Títol: Activitats JClic per a OLPC

Volum:1

Alumne: Roger Civit i Soler

Director/Ponent: Marc Alier Forment

Departament: ESSI

Data:

DADES DEL PROJECTE

Títol del Projecte: Activitats JClic per a OLPC

Nom de l'estudiant: Roger Civit i Soler
Titulació: Enginyeria Tècnica en Informàtica de Gestió
Crèdits:22,5
Director/Ponent: Marc Alier Forment
Departament: ESSI

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Enrique Mayol Sarroca

Vocal: Mònica Sánchez Soler

Secretari: Marc Alier Forment

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

ÍNDEX DE CONTINGUTS

1. INTRODUCCIÓ

1.1 Pròleg

La majoria dels dos mil milions de nens de països en desenvolupament no tenen una educació adequada o no en reben cap. Les conseqüències tant individuals com socials d'aquesta crisi global crònica són profundes. Els nens són relegats a la pobresa i aïllament sense saber el que els coneixements podrien aportar a la seva vida.

D'aquesta manera i per ajudar fomentar l'educació d'aquests nens sorgeix la One Laptop Per Child (OLPC), una organització que vol aportar el seu granet de sorra en aquest món fent-ho per la via tecnològica.

Més concretament, la OLPC és una organització creada per a promoure la creació d'un dispositiu educatiu accessible als països en desenvolupament. Segons ells, la seva missió o meta és la següent:

> *"Proveir els nens de països en desenvolupament amb un ordinador portàtil robust, de baix cost, consum reduït, amb possibilitats de connexió (tant entre els propis ordinadors com a Internet) i amb un programari dissenyat especialment per a una educació col·laborativa i capaç de promoure l'auto-aprenentatge de forma divertida"*

A partir d'aquí es va tirar endavant aquest projecte i es va realitzar el portàtil XO, conegut també com a OLPC o "el portàtil dels 100 dòlars". L'XO és un petit portàtil amb funcionalitats i rendiment limitats creat així per tal de poder reduir el màxim el seu preu, aconseguint que els governs i escoles d'aquests països puguin assumir el seu cost. El fet que sigui d'Open Source permet que qualsevol desenvolupador pugui implementar les seves aplicacions col·laborant aportant d'aquest manera qualsevol millora.

És per això que ens hem unit un grup de persones per realitzar un projecte amb el qual hem volgut col·laborar amb la OLPC aportant una aplicació que esperem que pugui contribuir en l'educació d'aquests nens i nenes: el SugarClic. El SugarClic és una eina basada en l'adaptació del reproductor d'activitats del JClic, un conjunt d'aplicacions que està força estès en diverses escoles i instituts de Catalunya, i en altres països d'Amèrica del Sud, i és utilitzada com una eina educativa i de suport a diverses matèries.

1.2 Motivació

Aquest projecte, a més a més de les motivacions implícites que suposa qualsevol projecte de final de carrera com l'aplicació de coneixements apresos durant la carrera, també ha tingut altres motivacions pròpies.

Des del principi ha sigut una oportunitat molt atractiva, ja sigui per les raons humanitàries òbvies que persegueix, com són les de contribuir en la part que sigui possible a la millora de l'educació de nens i nenes dels països en desenvolupament, com per altres raons que tenen un atractiu també especial per a mi com són: involucrar-me en un grup de treball dinàmic i formar part del desenvolupament d'un projecte de codi obert ubicat a Internet.

1.3 Estructura

La memòria està dividida en tres grans parts:

Primerament una posada en context sobre el projecte SugarClic, definint de forma resumida i global el propi projecte i els elements que en formen part com són el JClic i l'OLPC. Per a una informació més detallada hi ha referències als annexos al final del projecte.

Posteriorment explicarem la nostra aportació en aquest projecte de forma ordenada (mitjançant la descripció dels objectius, un anàlisi de requisits, una especificació, quines eines hem utilitzat, el disseny, la implementació, la planificació i les conclusions arribades).

I finalment els Annexos, on podrem veure millor temes relacionat amb el JClic i la OLPC, la implementació més detallada de les activitats implementades o un parell de manuals: un per saber utilitzar el SugarClic i un altre per començar a desenvolupar activitats o qualsevol altra funcionalitat per al SugarClic. També es detalla a on està ubicat el codi desenvolupat del projecte.

2. CONTEXT

En aquest apartat s'explica a grans trets el projecte SugarClic, per tenir una visió global de tot el conjunt amb les aplicacions de tots els membres que hem treballat en ell.

2.1. Definicions, acrònims i abreviatures

Primerament farem una breu descripció dels noms i abreviatures que puguem utilitzar en aquest apartat o en la memòria en general:

- **JClic**: Conjunt d'aplicacions en Java que permeten realitzar diferents tipus d'activitats educatives multimèdia: trencaclosques, associacions, exercicis de texts, etc.
- **Clic**: Conjunt de jocs definits que són reproduïts pel JClic.
- **OLPC**: One Laptop Per Child. Organització sense ànim de lucre que tracta d'apropar l'educació a tots els nens del planeta realitzant portàtils de baix cost.
- **XO:** Portàtil creat per al projecte OLPC. També denominat com el portàtil dels 100 dòlars o l'OLPC, igual que el nom de l'organització.
- **Sugar**: interfície gràfica d'usuari desenvolupada per l'XO.
- **SugarClic**: projecte que té com a objectius realitzar un reproductor de clics dins de Sugar.
- **PortalClic**: portal web que té com a objectiu oferir Clics descarregables per l'aplicació del SugarClic.

2.2 Les Bases

En aquest apartat farem breus explicacions sobre el JClic, l'OLPC i el projecte SugarClic, per situar-nos al context.

2.2.1 El JClic

El JClic és un conjunt d'aplicacions creades pel Departament d'ensenyament de la Generalitat de Catalunya amb llicència pública general GNU GPL. Aquest conjunt d'aplicacions l'utilitzen els professors per crear activitats interactives on es treballen aspectes procedimentals de diverses àrees, des d'educació infantil fins a secundària. Aquestes activitats interactives són resoltes pels alumnes de forma que poden aprendre interactuant amb l'ordinador.

Figura 1: Captura de pantalla d'una activitat del JClic

El JClic suporta diferents tipus d'activitats predeterminades. Cada una d'aquestes activitats -com podrien ser un trencaclosques, jocs de memòria, exercicis d'omplir buits, sopes de lletres, etc.- són jocs on l'alumne haurà d'aplicar el seu enginy per resoldre-les.

2.2.1.1 Components JClic

El Projecte JClic està format per quatre aplicacions:

- **JClic player:**
  Aquest és el principal programa i el que hem traduït per la plataforma OLPC.

És un programa independent, que un cop instal·lat permet realitzar les activitats des del disc dur de l'ordinador (o des de la xarxa) sense que calgui estar connectat a Internet.

