les dades a mostrar.

C. EMMAGATZEMATGE DE LES DADES

En la figura de sota podem veure el que seria una imatge normal del JClic, on estan marcats els elements que varien entre activitats, és a dir, els elements que no són comuns a cada una d'elles i que per tant estan guardats en una classe del subtipus corresponent d'activity (els elements gràfics comuns entre acitivitats estan guardats en la classe Player, que representa la finestra principal de l'aplicació).

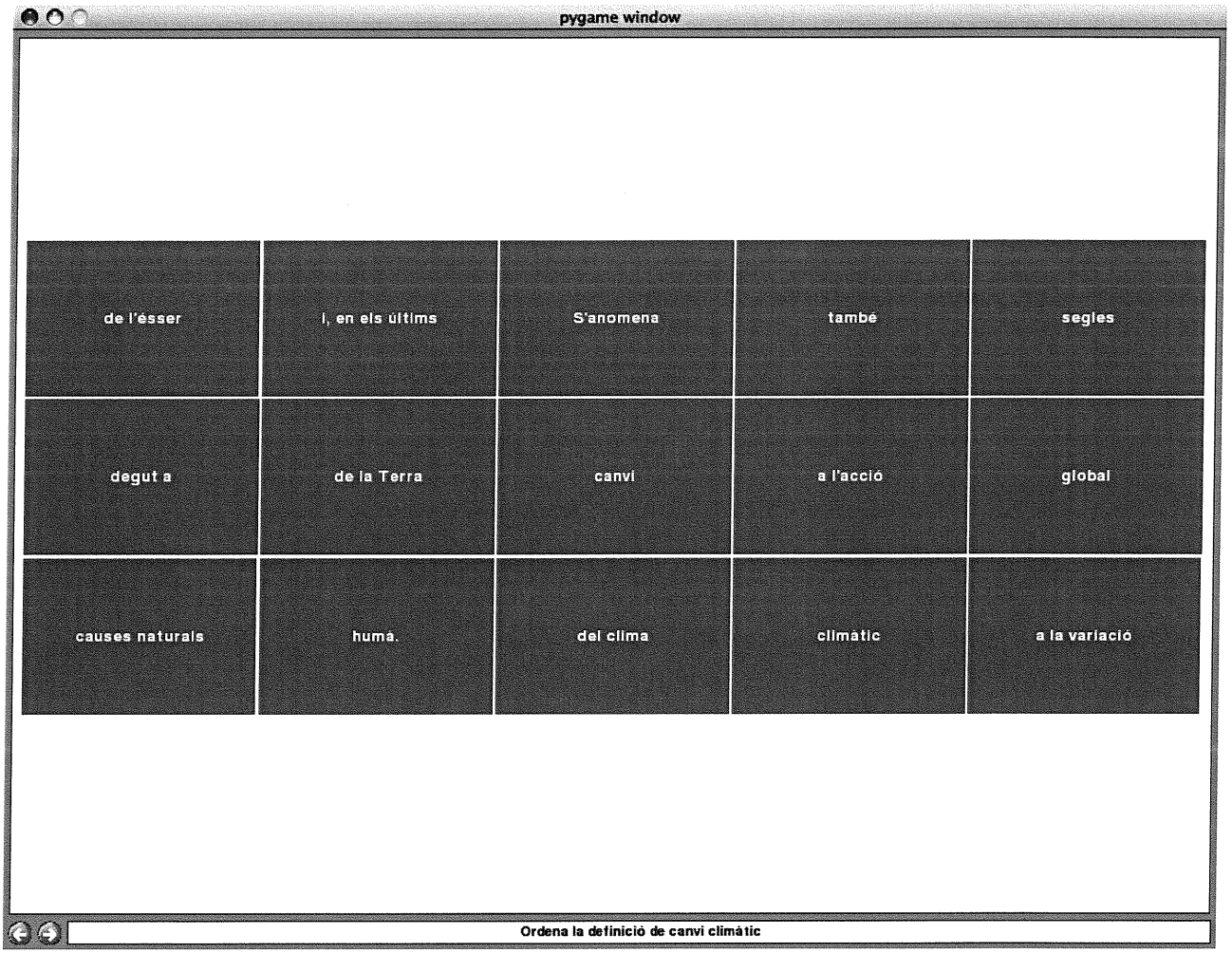


Figura 43: Elements gràfics

Com podem veure, hi han dos elements principals on es mostra informació que canvien dinàmicament a mida que l'activitat avança:

- Quadre de missatges: es mostren missatges d'informació sobre el què s'ha de fer a l'activitat, així com missatges on s'ens indica si estem fent bé o no l'exercici.
- Quadre de l'activitat o Activity.Panel: és on pròpiament es desenvolupa l'acció de l'activitat, on es mostren els quadres que la formen i que van canviant a mida que es capten els events que genera l'usuari.

Les dades que formen aquests dos quadres es guarden en dos atributs de l'objecte activity (o en la subclasse corresponent):

- Messages: objecte declarat en la classe activity, és a dir, és comuna a totes les subclasses d'activitat que poguem utilitzar. Es tracta d'un objecte del tipus array, format per objectes ActiveBoxContent. Aquest objectes són els que contenen la informació que es mostra en els ActiveBox, que com veurem més endavant, són els que formen la interfície gràfica.

El quadre d'activitat, tal i com podem veure en el diagrama de classes, al dependre de quina sigui l'aparença gràfica de l'activitat que estem tractant, està declarat en els diferents tipus de subclasse d'activitat. Pot ser un sol objecte o més d'un, ho veurem més extensament en la implementació gràfica.

Un cop tenim clar on està emmagatzemada la informació a mostrar, ja podem veure com es mostren aquestes dades per pantalla.

D. ESTRUCTURA DEL QUADRE DE L'ACTIVITAT

Primerament és interessant veure quins són els dos tipus d'objectes que poden formar un quadre d'activitat:

- **ActiveBox**: són la unitat mínima que pot formar un quadre de l'activitat. Cada un dels objectes ActiveBox té un objecte ActiveBoxContent associat, que és el contingut que s'ha de mostrar dins del box.
- **ActiveBoxGrid**: com el seu nom indica, es tracta d'una graella d'ActiveBox que formen el què es l'activitat en si. Com veurem més endavant, una activitat pot estar formada per una o més graelles.

Com acabem d'apuntar anteriorment, l'estructura del quadre depèn del tipus d'activitat que volem mostrar. Per tant, separarem aquest capítol en tres apartats, que corresponen als tres subtipus d'activitat que hem implementat.

1. Estructura de l'Exchange Puzzle

L'objectiu de l'exchange puzzle és ordenar una imatge o frase desordenada, intercanviant les caselles de dos en dos. És a dir, seleccionar

primerament la casella que no es troba en el lloc apropiat i a continuació clickar on creiem que es troba la seva posició correcta.

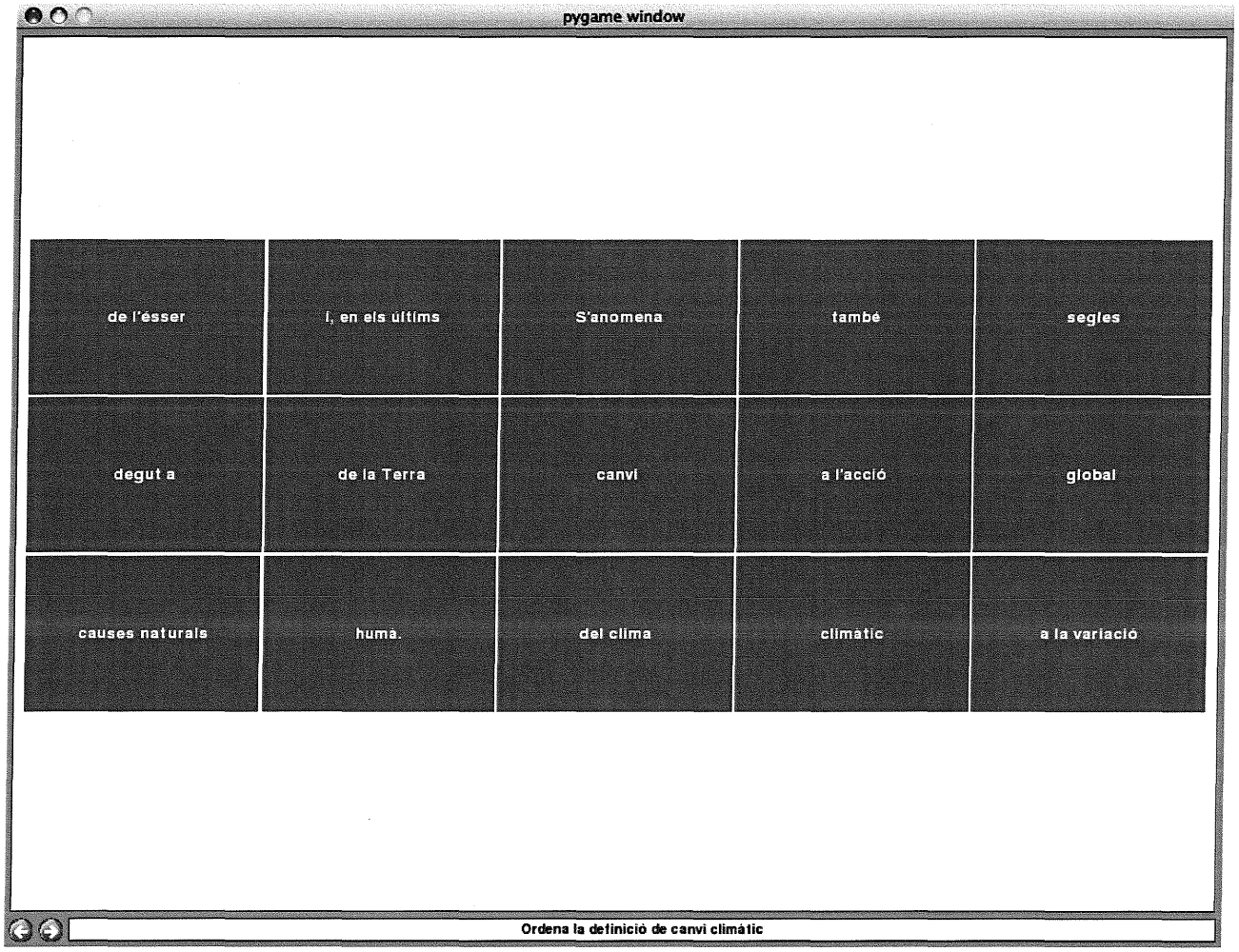


Figura 44: Elements gràfics

En la imatge superior veiem un exemple del que seria un exchange puzzle amb la frase "S'anomena canvi climàtic a la variació global del clima de la Terra degut a causes naturals i, en els últims segles també a l'acció de l'ésser humà." desordenada.

Per contra, en la imatge inferior podem veure un exchange puzzle en el que hem d'ordenar és una imatge d'un astronauta.

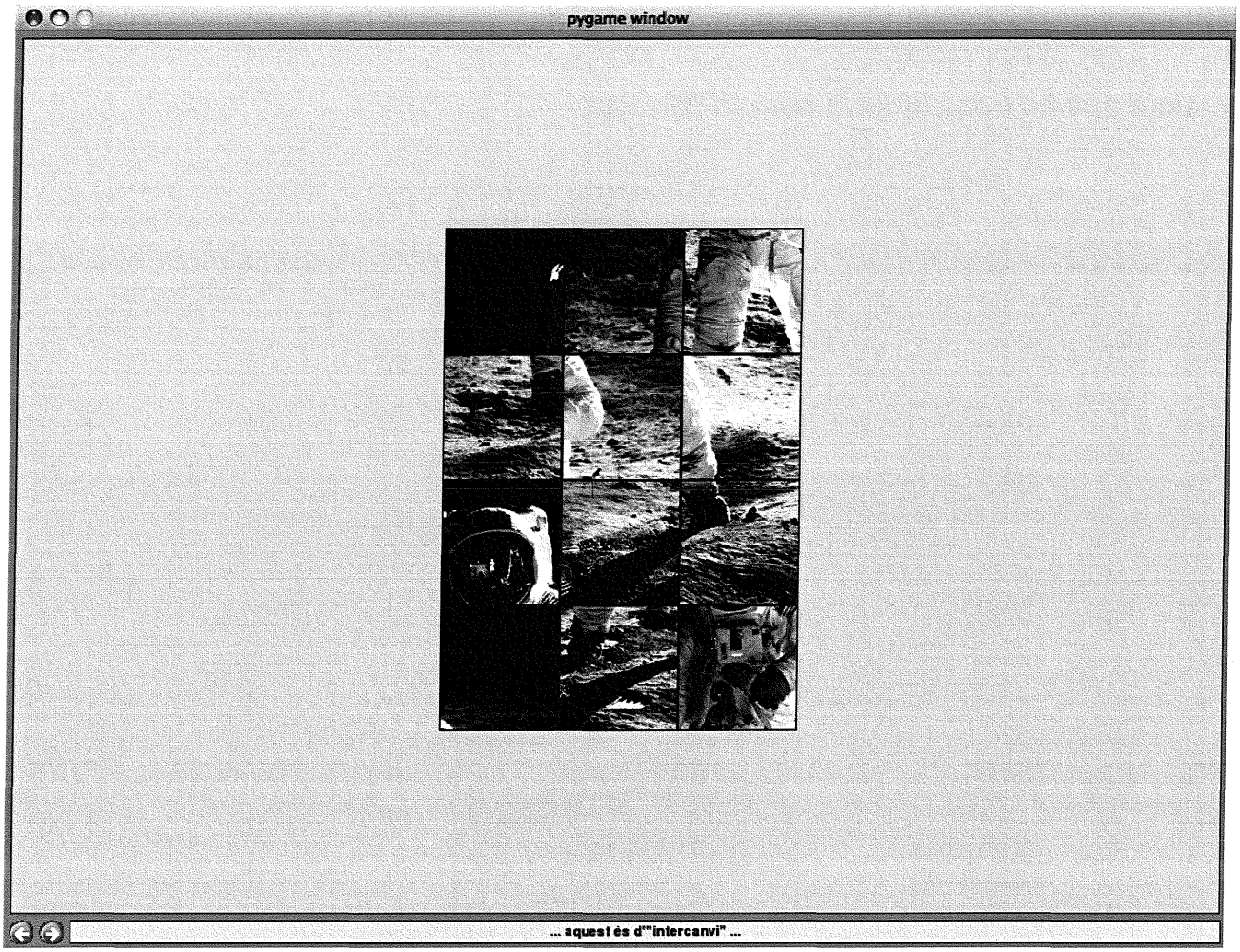


Figura 45: Exchange Puzzle

Com es pot veure en les dues imatges, l'exchange puzzle està format sempre per un sol ActiveBoxGrid sobre el cual s'ordena la imatge o frase desordenada. Al fer el segon click el que fem és intercanviar els ActiveBox i si el canvi que hem fet és correcte, ho contabilitza. Quan ja tenim totes les caselles en la posició correcta, donem el puzzle per acabat.

2. Double Puzzle

El double puzzle és similar a l'exchange puzzle, amb la diferència que la imatge o frase desordenada no s'ordena sobre ella mateixa, sinó que hem de moure la casella mal col·locada a una graella buida, on la col·locarem en la posició que creiem que és la correcta.

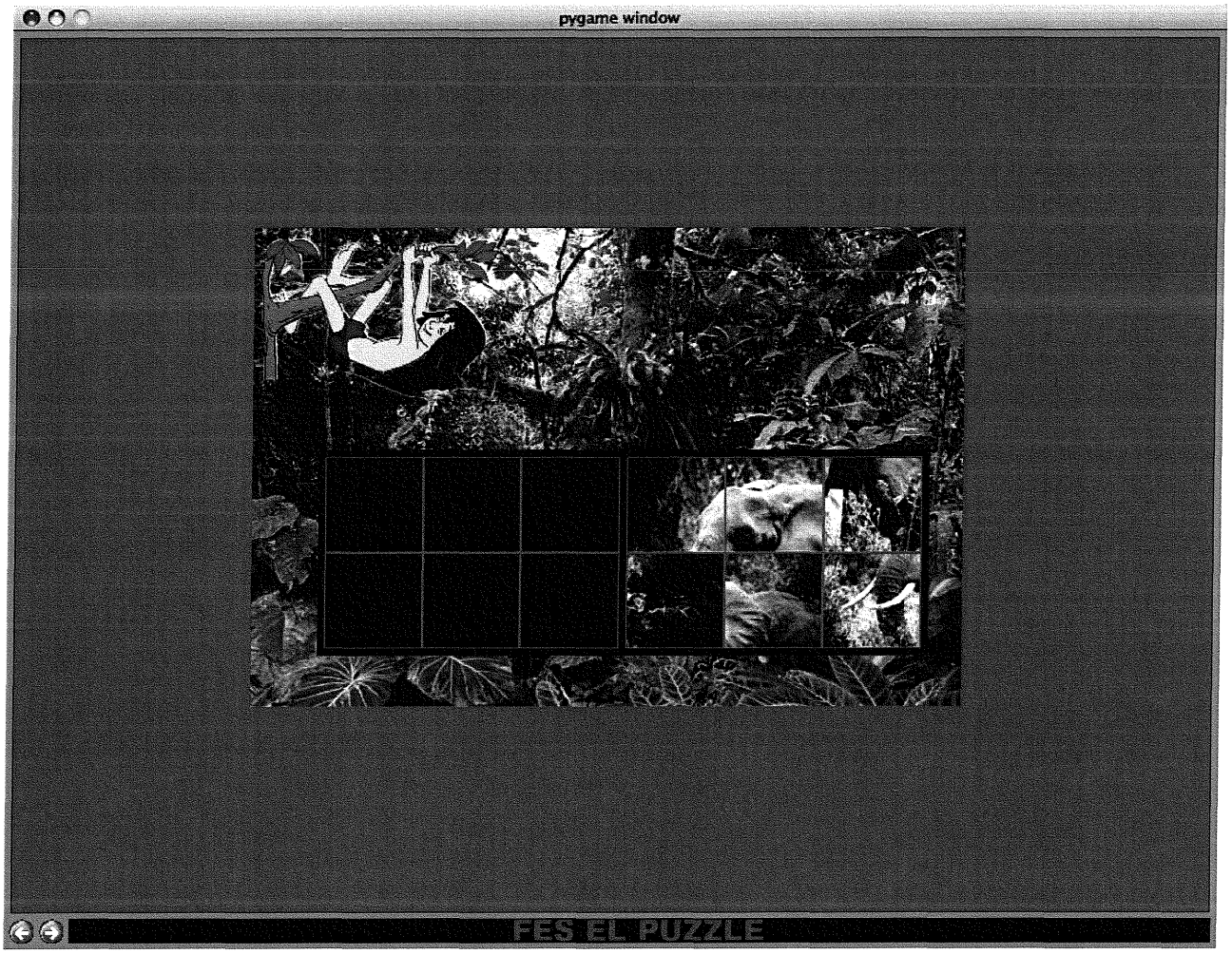


Figura 46: Double Puzzle

Per tal de implementar aquest tipus d'activitats, utilitzem dos ActiveboxGrids, el de la dreta està desordenat, el de l'esquerra està ordenat però ocult. Quan col·loquem

una de les caselles de la graella de la dreta en la posició correcta (en la graella de l'esquerra), el que fem és ocultar la primera casella i mostrar la segona.

3. Hole Puzzle

Ja per últim tenim el puzzle de forats, probablement el més complicat dels tres tipus. En aquest tipus de puzzle la imatge està desordenada i li falta una peça. La peça que falta la podem veure al costat del puzzle. L'objectiu és ordenar el puzzle però només podem moure les caselles que estan tocant el forat. L'activitat estarà finalitzada quan el puzzle estigui ordenat i el forat es trobi just on hauria d'anar la casella que tenim separada.