- **Altres:**
  - **JClic applet:** Un "applet" que permet incrustar les activitats JClic dins d'una pàgina web. Fa la mateixa funció que el JClic player però des d'una pàgina web.
  - **JClic author:** L'eina d'autor que permet crear, editar i publicar les activitats d'una manera més senzilla, visual i intuïtiva.
  - **JClic reports:** Un mòdul de recollida de dades i generació d'informes sobre els resultats de les activitats fetes pels alumnes.

2.2.1.2 Activitats JClic

El JClic permet realitzar set tipus bàsics d'activitats, tot i que alguns d'aquests tipus presenten diverses modalitats, donant lloc a 16 possibilitats diferents. Els tipus d'activitats són els següents:

- Les associacions pretenen que l'usuari descobreixi les relacions existents entre dos conjunts d'informació.
- Els jocs de memòria on s'han d'anar descobrint parelles d'elements iguals o relacionats entre ells, que es troben amagats.
- Les activitats d'exploració, identificació i informació que parteixen d'un únic conjunt d'informació.
- Els trencaclosques plantegen la reconstrucció d'una informació que es presenta inicialment desordenada. Aquesta informació pot ser gràfica, textual, sonora... o combinar aspectes gràfics i auditius alhora.
- Les activitats de resposta escrita que es resolen escrivint un text (una sola paraula o frases més o menys complexes).
- Les activitats de text plantegen exercicis basats sempre en les paraules, frases, lletres i paràgrafs d'un text que cal completar, entendre, corregir o ordenar. Els textos poden contenir també imatges i finestres amb contingut actiu.
- Les sopes de lletres i els mots encreuats són variants interactives dels coneguts passatemps de paraules amagades.

L'ANNEX B: Activitats JClic (pàgina 97) consta d'una descripció més detallada de cada tipus d'activitat i de captures de pantalla de cada una.

2.2.1.3 Fitxers amb extensió .jclic

La informació corresponent a les activitats es guarda en un fitxer amb extensió .jclic. Aquests fitxers són documents XML que contenen la descripció completa d'un projecte JClic.
L'element arrel dels documents jclic porta el nom <JClicProject> i conté quatre elements principals:

- **<Settings>**
  Informació sobre els autors del projecte, descriptors temàtics, revisions, etc.
- **<Activities>**
  Conté elements del tipus <activity> que defineixen el funcionament i les característiques pròpies de cada activitat.
- **<Sequence>**
  Descriu l'ordre en què s'han de presentar les activitats i el comportament dels botons d'avançar i retrocedir.
- **<MediaBag>**
  Relació del nom i la ubicació de tots els ingredients necessaris per a executar les activitats: imatges, sons, vídeo, MIDI, fonts TTF etc.

2.2.2 OLPC

L'OLPC, sigles de One Laptop Per Child, que podríem traduir com *Un Portàtil Per a cada Nen,* és una iniciativa creada per promoure l'elaboració d'un ordinador portàtil educatiu de baix cost per tal d'apropar el món de la informàtica als infants dels països del tercer món.
Aquest ordinador portàtil (anomenat OLPC, XO o Portàtil dels 100 dòlars) està equipat amb components electrònics adequats per a l'ús que se li donarà, és a dir, té una potència reduïda, disminuint així el seu cost de fabricació. Aquest cost també és reduït gràcies a que aprofita el món de l'Open Source o Codi Obert de tal manera que permet que qualsevol desenvolupador pugui implementar les seves aplicacions, col·laborant així amb la millora i qualitat del portàtil.

Figura 2: Portàtil XO

L'XO és una eina d'aprenentatge creada especialment per als infants. Per a dissenyar-la van treballar en col·laboració experts del món acadèmic i de la indústria informàtica. El resultat és un dispositiu flexible, resistent, de baix cost i amb un ús molt eficient de l'energia, que pretén que les nacions emergents del món puguin fer arribar les noves tecnologies a l'educació primària.

El portàtil consta d'un sistema operatiu basat en Linux Fedora (anomenat Sugar) limitat, amb una interfície molt simple i intuïtiva i proporciona de sèrie unes quantes activitats i jocs pels nens ja instal·lades. També disposa d'accés a la xarxa via wi-fi perquè es pugui connectar a Internet o amb altres OLPC.

Per a més informació, l'ANNEX C: EL PROJECTE OLPC (pàgina 106) fa una referència completa a tot el relatiu amb l'OLPC.

2.3 El projecte SugarClic

2.3.1 Idea principal

Per una banda tenim l'ordinador portàtil de baix cost OLPC, que aprofita el món del codi obert i que té una manca de contingut educatiu. Per altra banda tenim el JClic, un projecte també de codi obert que gaudeix de més de 1200 projectes o clics al seu portal, i alguns més repartits en altres portals.

A partir d'aquí sorgeix la idea de realitzar el JClicPlayer, és a dir, el reproductor d'activitats clic, per a la plataforma OLPC adaptant-ne el rendiment i funcions tenint en compte les restriccions de hardware amb què compta el portàtil.

Posteriorment hem ampliat també les seves funcionalitats de tal manera que no només hem fet un reproductor de clics, sinó que hem dut a terme un gestor de clics, que permet entre d'altres coses descarregar activitats clic des d'una mateixa pàgina web unificada.

En el SugarClic hem treballat diferents projectistes: en Jose Camallonga (encarregat de desenvolupar l'aplicació), el Marc Benito (desenvolupador web i d'activitats clic), en Carlos Castilla (desenvolupador d'activitats clic) i jo, en Roger Civit (desenvolupador d'activitats clic).

2.3.2 Breu història

2.3.1.1 Els inicis

Aquest projecte sorgeix del grup d'investigació GESSI de la UPC amb en Marc Alier i la Maria José Casany com a principals responsables.

Però aquest no és el primer intent de traduir el JClic al OLPC, aquest projecte es va iniciar amb el desenvolupament dels projectes de final de carrera de dos estudiants la FIB: en Marc Rodríguez Tristany i l'Aleix Palet, que també van implementar un reproductor de JClic para OLPC tot i que només reproduïa certs trencaclosques i tenia alguns problemes de rendiment amb l'XO.

Posteriorment, i ja a mitjans del passat any, uns quants alumnes de la FIB (en Jose Camallonga, el Marc Benito, en Carlos Castilla i jo) vam decidir unir-nos al projecte per tal de refer l'eina des de zero buscant corregir els problemes de rendiment que tenia l'anterior versió i desenvolupant una eina eficient amb grans expectatives de què aquesta sigui utilitzada pels nens i nenes dels països en desenvolupament.

2.3.1.2 El nom

En el nom de JClic, la "J" prové de Java, la plataforma amb la que està desenvolupat. La nova eina que hem desenvolupat i adaptat l'hem denominat SugarClic ja Sugar és que el nom de la interfície gràfica d'usuari del OLPC.

2.3.1.3 Actualitat

El projecte SugarClic ja té totes les activitats del JClic original implementades i està en fase de proves perquè diferents comunitats d'Amèrica del Sud el provin.

2.3.3 El nucli de desenvolupament

Com ja hem dit, encara que es partia d'una primera versió de l'aplicació, s'ha decidit començar des de nou amb l'objectiu d'aconseguir un millor rendiment a l'aplicació.
A la primera versió es va intentar aprofitar l'estructura de classes que proposava la versió de Java, molt extensa i completa. Això va provocar una gran pèrdua en el rendiment de l'execució. El disseny que es va realitzar per la versió de Java parteix d'uns requisits força diferents ja que aquest no compta amb les restriccions de maquinari que disposa l'OLPC.
Per aquesta segona versió s'ha tingut molt en compte el rendiment i per això s'ha desenvolupat un nucli d'activitats que vol ser eficient fent èmfasi sobretot en que no es podia refrescar la pantalla constantment de tal manera que el sistema quedés saturat. Amb aquest nucli també es va aportar una arquitectura on es poguessin aprofitar i reutilitzar les classes ja creades per al desenvolupament de tots els que intervinguin en el projecte del SugarClic.

2.3.4 Web per Gestionar Clics

El SugarClic també consta d'una funcionalitat que permet obtenir clics des d'una web. Des d'aquesta pàgina es pot accedir a llistat d'activitats ordenades per diferents criteris, així com afegir, esborrar i consultar aquestes activitats. Aquesta web s'ha anomenat Portal-Clic.
Amb el PortalClic s'ha volgut unificar la part de la obtenció de clics des del SugarClic, de tal manera que al professor o a l'alumne que estigui utilitzant-lo li sigui molt més fàcil obtenir nous

clics fent-ho de forma intuïtiva des de la mateixa aplicació. El JClic no permet això, sinó que s'ha d'obrir un navegador per buscar els clics, i aquests estaven no només per la web de la Xtec, sinó que és possible que estiguin dispersats per vàries pàgines o que cada organització tingui els seus clics.

Aquest portal també està dissenyat amb software lliure i pot ser replicat a qualsevol lloc de forma senzilla. D'aquesta manera qualsevol persona que vulgui crear un portal de clics per al propi ús, com per exemple escoles, instituts o organitzacions, pot descarregar-se el software i instal·lar-lo.

També s'ha donat la possibilitat d'afegir varis repositoris d'un mateix clic per si un no estigués funcionant.

3. OBJECTIUS

En aquest apartat explicarem tant els objectius del projecte SugarClic en general, com els objectius relatius a les nostres contribucions en aquest projecte.

El nostre principal objectiu, comú a tots els membres del SugarClic, és mitigar la falta de continguts disponibles per a OLPC adaptant-ne el projecte educatiu JClic, disponible per a Java, ja que de continguts i activitats per a JClic n'hi ha molts.
Personalment, el nostre objectiu principal ha estat una participació tangible en aquest projecte, aportant-hi una sèrie de solucions. Dins d'aquest objectiu principal podem dir que buscàvem complir els següents objectius secundaris propis:

- Contribuir en la realització d'una interfície de visualització d'activitats del JClic i en la comunicació d'aquesta interfície amb l'aplicació que permetrà a l'usuari gestionar totes aquestes activitats.
- Desenvolupar activitats per al SugarClic, més concretament fer l'adaptació de les activitats clic fetes per al JClic següents:
    - Associació Simple
    - Associació Complexa
    - Ordenar Text
    - Resposta escrita
- Solucionar possibles problemes que pugui tenir el projecte del SugarClic i dotar-lo de millores que puguin influir en la seva correcta visualització.

4. ANÀLISI DE REQUISITS

A continuació es defineixen els requisits de la nostra aplicació, que s'encarreguen de descriure les funcionalitats i el comportament que ha de tenir. Podem diferenciar entre dos tipus de requisits:

- **Funcionals:** Defineixen les funcionalitats que el nostre sistema ha de tenir. Són els encarregats de dir-nos què ha de fer el nostre sistema.

- **No funcionals:** S'entén per requisits no funcionals tots aquells requisits del sistema que no es recullen en la definició de la funcionalitat, tot i que alguns d'ells puguin tenir repercussions en el diagrama de casos d'ús o en el de classes. Es descriuen propietats com la fiabilitat, rendiment, temps de resposta o el tipus d'aparença que desitgem que la nostra aplicació tingui.

4.1 REQUISITS FUNCIONALS

Per la realització d'aquest projecte i tenint en compte els objectius marcats s'ha fet un desglossament de les funcionalitats i accions que hauria de dur a terme la nostra part de l'aplicació.

Primerament caldria dir que l'usuari es relaciona amb dos tipus de interfícies clarament diferenciades, encara que ell no ho pugui percebre:

- Una és la que s'encarrega de gestionar totes les pantalles que no són pròpiament de la visualització de les activitats del JClic: la gestió i descàrrega de clics mitjançant la web, l'ajuda, l'intercanvi de clics per USB, etc.

- L'altra, és l'encarregada de tot el que està relacionat directament amb la visualització de les activitats clic. Individualment la part per la qual s'ha treballat ha sigut aquesta última, tot i prèviament hem hagut de definir el protocol d'intercanvi entre una i l'altra.

Pel que fa a la nostra interfície pròpiament, podem definir les següent funcionalitats:

- Hi ha d'haver una interfície que reprodueix els clics, la qual es comunica amb la interfície de gestió de clics amb un protocol determinat acordat entre les dues parts.
- L'usuari ha de poder navegar per les activitats de tal manera que ha de poder passar a la següent, l'anterior, la primera o la última.
- El sistema ha de permetre reiniciar l'activitat que s'està fent quan l'usuari ho desitgi.
- L'usuari ha de poder jugar a les activitats SugarClic implementades per nosaltres:
    - Associació simple
    - Associació complexa
    - Ordenar text
    - Resposta escrita

4.2 REQUISITS NO FUNCIONALS

Els requisits no funcionals tinguts en compte en aquest projecte els hem dividit en cinc subgrups: Interfícies d'usuari i factors humans, Documentació, Consideracions de hardware, Característiques de rendiment i Entorn de desenvolupament.

4.2.1 Interfície d'usuari i factors humans

- El projecte està dissenyat per a usuaris amb edats d'aprenentatge i per tant la interfície ha de ser senzilla, amigable, fàcil d'utilitzar, entenedora, vistosa i llegible.
- S'ha de protegir als usuaris d'errors que deixin el programa sense resposta

4.2.2 Documentació

- L'usuari ha de tenir un manual senzill per poder saber de forma ràpida i senzilla com funciona l'aplicació si en té algun dubte.