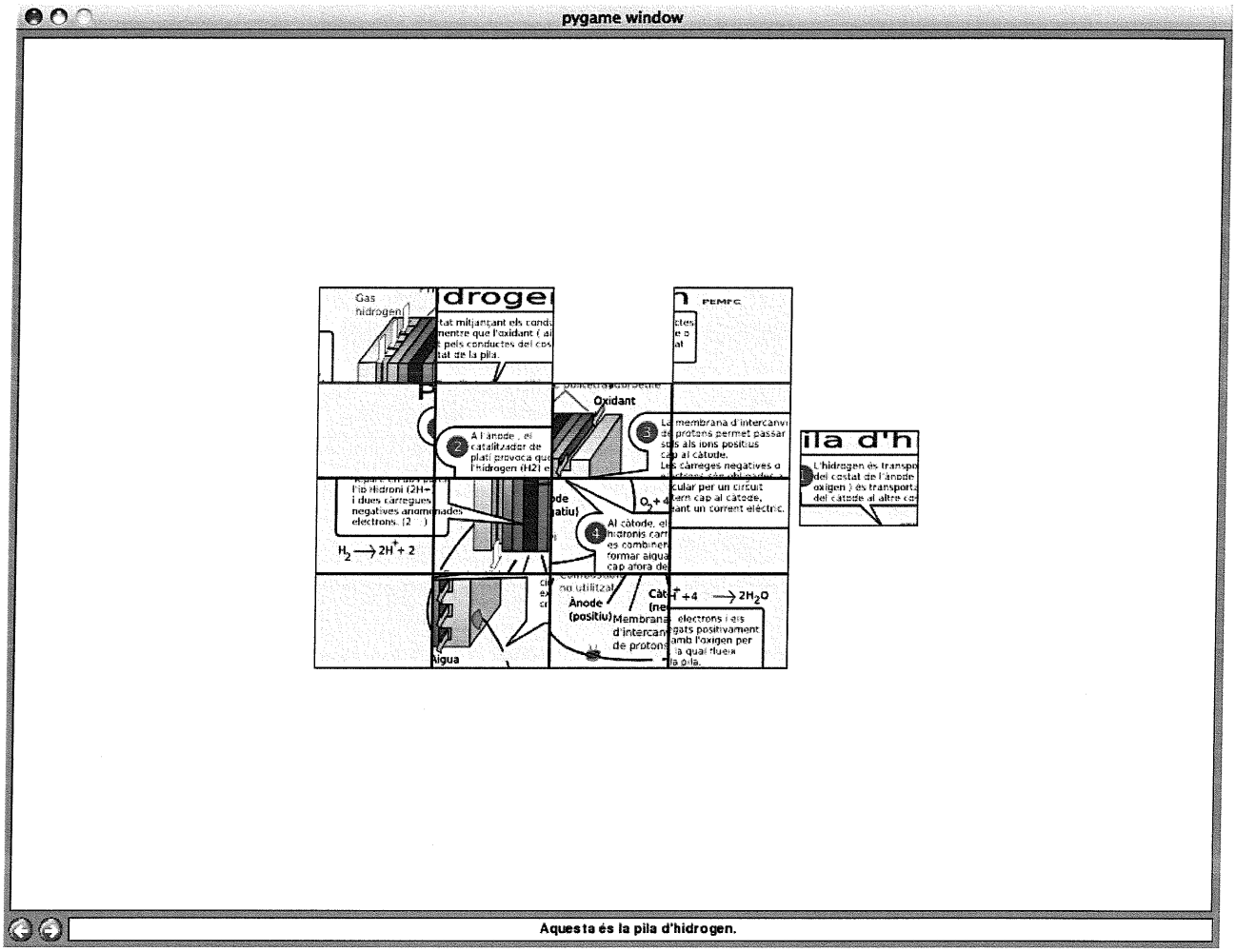


Figura 47: Hole Puzzle

Per tal d'implementar aquest tipus de puzzle utilitzem tres objectes: dos ActiveBoxGrid i un ActiveBox. L'ActiveBox correspon a la casella que es troba oculta en cada moment, el primer dels ActiveBoxGrid al que està desordenat i el segon ens serveix per mostrar la casella que es troba a part, així com per consultar si ja tenim el puzzle resolt.

E. IMPLEMENTACIÓ EN JAVA VS. IMPLEMENTACIÓ EN PYTHON

Un cop vist quines són les estructures de dades que s'amaguen sota l'aparença gràfica de l'aplicació, en aquest capítol veurem com es pinten aquestes dades a la pantalla, doncs hem introduït un canvi substancial en la implementació degut a la "filosofia" de les llibreries gràfiques utilitzades. En el cas de Java, s'utiltiza la llibreria Swing, i cada un dels ActiveBox es tracta d'objectes Rectangle. Alhora de dibuixar l'escena, els quadres només es dibuixen un sol cop, i quan s'han de moure s'utilitzen funcions del tipus resize, però no es tornen a pintar cada cop que la pantalla es repinta.

Això no ho permet el Pygame, ja que la "filosofia" que té darrera és la de pintar completament l'escena cada cop. Per tant hem hagut d'introduir una implementació completament diferent, a través d'un bucle que pinta l'escena en cada iteració.

```
while going:
    self.paintBackground()
    if self.messages:
            self.ab.update(self.screen, self)
    self.paintButtons()
    self.actPanel.repaint()
```

Com es pot veure en el fragment de codi superior extret del fitxer Player, el bucle comença pintant la part comuna de totes les activitats, pinta el fons, els botons i els missatges. Llavors delega a Activity pintar el que és pròpiament l'activitat. Aquest pintarà coses superficials, com ara el color de fons del quadre de l'activitat, però no

pot pintar res més ja que tal i com hem explicat la informació del quadre està emmagatzemada en la subclasse corresponent, així que es torna a delegar. I és llavors la subclasse, que conté la informació dels ActiveBoxGrids, etc... l'encarregada de mostrar-los per pantalla.

Pel que fa al recull i processat d'events, també segueix una implementació similar:

```
events = pygame.event.get()
    for event in events:
        if event.type == QUIT:
            exit()
        p = pygame.mouse.get_pos()
        if event.type == MOUSEBUTTONDOWN:
            if self.isOver(p, self.pointNext, self.next):
                self.nextActivity()
            elif self.isOver(p, self.pointPrevious,
self.previous):
                self.previousActivity()
        self.actPanel.processEvent(events)
```

En Pygame, els events es recullen a través de la funció pygame.event.get(). Aquesta funció ens retorna un vector d'events els quals s'han de consultar un a un per saber de quin event es tracta. El que hem fet, de la mateixa manera que amb l'aspecte gràfic, és consultar en quin lloc de la pantalla s'està produïnt l'event i llavors el processa la classe corresponent. És a dir, si estem apretant el botó de següent activitat, al estar dins el context del Player, es processa en aquest mòdul (de fet es pot veure en el codi superior, extret d'aquesta mateixa classe, si clickem sobre un dels seus dos botons, l'acció té lloc dins del Player). En cas de què

l'event no tingui lloc aquí, es delega cap a la classe Activity i posteriorment cap a la subclasse corresponent.

XI. COM CREAR UNA ACTIVITAT A SUGAR

En aquest apartat parlarem de com crear una activitat en Sugar a partir d'una aplicació creada en una altra plataforma, com podrien ser un dels jocs implementats o el nostre projecte, els elements imprescindibles per a poder-ho fer i els passos que s'han de seguir perquè això sigui possible.

El primer i essencial és disposar d'una aplicació, sigui petita o gran, per tal de poder-la convertir en una activitat preparada per a Sugar. Per tant, amb una aplicació "Hello World" en Python serviria per a poder-ho provar, però ja que vem implementar dos jocs (El Conill i Hobbies) de prova, els podem utilitzar per a crear l'activitat i així de pas provem que el Python amb Pygame funciona correctament a Sugar.

Així doncs, el primer que hem de fer és passar el nostre codi, per exemple, el del joc Hobbies a la màquina XO.

Com que les proves les vem fer en el Sugar virtualitzat amb VMware en MAC, explicarem tot el procés en aquest cas concret.

Per a fer-ho, primer hem d'instal·lar a la màquina XO en mode consola un client d'ftp, en el nostre cas sftp (ftp funcionant en ssh, instal·lat amb la comanda *yum install sftp* en mode superusuari), com hem explicat en un apartat anterior i executar en Mac el servidor ssh per a establir la comunicació entre el SO físic i el

virtualitzat (podem veure-ho a la foto de sota) i poder transferir el codi des d'un a l'altre.

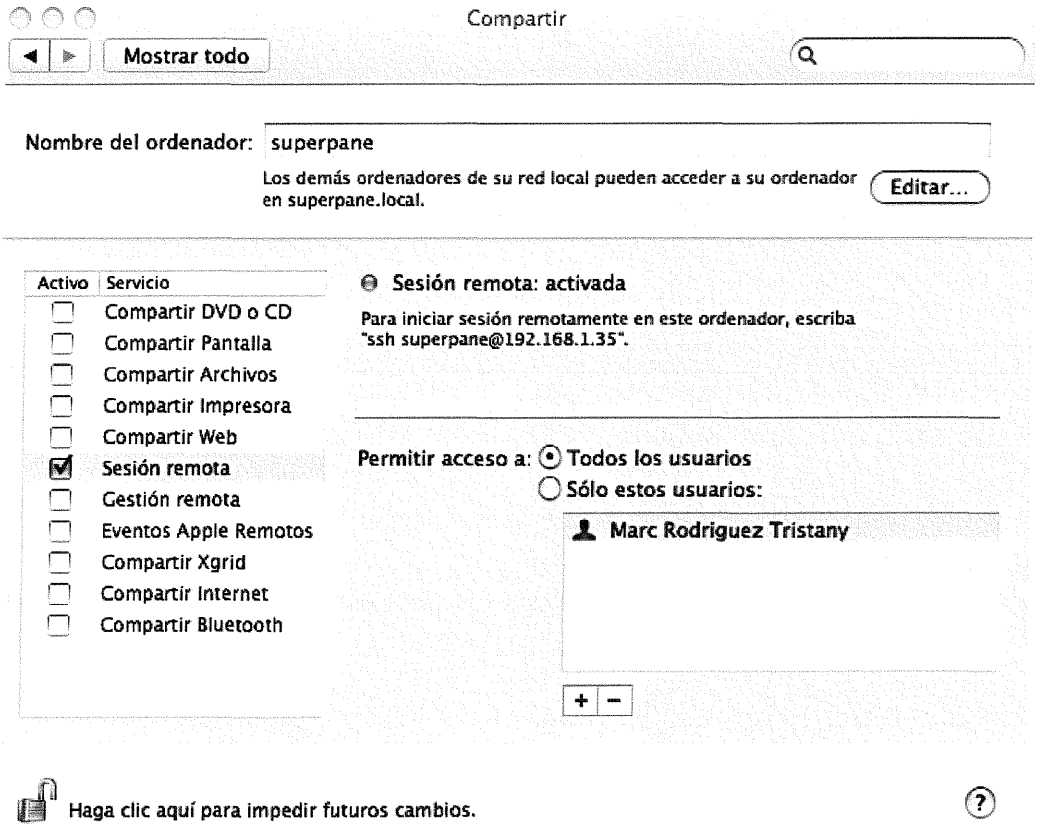


Figura 48: Activar sessió remota

Un cop tenim això fet, ja podem transferir el codi del joc, previament comprimit en format .tar.gz a Sugar, conectant-nos per sftp a la direcció IP virtual que ens proporciona VMware, que fa d'enllaç entre el sistema físic i virtual.

Ja connectats per sftp, només fa falta fer un *get* de l'arxiu que conté el codi. L'usuari el qual hem d'introduïr per tal de connectar-nos al servidor d'ftp és l'usuari de la màquina Mac.

Ja tenim el codi de Hobbies comprimit dins Sugar. El col·loquem a /home/olpc/ que és el directori bàsic d'usuari i el descomprimim mitjançant la comanda *tar xzvf nom_del_arxiu.tar.gz.*

Per altra banda, hem de descarregar una petita aplicació bàsica per a la creació de les activitats (arxius .xo). Aquesta és OLPCGames, que no fa més que generar un wrapper (tipus de paquet) per la nova activitat i afegir els diferents arxius de configuració i instal·lacio de l'activitat. La podem descarregar des del mateix browser de Sugar, buscant-la a la wiki de l'OLPC.

En el primer moment que vem probar de crear una activitat, existia la versió 1.4 i és amb aquesta que vem treballar en primera instància, per les proves dels dos jocs inicials, però en el moment de provar JClic traduit a Python en Sugar, vàrem veure que existia la versió 1.6, i és la que finalment hem usat.

Tant la 1.4 com la 1.6 treballen de forma similar, però tenen alguna que altra diferència, així que explicarem el funcionament de la 1.6 i seguidament citarem les petites diferències entre un i l'altre.