- Els professors que supervisen els usuaris també han de tenir un manual més complet per poder aconsellar als alumnes en cas que aquests no sàpiguen que fer en algunes de les activitats.
- Hem de tenir una petita guia o manual d'instal·lació de l'entorn així com de desenvolupament d'activitats per a que propers projectistes vinculats a l'aplicació puguin saber per on començar.

4.2.3 Consideracions de hardware

- El programa ha de ser totalment compatible amb l'OLPC i tots els continguts han d'adaptar-se al seu format de pantalla.

4.2.4 Característiques de rendiment

- El programa ha tenir un temps de resposta considerable, per això hem decidit que no cal que sigui una conversió exacta del JClic de Java, sinó que provarem les funcionalitats i les adaptarem el més eficientment possible per disminuir aquest temps de resposta.

4.2.5 Entorn de desenvolupament

- El codi es desenvoluparà amb Python i els mòduls d'aquest llenguatge que ja inclou l'OLPC, ja que són els entorns amb que està desenvolupat el Sugar i són compatibles totalment amb aquest entorn.
- No es farà servir cap altra llibreria addicional que requereixi d'una instal·lació a part perquè els usuaris no hagin de instal·lar res més que la nostra aplicació per començar a utilitzar-la.

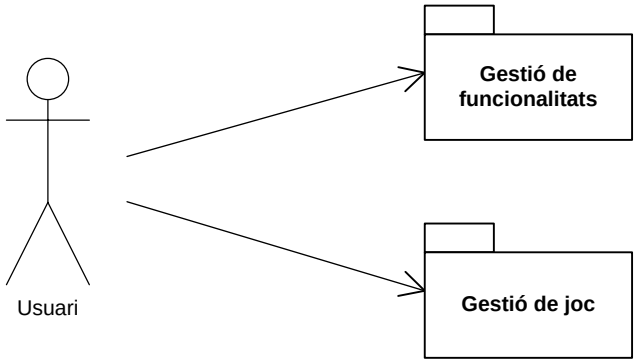
5. ESPECIFICACIÓ

Una vegada realitzat l'anàlisi dels requisits s'ha de definir detalladament el funcionament d'aquestes operacions. A continuació definirem més detalladament els possibles casos d'ús.

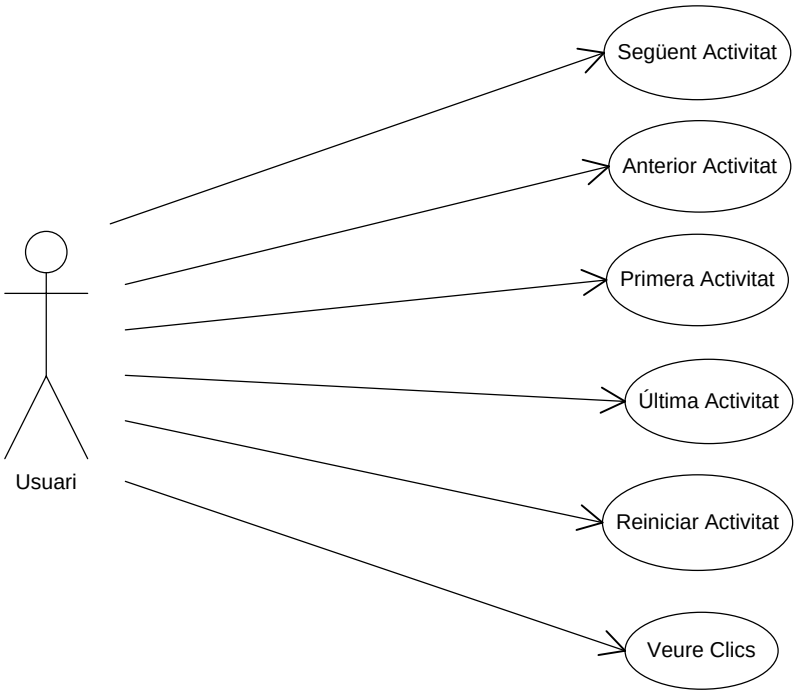
5.1 DIAGRAMA DE CASOS D'ÚS

En el nostre sistema, el reproductor de clics, podem trobar només un tipus d'actor: *l'usuari*, que representa el rol de la persona que interactuarà amb els jocs i accedirà les funcionalitats del SugarClic.

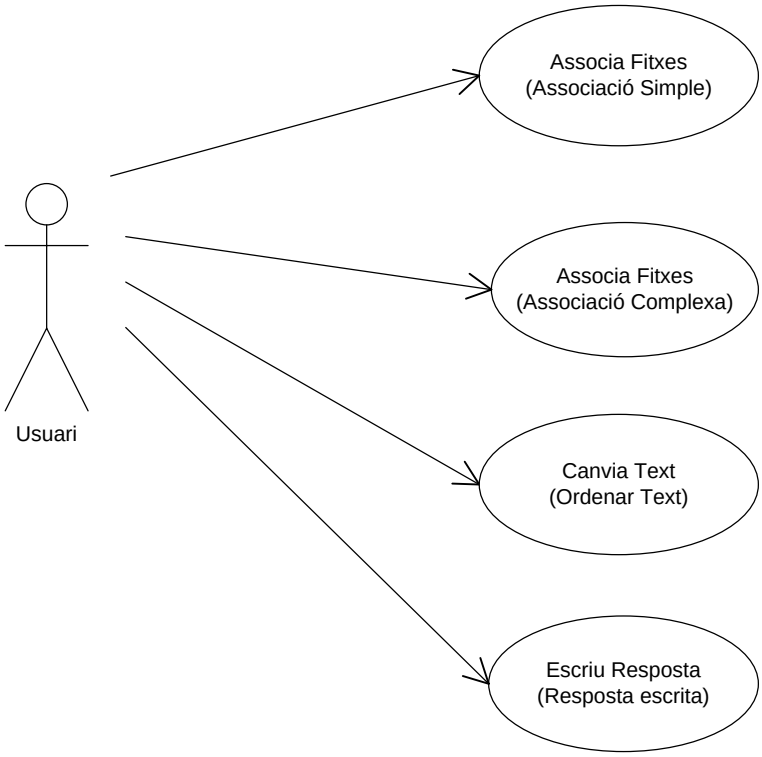
Per tal de facilitar la lectura i comprensió dels casos d'ús, els hem dividit en dos subgrups:
Gestió d'opcions: Conté els casos d'ús que relacionats amb la interacció de l'usuari directament amb les funcionalitats de la nostra interfície com la navegació per les diferents activitats o clics.
Gestió de joc: Conté els casos d'us relacionats directament amb la resolució de cadascuna de les quatre activitats desenvolupades.



Gestió d'opcions



Gestió de joc



5.2 DESCRIPCIÓ DELS CASOS D'ÚS

5.2.1 Activitat següent

Descripció L'usuari desitja veure la següent activitat Clic.

Actors: Usuari

Precondicions:

- Es mostra una activitat en pantalla
- No estem a la última activitat

Curs típic d'esdeveniments:

1. L'usuari pressiona el botó següent.
2. El sistema mostra la següent activitat disponible.

Possibles errors:

Si trobem un error en l'activitat següent s'ha de saltar a la següent d'aquesta.

5.2.2 Activitat Anterior

Descripció: Permet a l'usuari jugar amb l'activitat anterior.

Actors: Usuari

Precondicions:

- Es mostra una activitat en pantalla

Curs típic d'esdeveniments:

1. L'usuari pressiona el botó anterior.
2. El sistema mostra l'activitat anterior de la seqüència per pantalla

Possibles errors:

Si trobem un error en l'activitat anterior s'ha de saltar a l'anterior d'aquesta.

5.2.3 Primera activitat

Descripció: L'usuari desitja veure la primera activitat.

Actors: Usuari

Precondicions:

- Es mostra una activitat en pantalla

Curs típic d'esdeveniments:

1. L'usuari pressiona el botó primera activitat.
2. El sistema mostra la primera activitat de la seqüència

Possibles errors:

Si trobem un error a la primera activitat s'ha de saltar a la següent d'aquesta.

5.2.4 Última activitat

Descripció: L'usuari desitja veure la última activitat.

Actors: Usuari

Precondicions:

- Es mostra una activitat en pantalla

Curs típic d'esdeveniments:

1. L'usuari pressiona el botó primera activitat.

2. El sistema mostra la última activitat de la seqüència

Possibles errors:

Si trobem un error a la última activitat s'ha de saltar a l'anterior d'aquesta.

5.2.5 Reiniciar activitat

Descripció: L'usuari al botó d'activitat següent

Actors: Usuari

Precondicions:

- Es mostra una activitat en pantalla

Curs típic d'esdeveniments:

1. L'usuari pressiona el botó reiniciar.
2. El sistema tornar a mostrar l'activitat en la que estem reiniciant-la.

5.2.6 Mostrar clics

Descripció: L'usuari vol veure els clics disponibles

Actors: Usuari

Precondicions:

- Es mostra una activitat en pantalla

Curs típic d'esdeveniments:

1. L'usuari pressiona el botó de "Mis clics".
2. El sistema mostrarà els clics disponibles.

5.2.7 Associa fitxes (Associació simple)

Descripció: L'usuari vol associar les fitxes de l'Associació simple

Actor: Usuari

Precondicions:

Es mostra una activitat d'*Associació simple* en pantalla

No hi ha cap cel·la seleccionada.

Curs típic d'esdeveniments:

1. El sistema espera que l'usuari interaccioni amb ell.
2. L'usuari clica un cel·la
3. El sistema comprova si la cel·la clicada és vàlida.
4. El sistema mostra la cel·la clicada emmarcada.
5. L'usuari clica sobre una altra cel·la.
6. El sistema desmarca la cel·la anterior.
7. El sistema comprova que la última cel·la clicada sigui vàlida
8. El sistema comprova que les cel·les clicades estiguin relacionades
9. El sistema canvia el contingut de les dues cel·les clicades.

Curs alternatiu:

3a. Si la cel·la clicada conté un contingut no vàlid torna a l'1.

7a. Si la cel·la clicada conté un contingut no vàlid torna a l'1.

8a. Si el contingut de les cel·les no està relacionat es torna a l'1

5.2.8 Associa fitxes (Associació complexa)

Descripció: L'usuari vol associar les fitxes de l'Associació complexa

Actor: Usuari

Precondicions:

Es mostra una activitat d'*Associació complexa* en pantalla

No hi ha cap cel·la seleccionada

Curs típic d'esdeveniments:

1. El sistema espera que l'usuari interaccioni amb ell.
2. L'usuari clica un cel·la
3. El sistema comprova si la cel·la clicada és vàlida.
4. El sistema mostra la cel·la clicada emmarcada.
5. L'usuari clica sobre una altra cel·la.
6. El sistema desmarca la cel·la anterior.
7. El sistema comprova que la última cel·la clicada sigui vàlida
8. El sistema comprova que les cel·les clicades estiguin relacionades
9. El sistema canvia el contingut d'una de les cel·les clicades, la que correspongui.

Curs alternatiu:

3a. Si la cel·la clicada conté un contingut no vàlid torna a l'1.

7a. Si la cel·la clicada conté un contingut no vàlid torna a l'1.

8a. Si el contingut de les cel·les no està relacionat es torna a l'1

5.2.9 Canvia text (Ordenar Text)

Descripció: L'usuari vol canviar text a l'activitat Ordenar Text

Actor: Usuari

Precondicions:

Es mostra una activitat d'*Ordena Text* en pantalla

No hi ha cap text seleccionat

Curs típic d'esdeveniments:

1. El sistema espera que l'usuari interaccioni amb ell.
2. L'usuari clica una part del text
3. El sistema comprova si la part de text clicat és vàlid.
4. El sistema mostra la part de text clicat d'un altre color.
5. L'usuari clica sobre una altra part de text.
6. El sistema desmarca la part de text anterior.
7. El sistema comprova que la última cel·la clicada sigui vàlida
8. El sistema comprova que les cel·les clicades estiguin relacionades
9. El sistema canvia el contingut d'una de les cel·les clicades, la que correspongui.

Curs alternatiu:

3a. Si l'usuari no clica sobre una part de text vàlida tornem a l'1

5.2.10 Escriu resposta (Resposta escrita)

Descripció: L'usuari vol relacionar una cel·la amb un text escrit

Actor: Usuari

Precondicions:

Es mostra una activitat de *Resposta Escrita* en pantalla

Curs típic d'esdeveniments:

1. El sistema espera que l'usuari interaccioni amb ell.
2. L'usuari clica un cel·la
3. El sistema comprova si la cel·la clicada és vàlida.
4. El sistema mostra la cel·la clicada emmarcada.
5. L'usuari escriu el text relacionat al quadre de text i prem Enter.
6. El sistema esborra el contingut del quadre de text.
7. El sistema comprova que el text es relacioni amb la cel·la clicada..
8. El sistema canvia el contingut de la cel·la clicada.

Curs alternatiu:

1a. L'usuari escull la cel·la seleccionada per defecte

    1a1. Es passa al punt 5.

3a. Si l'usuari no clica sobre cel·la vàlida es passarà al punt 1.

6a. Si no es correspon el text amb la cel·la clicada es torna al punt 5.

6. Tecnologies utilitzades

A continuació, en aquest apartat, detallarem les tecnologies utilitzades, és a dir, el programari emprat per al desenvolupament dels nostres objectius.