Ja tenim descarregada doncs l'aplicació que ens serveix per a crear les activitats. L'hem de descomprimir també de la mateixa manera que hem vist anteriorment, i ja podem començar a crear l'activitat Hobbies.

El primer que hem de fer és entrar a la carpeta OLPCGames-1.6/skeleton. Allà trobarem un script en Python, buildskel.py, que hem d'executar mitjançant la comanda *./buildskel.py -n nom_de_la_activitat -t titol_de_la activitat -m*

nom_de_la_classe_principal_de_la_aplicacio:nom_de_la_funcio_que_executa_la_aplicacio, on com veiem, hem de dir-li el nom que voldrem donar a l'aplicació, el títol que tindrà quan s'executi en Sugar, el nom de la classe principal de l'aplicació i el nom de la funció principal que està continguda dins la classe principal.

En el nostre cas seria *./buildskel.py -n hobbies -t "Els Hobbies" -m Main:main*

Quan s'executa la comanda, en el mateix directori skeloton es crea una carpeta amb el nom que li hem donat a l'aplicació mes *.activity.* En el nostre cas quedaria com *hobbies.activity/.*

Si entrem a dins la carpeta creada, veurem que ja conté diferents arxius i subcarpetes. Aquests son els que doten a la aplicació de la qual volem crear l'activitat dels elements necessaris per tal que així pugui ser. Podem veure una subcarpeta /activity i arxius com setup.py i MANIFEST.in, entre d'altres, que haurem de retocar perquè tot funcioni correctament.

```
[olpc@xo-82-67-11 skeleton]$ cd hobbies.activity/
[olpc@xo-82-67-11 hobbies.activity]$ ls
activity     buildmanifest.py  NEWS        POTFILES.in  setup.py
activity.py  MANIFEST.in       olpcgames   run.py
[olpc@xo-82-67-11 hobbies.activity]$ pwd
/home/olpc/OLPCGames-1.4/skeleton/hobbies.activity
[olpc@xo-82-67-11 hobbies.activity]$
```

Figura 49: Carpeta /hobbies.activity

El següent pas és passar tots els arxius que conté la nostra aplicació a dins de la carpeta *hobbies.activity*. Hem de copiar-hi tots els arxius que hi han, també els elements multimedia i subcarpetes amb els seus arxius pertinents.

Un cop tenim tots els arxius copiats arriba el moment de tocar algun dels arxius de configuració que conté la carpeta.

Primer de tot hem de mirar l'arxiu *activity.py,* que conté els noms de l'activitat, de la classe principal i de la funció principal. En teoria ja hauria d'estar bé, ja que li hem indicat al moment de crear el wrapper, però per assegurar-nos-en, obrim l'arxiu amb el *vi* mateix i mirem si la informació és correcta. Si és així no cal tocar res, si no, canviem el que sigui necessari.

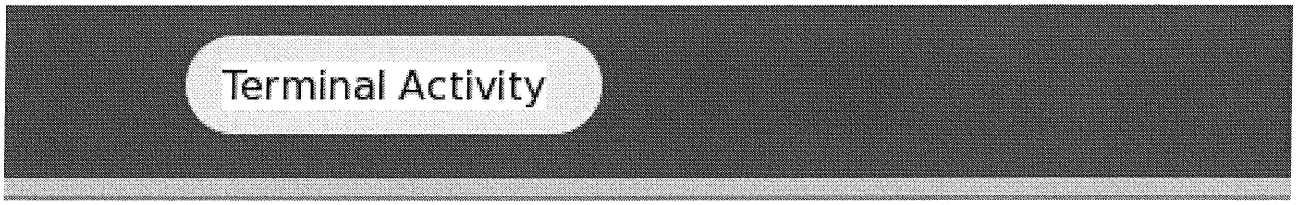
Un dels altres arxius que ens hem de mirar és el MANIFEST.in, que no és més que l'arxiu que agrupa tots els noms dels arxius que conté la carpeta de l'activitat, juntament amb les seves subcarpetes, i que un cop s'executi l'script que genera realment l'activitat, es guardaran en l'arxiu MANIFEST.

Aquí haurem d'afegir una línia si en l'aplicació que li hem passat tenim alguna subcarpeta, ja que si no se l'hi explicita, no ens la trobarà. En el cas que no tingui cap subcarpeta, no caldrà tocar res.

Si no és així, la línia que s'ha d'afegir és del tipus següent: *recursive-include nom_de_la_subcarpeta *.extensio_de_l'arxiu_1 *.extensio_del_arxiu_2 ... *.extensio_del_arxiu_n.*

En el cas de la nostra aplicacio Hobbies, com que tenim una subcarpeta on hi guardem les imatges, haurem d'afegir la línia que queda de la següent forma: *recursive-include media *.jpg *.gif *.png.*

A sota podem veure l'arxiu MANIFEST.in en el cas de la nostra aplicació, un cop modificat.




```
include *.py *.txt *.png *.jpg NEWS README
recursive-include media *.jpg *.gif *.png
recursive-include activity *.svg *.info
include olpcgames/COPYING
recursive-include olpcgames *.py
~
~
```

Figura 50: MANIFEST.in

Els altres fitxers que conté la carpeta i que no son de l'aplicació que hi hem copiat no els hem de tocar, els deixem tal i com estaven.

Arribats en aquest punt, ja estem en disposició de crear l'activitat en si, o sigui, crear l'arxiu amb extensió *.xo* per a ser instal·lada posteriorment en qualsevol màquina XO i ser executada i utilitzada per tots els usuaris.

Per a fer-ho haurem d'executar un dels arxius que estan dins la carpeta creada de l'activitat, el *setup.py.* Es tracta d'un script escrit en Python que ens genera l'arxiu activitat instal·lable.

Per tal de poder-lo executar, primer li haurem de donar permisos per a tal efecte. La comanda que hem de posar serà *chmod 755 setup.py.* Podem comprobar que si no li donem els permisos pertinents i provem d'executar l'script, no podrem.

Així doncs, ja podem executar l'script, de la forma habitual, mitjançant la següent sentencia: *./setup.py dist.*

Després d'executar aquesta sentència, veurem que ens apareix l'arxiu *.xo* amb l'activitat creada a dins la carpeta de l'activitat, en el nostre cas *hobbies.activity*, com podem veure en la figura de sota.

```
[olpc@xo-82-67-11 hobbies.activity]$ ls
activity     buildmanifest.py  MANIFEST.in  olpcgames     run.py
activity.py  hobbies.xo        NEWS         POTFILES.in   setup.py
[olpc@xo-82-67-11 hobbies.activity]$
```

Figura 51: Activitat creada

Com hem dit anteriorment, l'aplicació utilitzada per a crear les activitats, OLPCGames, té dos versions que nosaltres hem utilitzat. Tot i que finalment, per a la creació de l'activitat JClic per a Sugar hem utilitzat la versió 1.6, durant el transcurs del projecte, vàrem utilitzar la versió 1.4, ja que la posterior encara no era estable.

Entre una versió i altra hi han molt poques diferències, però si que cal destacar-ne una.

En el moment en que hem modificat l'arxiu MANIFEST.in, en la versió 1.6, executem seguidament l'script que ens crea l'activitat i aquest mateix ja ens genera l'arxiu MANIFEST amb tots els noms dels arxius continguts dins d'aquesta.

En canvi, en la versió 1.4, això no passa, i som nosaltres mateixos els que hem de crear aquest arxiu MANIFEST. Això ho farem a partir d'un altre script escrit en Python anomenat *buildmanifest.py,* que no és més que un petit programa escrit també en Python que ens crea l'arxiu i hi guarda els noms dels arxius que són

continguts dins del directori d'on pertany i dels arxius que hi ha a les subcarpetes d'aquest directori, de forma recursiva.

Un cop executat el *buildmanifest.py*, i si no ens dona error, ja podrem executar el *setup.py,* per tal de crear l'arxiu *.xo*.

A sota podem veure com queda l'arxiu MANIFEST, un cop creat. Podem diferenciar els arxius que estan en l'arrel de la carpeta o que estan dins d'una subcarpeta, com pot ser en el nostre cas, /media.

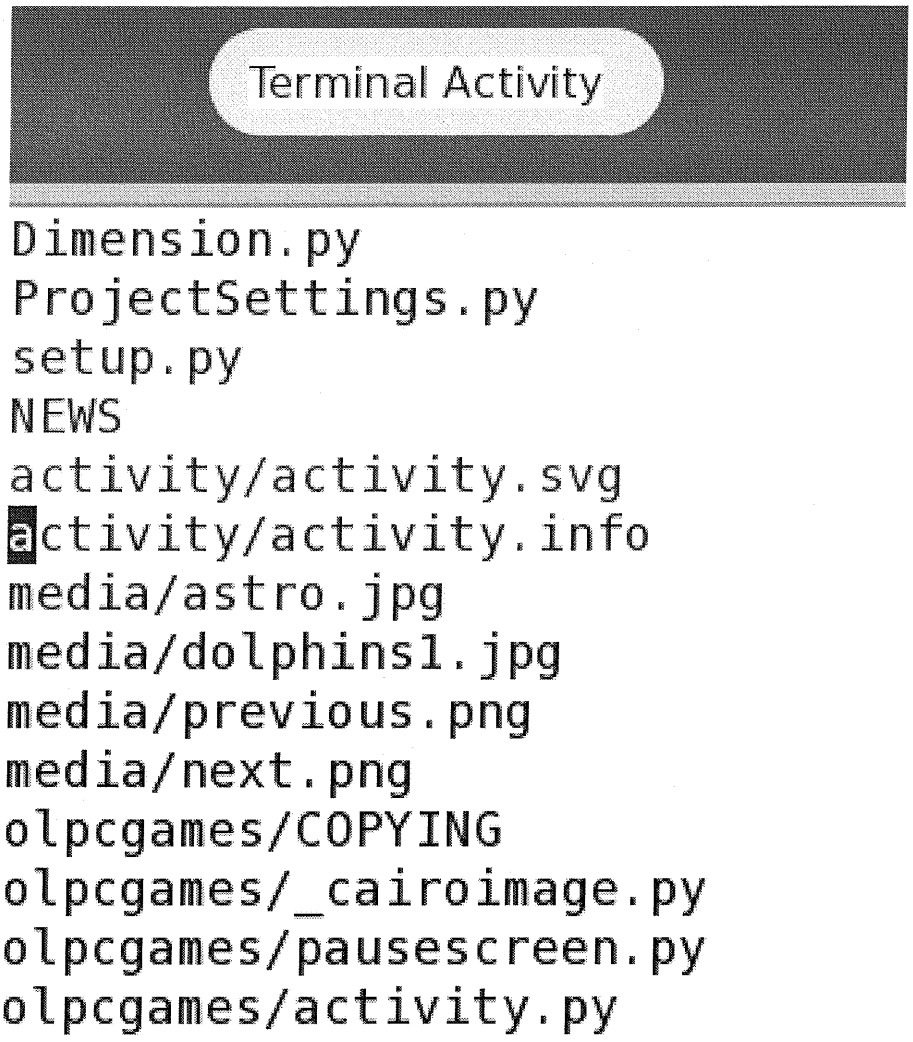


Figura 52: MANIFEST creat

XII.MANUAL D'INSTALACIÓ DE JCLIC EN SUGAR

Un cop hem explicat com crear una activitat en Sugar i comprimir-la en un arxiu *.xo*, passarem a explicar com instal·lar-la, tant una de creada per nosaltres com una de descarregada del web.

A. INSTAL·LACIÓ D'ACTIVITAT CREADA PER NOSALTRES

Un cop hem creat l'activitat, ja podem passar a instal·lar-la al nostre sistema. Ho explicarem seguint amb l'exemple de l'activitat Hobbies.

Com hem dit anteriorment, l'arxiu activitat ha estat creat a la carpeta */home/olpc/OLPCGames-1.6/skeleton/hobbies.activity.*

El que hem de fer és copiar-la a la carpeta */home/olpc/Activities* amb la comanda *cp hobbies.xo /home/olpc/Activities*.

Si la carpeta *Activities* encara no està creada, la creem des de */home/olpc* amb la comanda *mkdir Activities.*

Un cop copiada l'activitat a la carpeta apropiada, l'hem de descomprimir, mitjançant la comanda *unzip hobbies.xo.* Un cop descomprimida, tindrem a la carpeta el que veiem a sota:

```
[olpc@xo-82-67-11 Activities]$ ls
hobbies.xo  hobies.activity
[olpc@xo-82-67-11 Activities]$
```

Figura 53: Activitat descomprimida

Tenim doncs, l'arxiu .xo i l'activitat descomprimida.

Ja només falta reiniciar la màquina i automàticament l'activitat estarà instalada en Sugar, i la podrem trobar al menú amb la seva icona, que si no hem canviat, serà un martell, amb el nom Hobbies.

B. INSTAL·LACIÓ D'ACTIVITAT DESCARREGADA

També podem instal.lar altres activitats que no siguin creades per nosaltres, des de la wiki d'OLPC, on hi ha un repositori, com podem veure en la imatge de sota.

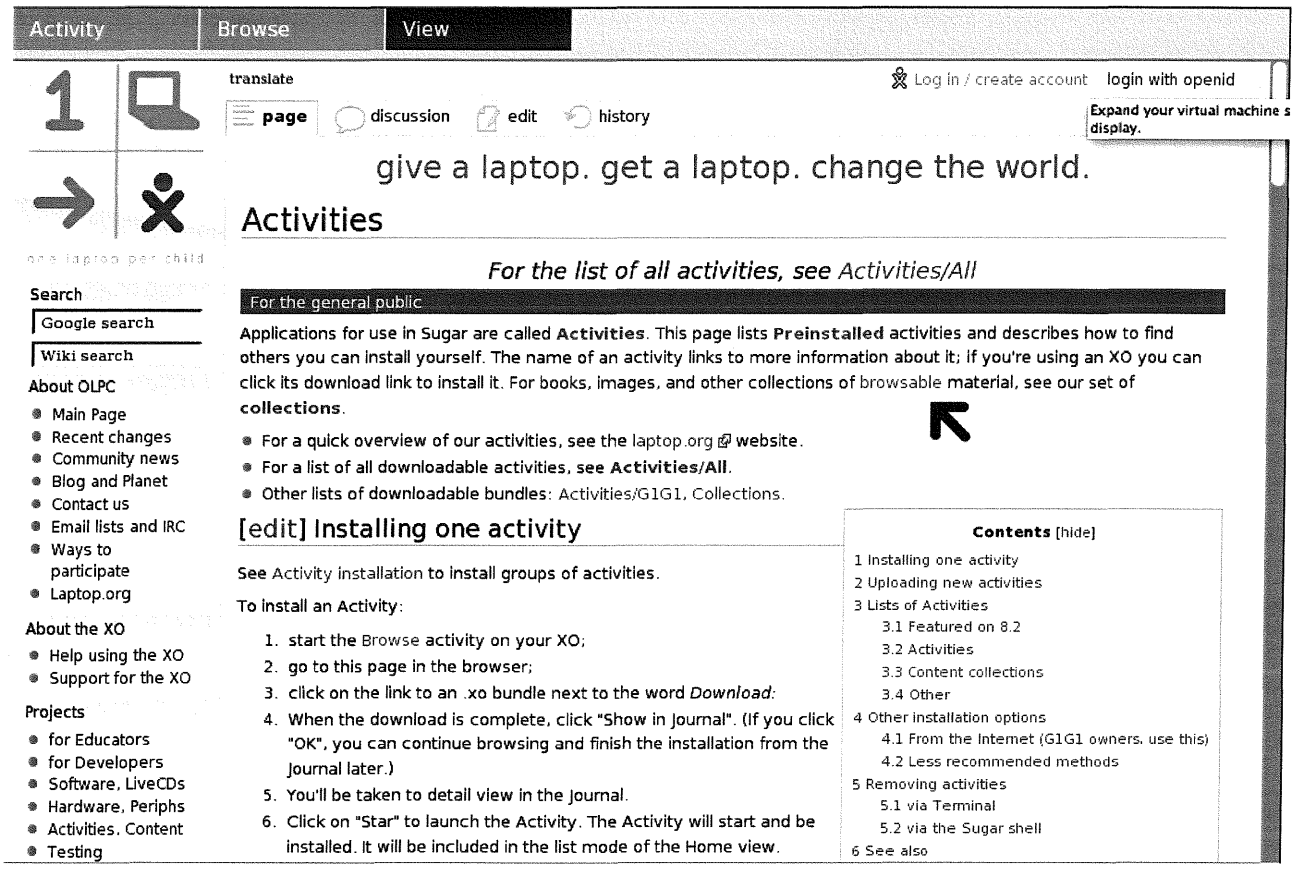


Figura 54: Repositori activitats

Així doncs, des d'aquí descarreguem qualsevol activitat, clicant al link que volguem.

Un cop cliquem a l'arxiu .xo a descarregar, ens demanarà si el hi estem d'acord, i li diem que OK.

Ja tenim descarregat l'arxiu, i aquí és on ens demana si volem continuar sense executar-lo, o si volem obrir-lo. Cliquem a Open i ell mateix ja ens instal·la l'activitat.

És en aquest moment en que al menú ens hauria d'apareixer ja l'activitat amb la seva icona. Si no sortís, reiniciem la màquina.

Ja podem utilitzar doncs l'activitat descarregada.

Cal dir també que l'activitat, com l'altra, queda guardada en la carpeta */home/olpc/Activities*.

XIII.CONCLUSIONS

A. OBJECTIUS ASSOLITS

- Instal·lació i configuració de l'entorn de desenvolupament, tant de l'interpret de Python com de les llibreries necessàries (Pygame,wx), entre d'altres, del IDE, que en el nostre cas ha sigut Eclipse, que hem hagut de configurar per tal que pogués interpretar el Python, i VMWare per tal de poder virtualitzar el sistema operatiu Sugar.

- Aprenentatge, tant del funcionament de l'entorn de desenvolupament Eclipse, com del llenguatge de programació Python i Pygame, nous per a nosaltres, com també de l'entorn Sugar i la seva configuració. Per tal d'entrar en contacte amb tots aquests conceptes, i així començar-los a dominar, vàrem implementar dos activitats JClic.

- Implementació del JClic Player amb el corresponent estudi del diagrama de classes, l'estructura dels fitxers que porten la informació de cada activitat, en format .jclic (realment un fitxer XML), i a partir d'aquí, implementació del parsejador de l'activitat, el visor d'aquestes, els tres tipus de puzles escollits, i finalment les proves pertinents per comprovar que totes les funcionalitats treballin de forma correcta.

- Adaptació de tot el projecte (implementat en Mac) al sistema operatiu Sugar.

B. FEINA FUTURA

A partir d'aquest punt, on el Player de JClic ha estat traduït de Java a JClic per a uns quants tipus d'activitats, la feina futura a realitzar per a fer realitat la traducció total i implantació generalitzada de JClic al projecte OLPC, caldria implementar el Player per als altres tipus d'activitat i començar amb la traducció del JClic Author.

Una altra feina futura seria millorar la forma com executar JClic en Sugar i pensar la forma de poder obrir-lo de diferents formes segons les preferències de l'usuari, obrint ell mateix el projecte JClic que desitgi, o bé que JClic s'executi directament al descarregar un projecte des del web.

C. CONCLUSIONS PERSONALS

Un cop acabat el projecte, i a punt de defensar-lo, puc fer ja una valoració global del què ha estat per a mi i les sensacions que he tingut al realitzar-lo.

En un principi, quan vem tenir el projecte escollit al davant, el primer pensament va ser, Com farem això? Com ens en sortirem? Com comencem?

Aquestes són sempre les primeres preguntes que un es fa davant d'una tasca que penses que és impossible o molt dificil de fer-se realitat, a més amb el nom que té i que de bon principi ja intimida, Projecte de Final de Carrera.

Però com tot, les coses no sempre són exactament com un es pensa que són de bon principi, i sent constant i prenent-se les coses seriosament, els problemes que en un principi et penses que tindràs i que no sabràs solucionar, van desapareixent a mesura que es va treballant i així es van transformant en un enriquiment personal constant.

Això ha sigut exactament el que ha passat, que amb constància, esforç i els coneixements adquirids durant la carrera, el projecte ha anat tirant endavant fins al dia d'avui, a punt d'arribar al dia de defensar-lo.

Com he dit, i vull remarcar, la realització d'aquest projecte ha sigut possible gràcies als coneixements adquirits durant tots els anys d'universitat. Aquest coneixement no es basa només en els conceptes concrets que hem anat aprenent durant aquesta etapa, sinó en la capacitat adquirida per aprendre ràpidament, buscar-te la vida per aconseguir una cosa que mai havies vist o realitzat, la manera d'enfrontar-te als problemes, la manera de solucionar-los, la manera d'adaptar-se a aquest món anomenat informàtica vesat al canvi constant. Aquests són realment els coneixements que personalment crec més importants per a un Enginyer Informàtic, per tal d'enfrontar-nos a l'etapa que ens espera a partir d'ara.

Un cop arribats en aquest punt, puc dir que en aquest mesos realitzant el projecte he après principalment, apart de el que ja he esmentat, a treballar amb diferents

eines i tecnologies noves per a mi i adaptar-me a elles, a treballar conjuntament, a entendre el projecte OLPC, però sobretot a adonar-me de la necessitat que tenim de fer arribar aquestes màquines que anomenem ordinadors, que per a la nostra societat son un element més que imprescindible, a països que no tenen coneixement d'aquesta cultura de la tecnologia, a causa de la seva situació econòmica i cultural.

A part d'aquests coneixements adquirits, ha sigut una grandíssima satisfacció personal el fet d'haver pogut realitzar un projecte per a un ordinador solidari, un projecte per fer arribar la informàtica als llocs més hinòspits del món, i que un nen de qualsevol lloc del planeta pugui disfrutar jugant amb una eina realitzada amb les nostres pròpies mans.