6. 1 Sugar

El Sugar és l'entorn gràfic del Sistema Operatiu dissenyat per l'XO. Un disseny força diferent als sistemes operatius que estem acostumats a veure avui en dia, creat especialment perquè els nens puguin aprendre divertint-se.

Aquest sistema operatiu és sobre el que hem estat treballant per desenvolupar la nostra aplicació i tot i que hem tingut un OLPC per fer les proves, també hem instal·lat una Màquina Virtual per comoditat de no haver d'endur-nos-el amunt i avall per fer les proves pertinents.

Figura 3: Sugar

6.2 VMWare

El VMWare és un sistema de virtualització per programari. Un sistema virtual per programari és un programa que simula un sistema físic (un ordinador) amb unes característiques i un hardware determinat. Quan s'executa el programa (simulador), proporciona un ambient d'execució similar a tots els efectes d'un ordinador físic (excepte en el pur accés físic al maquinari simulat), amb CPU

(pot ser més d'una), BIOS, targeta gràfica, memòria RAM, targeta de xarxa, sistema de so, connexió USB, disc dur (poden ser més d'un), etc...

Per agilitzar les proves de l'aplicació en un entorn OLPC o en el cas de no tenir un portàtil XO s'ha utilitzat en VMWare un sistema que l'emula de tal manera que podem executar aquest entorn en el nostre ordinador. Un virtualitzador de software com el VMWare o qualsevol altre permet executar o simular varis sistemes operatius dins d'un mateix hardware de forma simultània i així aprofitar més els recursos.

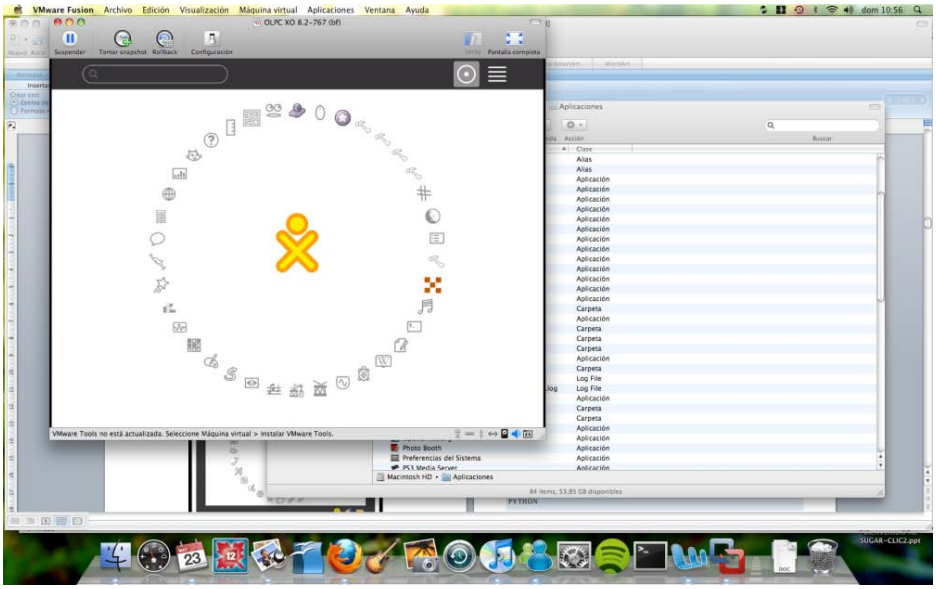


Figura 4: Sistema operatiu Sugar executant-se al VMWare

Tot i la comoditat que ens dóna el VMWare per fer proves cal dir que, com que l'OLPC depèn d'un sistema força limitat, també ha sigut necessari fer les proves directament sense virtualitzar-lo ja que simulant-lo en el nostre ordinador les respostes de les aplicacions solen ser més ràpides. Així doncs, ha sigut necessari el testeig directe al XO per comprovar el funcionament adequat de l'aplicació en aquest.

6.3 Ubuntu

Ubuntu és una distribució del sistema operatiu GNU/Linux basada en Debian que proporciona a l'usuari gran facilitat d'ús i d'instal·lació del sistema.

A més, ofereix la garantia que cada 6 mesos sortirà una nova versió del sistema. Els guanys econòmics sols provenen del suport tècnic, el que la diferencia de la resta de les altres distribucions comercials.

Figura 5: Sistema operatiu Ubuntu

6.4 Python

Python és un llenguatge de programació d'alt nivell de propòsit general que pot ser utilitzat per a tota classe de software. Va ser creat per Guido van Rossum, un científic computacional que actualment treballa per l'empresa Google. Com a anècdota podem dir que el nom de Python prové de la sèrie còmica britànica Monty Python's Flying Circus (*El Circ Ambulant de Monty Python*).

Molt sovint Python és comparat amb altres llenguatges com el Perl, el Java o el Ruby. Com a principal característica el Python té una gran quantitat de mòduls disponibles que proporcionen crides al sistema, lectura i escriptura de fitxers, etc. a més de suport per a l' integració amb altres llenguatges i eines.

Python és un llenguatge de programació multiparadigma, és a dir, que permet crear programes utilitzant més d'un estil de programació: orientat a objectes, programació estructurada i programació funcional.

A diferència dels altres llenguatges de programació, Python utilitza sagnats com a delimitadors de blocs, de tal manera que obliga al programador a crear un codi molt més clar i llegible, com es pot veure al següent exemple, que és una funció que ens diu si el paràmetre d'entrada és parell o senar:

```
def es_parell(numero):
    if (numero % 2 == 0) :
        print "És parell"
        return 0
    else:
        print "És senar"
        return 1
```

Python és la base del Sugar i és totalment compatible amb l'OLPC, és per això hem triat aquest llenguatge de programació, perquè sabíem que no ens donaria problemes.

6.5 Pygame

Pygame és una llibreria de Python dissenyada especialment per la creació de videojocs. Disposa d'un conjunt de mòduls i que tracten tant gràfics, creació de figures de vàries formes, esdeveniments de tot tipus (com de teclat o ratolí) i disposa d'una llibreria especial per a sons.

És capaç d'executar els jocs a 30 fotogrames per segon, o sigui, que no és molt potent, però és molt útil per realitzar jocs amb dues dimensions i de poca envergadura.

S'adequa perfectament a les nostres necessitats ja que el que volem realitzar és una eina que consumeixi pocs recursos per l'entorn en què el desenvolupem (l'OLPC).

Consta de diferents mòduls per tractar diferents apartats. Per exemple el mòdul *Display*, que ens permet controlar la visualització de la finestra o l'*Event*, que ens permet tractar els diferents esdeveniments que l'usuari introdueix a l'aplicació.

6.6 PyGTK

El PyGTK és un mòdul de Python que permet dissenyar interfícies gràfiques d'usuari atractives i sofisticades sense tracta amb els detalls del baix nivell com el pintat o la interacció amb el dispositiu.