XIV.AGRAÏMENTS

Aquest apartat serveix per rendir un petit homenatge a tota la gent que ens ha donat suport durant el temps de realització del projecte i també durant la carrera.

Primer de tot donar gràcies a la Mª José per l'esforç realitzat perquè aquest projecte tirés endavant i pel tracte que hem rebut, i en general, a tots els professors, per ensenyar-me tot el què sé.

També donar les gràcies a tots els companys que he tingut durant la carrera, per haver passat amb ells grans moments, tant de classe, d'estudi, de bar, de festa, entre molts d'altres que sempre tindré guardats a la memòria.

També, com no, he de donar gràcies a la meva familia, i especialment als meus pares, per haver-me donat tot el que tinc i que sóc, i tot el que han fet perquè hagi pogut arribar fins aquí. També al meu germà, per ser el meu germà, simplement.

I finalment, com no podia faltar, donar gràcies al meu company de projecte, l'Aleix, company des dels inicis de la carrera, perquè hem creat un equip fantàstic i perquè n'hem passat de tots colors, tensions, riures, bones estones, també de dolentes, nervis, etc, però que al final hem arribat on voliem.

I ja per acabar, gràcies a tothom que ha estat al meu costat tot aquest temps, als que s'alegren per mi, als que m'han donat suport, m'han ensenyat, m'han animat, i a tota la gent que em fa riure dia rere dia.

XV.ANNEXOS

A. MANUAL D'USUARI JCLIC PER OLPC

En aquest apartat tenim el manual d'usuari de JClic per OLPC, on explicarem les funcionalitats bàsiques i el que es pot fer amb l'aplicació, encara que com hem dit, hem prioritzat la facilitat d'ús per sobre d'altres requeriments, i això comporta que el manual sigui bastant trivial.

JClic per OLPC s'executa mitjançant l'icona que ens apareix al menú de Sugar amb el nom de JClic, que executarà un conjunt d'activitats que ja té carregades per defecte.

Si el projecte carregat conté activitats que no són del tipus que hem implementat per al nostres sistema, o sigui, ExchangePuzzle, DoublePuzzle o HolePuzzle, el mateix sistema les descartarà i no les carregarà. Només carregarà si són dels tipus que acabem de dir.

Un cop tenim el projecte JClic carregat, la interfície gràfica s'inicialitza i ens apareix la primera activitat que és d'un dels tipus abans especificats. En aquest moment, l'usuari ja pot començar a jugar amb l'activitat.

A sota podeu veure tres imatges dels tres tipus d'activitats a les quals es pot jugar.

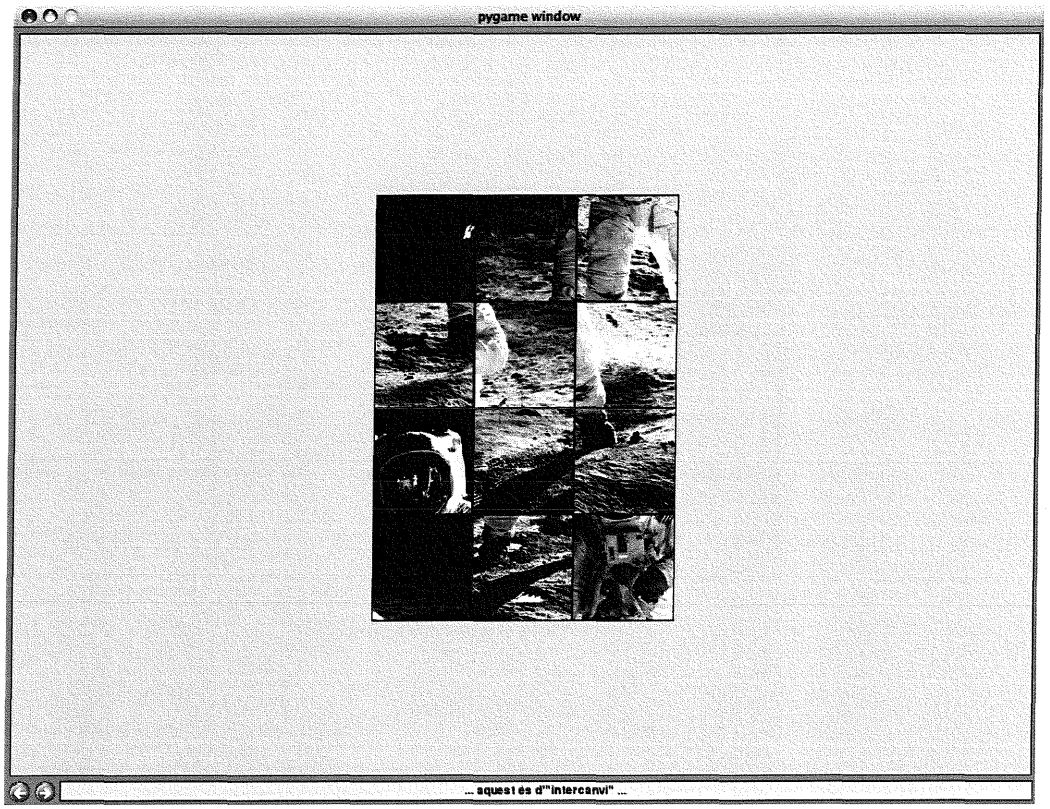


Figura 55: Exchange Puzzle

Figura 56: Hole Puzzle

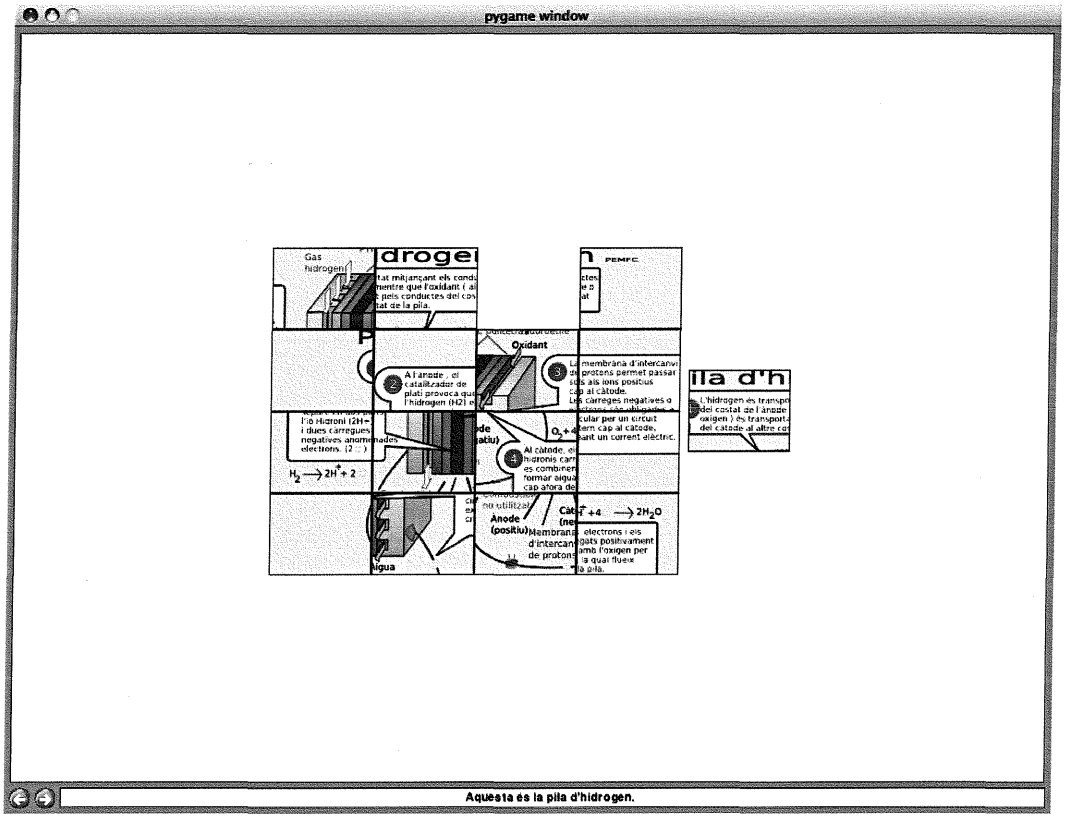


Figura 57: Hole Puzzle

Un cop hem acabat l'activitat o simplement volem passar a la següent o anterior, hem de clicar un dels dos botons de què disposem, un per tirar a la següent activitat disponible i l'altre per anar a l'anterior. Això ens carregarà o bé la següent o l'anterior, respectivament.

Si estem a la primera activitat i cliquem el botó d'anar a l'activitat anterior, es carregarà l'última activitat que conté el projecte.

Finalment, si es vol sortir de l'aplicació, hem de clicar la tecla Q.

1. Modificació del fitxer de configuració

Existeix en el nostre projecte instalat a Sugar un fitxer de configuració que s'utilitza per a carregar el projecte JClic per defecte.

En el fitxer de configuració de l'aplicació trobem 4 paràmetres els quals podem modificar:

projectPath: es tracta del path del projecte que estem executant. Tant pot ser un projecte amb extensió .jclic (sempre i quan en la mateixa carpeta on hi ha el fitxer .jclic hi hagi tots els fitxers multimèdia que es facin servir en el projecte) com un fitxer .jclic.zip.

nextPath i previousPath: es tracta dels paths de les icones que es fan servir per al botó de següent i anterior activitat. Cal dir que no cal que tinguin cap mida en especial, doncs el programa els encongeix fins a les mides desitjades. Formats suportats: jpeg, gif, i png.

backgroundColor: es tracta del color de fons del Player (no del quadre d'activitat). Cal que sigui una tupla de 3 valors en format RGB.

Notar que en el cas del paths han de ser paths complets, és a dir, desde el directori arrel del ordinador on estem executant la apliació

B. MANUAL D'USUARI HOBBIES

Per a poder jugar a Hobbies un cop instal·lat en l'XO, hem de clicar en l'icona de Hobbies (un martell) en el menú del Sugar.

Un cop hem entrat el programa ens sortirà una pantalla com la de sota, on podem clicar a cada imatge, pertanyent a diferents tipus de hobbies. Al clicar a una d'elles al requadre gris de sota ens sortirà la paraula en anglès que correspon a la imatge clicada.

Aquest pas serveix per recordar el nom de cada hobbie. Quan ja ens hem après els noms, ja podem clicar al botó Jugar.

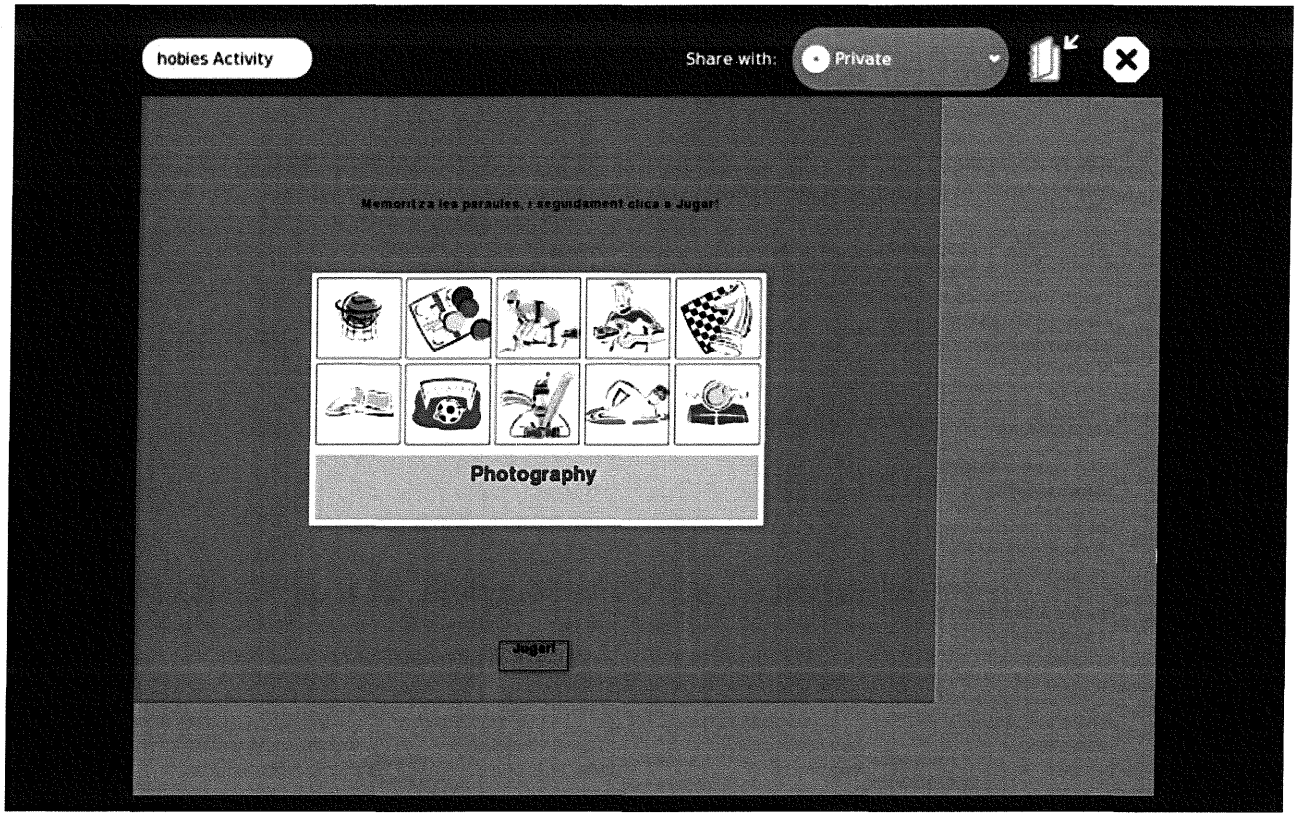


Figura 58: Hobbies

Ens apareix en aquest moment una altra pantalla amb les mateixes imatges, però a sota totes les paraules que s'identifiquen amb alguna de les imatges.

El que hem de fer és relacionar cada imatge amb el seu nom, mitjançant el ratolí, seleccionant primer una imatge o paraula, i seguidament ens apareixerà una fletxa que hem de fer arribar amb el que pensem que és la seva parella.

Si encertem, els dos requadres es tornaran de color gris, com veiem a la imatge de sota, sino, ens els deixarà com estaven.

Hem de realitzar aquesta tasca per cada parella.

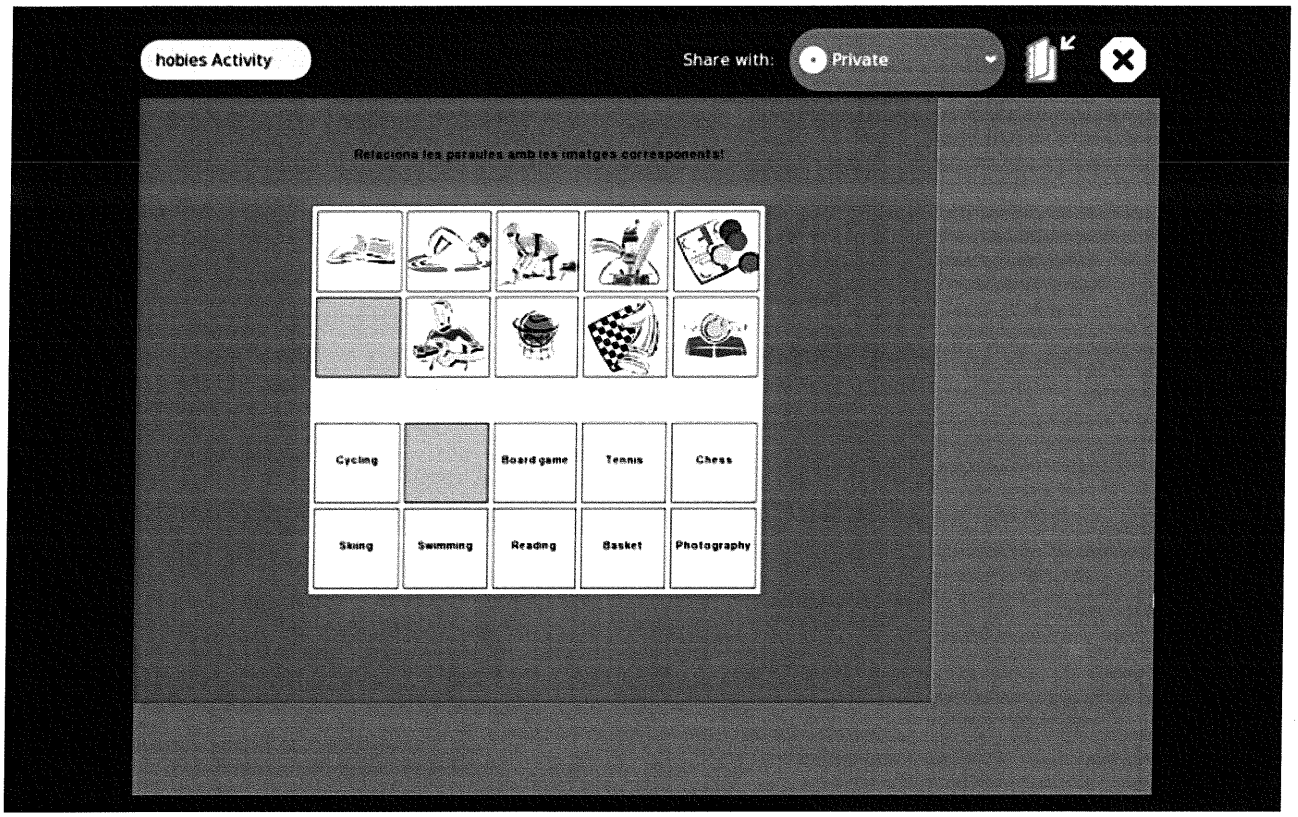


Figura 59: Hobbies

Un cop tinguem tots els requadres de gris, com veiem en la imatge de sota, el joc haurà acabat, ens felicitarà.

Ja hem acabat. Per a sortir de l'aplicació, hem de clicar el botó de tancar de la part superior dreta de la pantalla, com veiem en la imatge.

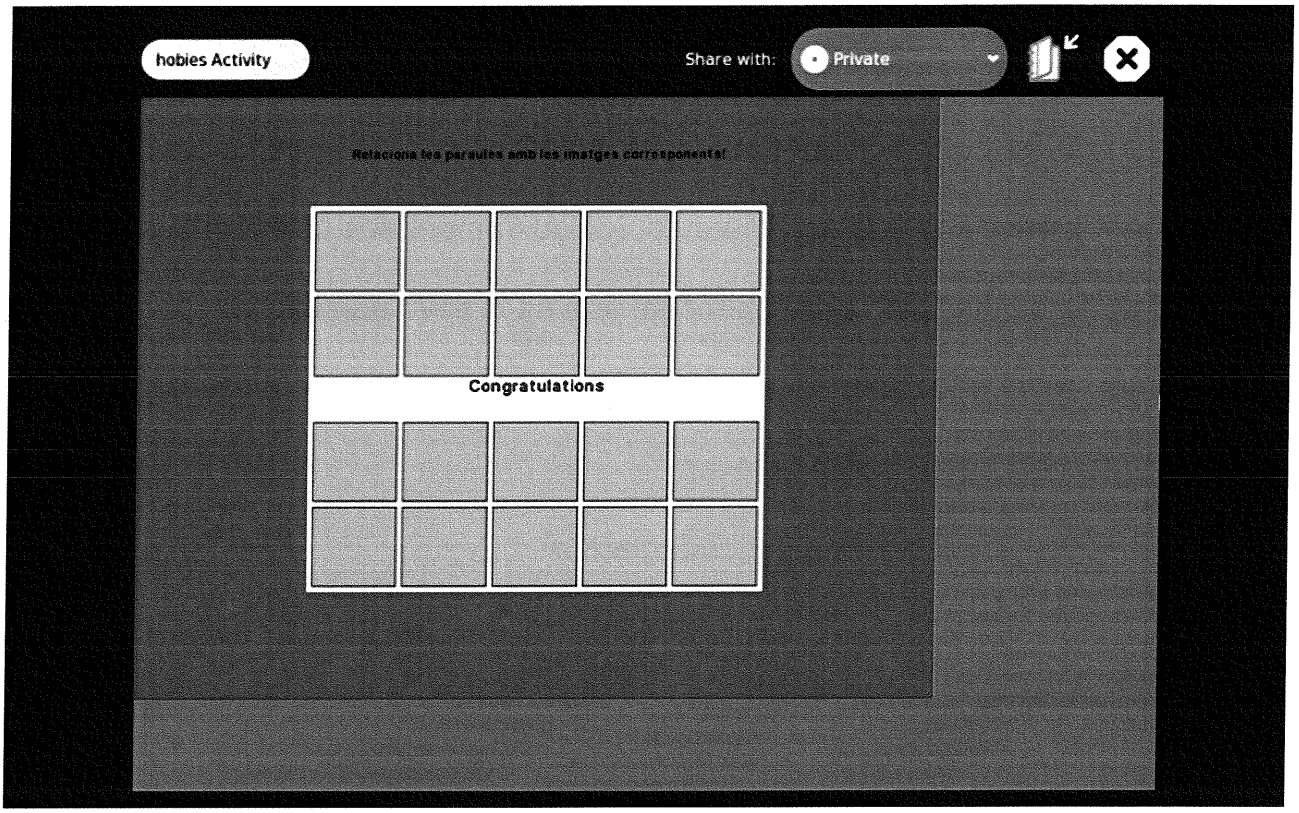


Figura 60: Hobbies

C. MANUAL D'USUARI ELS CONILLS

Per a poder jugar a El Conill, un cop instal.lat en l'XO, hem de clicar en l'icona d'El Conill (un martell) en el menú del Sugar.