Tot i que no hem programat directament amb PyGTK, ha sigut una eina que s'ha tingut en compte per realitzar la interfície amb què és comunica l'usuari i ha influït directament en el projecte SugarClic.

6.7 XML

XML, de l'anglès eXtensible Markup Language (traduït com llenguatge de marques extensible), és un metallenguatge extensible, d'etiquetes, desenvolupat pel World Wide Web Consortium (W3C). És una simplificació i adaptació de l'experimentat SGML, i permet definir la gramàtica de llenguatges específics (de la mateixa manera que HTML és, alhora, un llenguatge definit per SGML).

Per tant, XML no és realment un llenguatge en particular, sinó una manera de definir llenguatges per a diferents necessitats. Alguns dels llenguatges que empren XML per a la seva definició són XHTML, SVG o MathML. XML no ha nascut només per a la seva aplicació a Internet, sinó que es proposa com a un estàndard per a l'intercanvi d'informació estructurada entre diferents plataformes. Es pot utilitzar per a bases de dades, editors de text, fulls de càlcul i per moltes altres aplicacions diverses. XML és una tecnologia relativament senzilla que té al seu voltant altres que la complementen i la fan notablement més extensa, a més de proporcionar-li unes possibilitats molt més grans. A l'actualitat té un paper molt important, ja que permet la compatibilitat entre sistemes, permetent de compartir informació d'una manera segura, fiable i fàcil.

Els arxius amb extensió .jclic no són res més que una forma de XML. De manera que la informació que contenen aquests arxius és traduïda o *parsejada* tant des del JClic original com des del nostre projecte, cosa que ens permet generar activitats a partir d'un codi on hi ha unes etiquetes que ens diran com s'ha de comportar la nostra aplicació.

6.8 Minidom

El minidom és un mòdul del Python que ens permet llegir i parsejar l'XML i navegar per ell com si fos un arbre, navegant pels seus fills. Està basada en el DOM i hi ha altres llibreries que fan la mateixa funció que aquesta per a Python, però hem optat per aquesta ja que l'hem considerat la més òptima per a nosaltres per la seva lleugeresa i simplicitat.

6.9 NetBeans

Netbeans és un entorn de desenvolupament integrat (IDE) per al desenvolupament amb Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, etc. pensada per escriure, compilar, depurar i executar programes.

L'IDE NetBeans està escrit en Java i funciona a tot arreu on s'ha instal.lat un JDK. Un JDK és un software necessari per al desenvolupament de Java, però no és necessari per al desenvolupament en altres llenguatges de programació.

La plataforma NetBeans permet que les aplicacions que es va desenvolupar a partir d'un conjunt de components de programari anomenats mòduls. Les aplicacions basades en la plataforma NetBeans (inclòs el IDE NetBeans) pot estendre's per part de desenvolupadors.

NetBeans permet crear aplicacions amb Python ja que té un motor per escriure (ressaltant la sintaxi), identificar errors i el debugger. Sens dubte, NetBeans s'ha convertit en un IDE apte per a la majoria dels llenguatges de programació de codi obert moderns.

6.10 GoogleCode

Google Code és la pàgina web de Google per a desenvolupadors. Disposa d'eines de desenvolupament, APIs i recursos tècnics.

La eina que hem fet servir de Google Code és la d'allotjament o hosting de projectes. Aquesta eina ens ha permès penjar el nostre codi de forma oberta de tal manera que els usuaris interessats el poden consultar i ens ha proporcionat un control de versions com Mercurial, què és el que hem triat nosaltres (explicat a continuació), o Subversion de forma que només hem pogut modificar el projecte els desenvolupadors que formem part del projecte SugarClic.

L'allotjament de projectes del Google Code conté, a part de fer un seguiment del control de versions, una wiki per la documentació del projecte, un espai per poder descarregar els fitxers relacionats amb el projecte i una pàgina per poder fer el seguiment de les incidències que van sorgint, eina important del projecte.

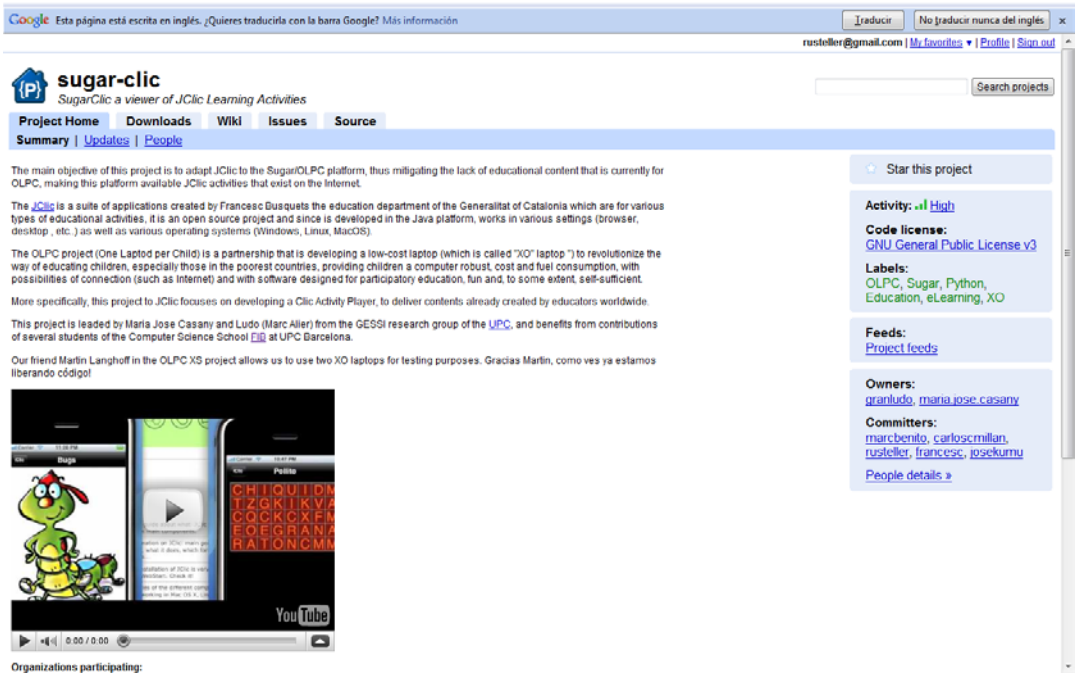


Figura 6: Pàgina principal del Google Code del SugarClic

Cal que fem especial èmfasi en aquesta eina de seguiment de les incidències, que ens ha permès interactuar entre els desenvolupadors del projecte de manera senzilla.

6.11 Mercurial

Mercurial és un sistema de control de versions per desenvolupadors de software que es caracteritza per la seva rapidesa, per tenir integrat també una interfície web, pel seu funcionament completament distribuït i per la gestió avançada de branques i de les fusions. Aquest sistema de control està dins de Google Code i de Netbeans, és pera això que s'ha considerat la millor opció, tot i que el Subversion també està integrat dins d'aquest dos sistemes.