Un cop hem entrat el programa ens sortirà una pantalla com la de sota,que només és una presentació del joc. Hem de clicar el botó de Som-hi, i entrarem al joc.

Figura 61: El Conill

En aquesta primera pantalla, que veiem a sota, hem de trobar i seleccionar a sobre de totes les paraules "conill" que trobem en la definició de conill que veiem per pantalla.

Podem seleccionar qualsevol de les paraules que conté el text, per tant ens podem equivocar.

Un cop creiem que hem seleccionat totes les que hi han, hem de clicar el botó Avalua't.

L'aplicació ens evaluarà si les hem trobat totes o no. Si no és així, haurem de revisar les nostres seleccions i tornar a clicar Avalua't quan creguem que ho tenim correcte.

Si ho tenim correcte, ens apareixerà un botó per seguir endavant en el joc.

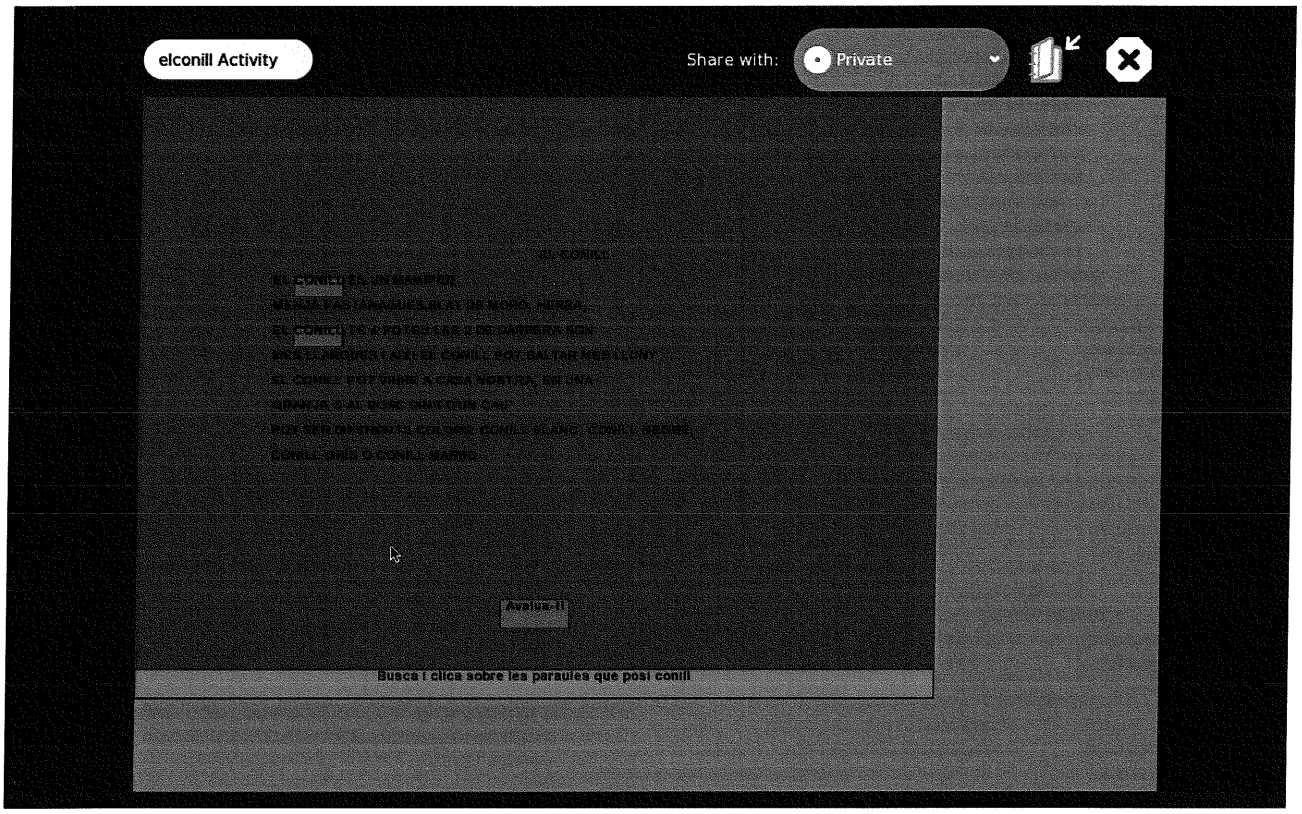


Figura 62: El Conill

Si hem clicat el botó endavant, entrarem al segon joc que conté l'activitat El Conill, que no és més que una sopa de lletres, on hem de trobar les diferents parts del cos d'el conill, resseguint les lletres de la sopa on hi hagi cada paraula. En el moment en que en trobem una, ens apareixerà al costat una imatge d'aquesta part del cos, com podem veure en la imatge de sota, i la paraula es pintarà de negre dins la matriu de lletres.

En el moment en què trobem totes les paraules dins de la sopa de lletres, l'aplicació ens avisarà de que l'hem acabat i el joc s'acaba.

Per sortir-ne, com abans, hem de clicar el botó de sortir a la part superior dreta de la pantall.

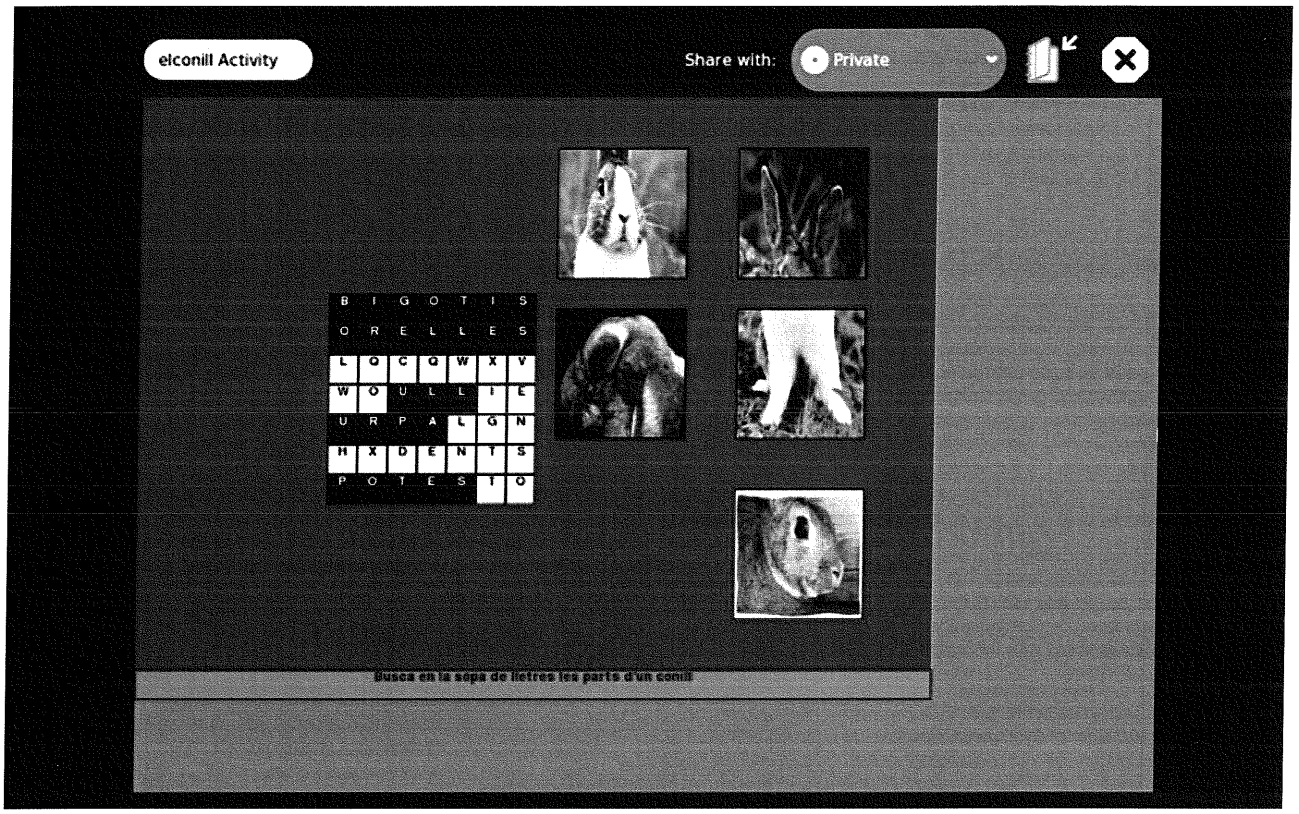


Figura 63: El Conill

XVI.BIBLIOGRAFIA

[1] morguapu.upc.es/crom

[2] http://www.box.net/files

[3] http://wiki.python.org/moin/PythonXml

[4] http://epydoc.sourceforge.net/stdlib/xml.dom-module.html

[5] http://wiki.laptop.org/go/Journal_Activity

[6] http://wiki.laptop.org/go/Sugar.activity.activity

[7] http://wiki.laptop.org/go/Sugar.mime

[8] http://wiki.laptop.org/go/PyGTK/Hello_World_Tutorial

[9] http://wiki.laptop.org/go/Activity_Bundles

[10] http://wiki.laptop.org/go/Game_development_HOWTO#Installing_and_Testing

[11] http://wiki.laptop.org/go/Activity_tutorial

[12] http://wiki.laptop.org/go/Creating_an_activity

[13] http://wiki.laptop.org/go/Porting_pygame_games_to_the_XO

[14] http://www.fabioz.com/pydev/manual_101_install.html

[15] http://www.pygame.org/wiki/about

[16] http://www.lsi.upc.edu/~lpv/pdf.dir/Requeri.pdf

[17] http://wiki.laptop.org/go/Activities

[18] bradmontgomery.blogspot.com/2007/09/pygame-on-os-x-with-python-25.html

[19] http://www.box.net/files

[20] http://wiki.python.org/moin/PythonXml

[21] http://epydoc.sourceforge.net/stdlib/xml.dom-module.html

[22] http://wiki.laptop.org/go/Journal_Activity

[23] http://wiki.laptop.org/go/Sugar.activity.activity

[24] http://wiki.laptop.org/go/Sugar.mime

[25] http://wiki.laptop.org/go/PyGTK/Hello_World_Tutorial

[26] http://wiki.laptop.org/go/Activity_Bundles

[27] http://wiki.laptop.org/go/Game_development_HOWTO#Installing_and_Testing

[28] http://wiki.laptop.org/go/Activity_tutorial

[29] http://wiki.laptop.org/go/Creating_an_activity

[30] http://wiki.laptop.org/go/Porting_pygame_games_to_the_XO

[31] http://www.fabioz.com/pydev/manual_101_install.html

[32] http://www.pygame.org/wiki/about

[33] http://www.lsi.upc.edu/~lpv/pdf.dir/Requeri.pdf

[34] http://wiki.laptop.org/go/Activities