Per a accedir al repositori del projecte a Google Code s'utilitza un protocol eficient, basat en HTTP, que redueix el tamany de les dades transferides i la generació de peticions i connexions noves.

La característica que el diferencia del Subversion és que té la possibilitat de realitzar commits abans de realitzar l'actualització de tal manera que s'evita fer múltiples peticions un mateix dia.

6.12 JClic Author

Aquesta eina és una de les aplicacions que consta el JClic. La seva principal funcionalitat és que permet crear i modificar projectes JClic, en un entorn visual molt intuïtiu i immediat.

És la eina que hem utilitzat per a crear els clics que serveixen com a manual d'usuari i com a about.

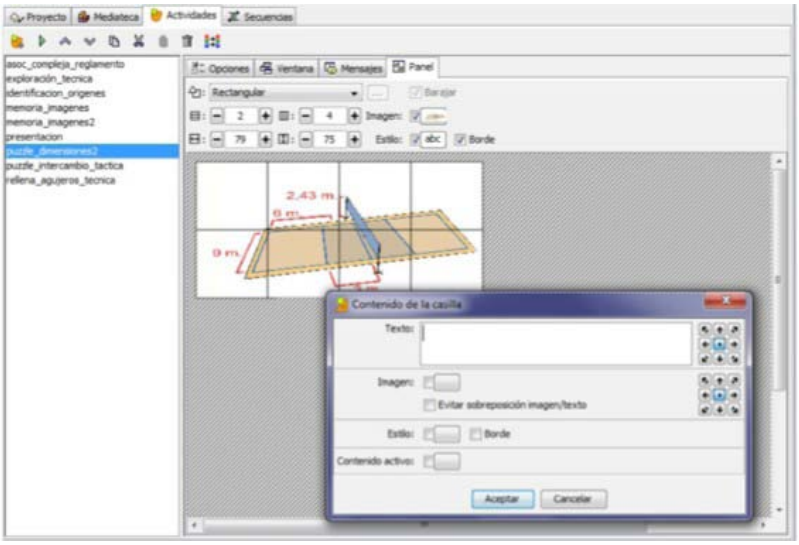


Figura 7: JClic Author

7. Disseny i implementació de les solucions

En aquest apartat detallarem el disseny i la implementació de cada una de les nostres aportacions en el projecte marcades com a objectius a l'inici d'aquest.

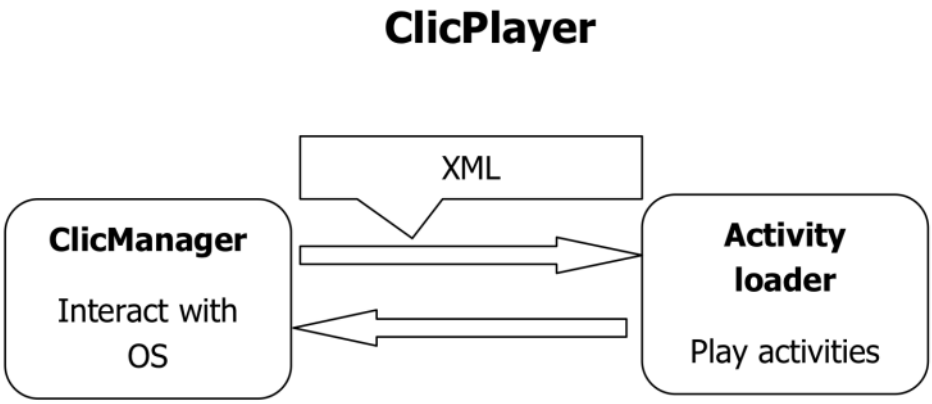
7.1 Reproductor de clics i comunicació amb l'aplicació

En aquest apartat s'explicarà com vam començar a desenvolupar el reproductor de clics, passant primer per realitzar un protocol d'intercanvi d'informació entre dos llenguatges de programació diferents.

Al iniciar el projecte, es va decidir que cadascú es faria càrrec de diferents parts del SugarClic optimitzant el treball fet per cadascú especialitzant-nos en una part. D'aquesta manera en Jose Camallonga es faria càrrec de la part de desenvolupament de l'aplicació externa que gestionaria els clics, mentre que nosaltres desenvoluparíem el reproductor de clics.

7.1.1 Disseny

A partir de la idea anterior es va dissenyar la següent solució:



ClicManager: És la interfície desenvolupada en PyGTK. La que s'encarrega de gestionar totes les pantalles que no són pròpiament de la visualització de les activitats del JClic: la gestió i descàrrega de clics mitjançant la web, l'ajuda, l'intercanvi de clics per USB, etc.

ActivityLoader (Reproductor de clics): És la part encarregada de tot el que està relacionat directament amb la visualització de les activitats Clic.

7.1.2 Implementació

Primerament i com a procés d'aprenentatge en el món de l'OLPC, es va crear una aplicació exclusivament en Pygame consistent en un trencaclosques amb diverses imatges.

Paral·lelament un segon projectista del SugarClic, en Jose Camallonga, un dels altres projectistes, estava experimentant també amb l'OLPC, però fent èmfasi en el llenguatge de programació PyGTK.

A partir d'aquí els dos ens vam unir per realitzar l' estructura de comunicació entre les dues parts: la part de pygame i la part de pygtk. Per això vam haver de posar-nos d'acord per realitzar un protocol que intercanviés la informació d'una part a l'altra .

Cal fer esment a que Pygame i PyGTK no són dos mòduls creats per una compatibilitat entre ells, ja que els dos tenen un gestor de esdeveniments d'entrada diferents (teclat, mouse, etc.) i a més, els dos utilitzen un bucle principal executat contínuament des d'una funció principal.

El problema de la traducció dels esdeveniments es va solucionar gràcies a un mòdul del python: OLPCGames, ja realitzat per la finalitat d'executar jocs dissenyat en Pygame dins de Sugar i que tradueix els esdeveniments de teclat i ratolí de PyGTK a Pygame.

El punt dels dos bucles principals es va solucionar de la següent manera:

Primerament vam idear només que la classes de pygtk, des del seu bucle principal, feien crides a la funció que actualitzava el codi de pygame. També, i després de cridar aquesta funció, cridava a una altra funció que li feia saber a la interfície de PyGTK quan l'usuari havia clicat a les fletxes d'activitat següent o anterior, i quan l'usuari havia comés un encert o un error en el trencaclosques.

El trencaclosques, per la seva part, mostrava per pantalla una part de l'XML d'un .jclic qualsevol, concretament el títol de l'activitat a la qual estàvem i que variava mitjançant els botons de anterior i següent. També agafava un codi XML contingut al .jclic que ens definia la imatge i les files i columnes que tindria el trencaclosques.

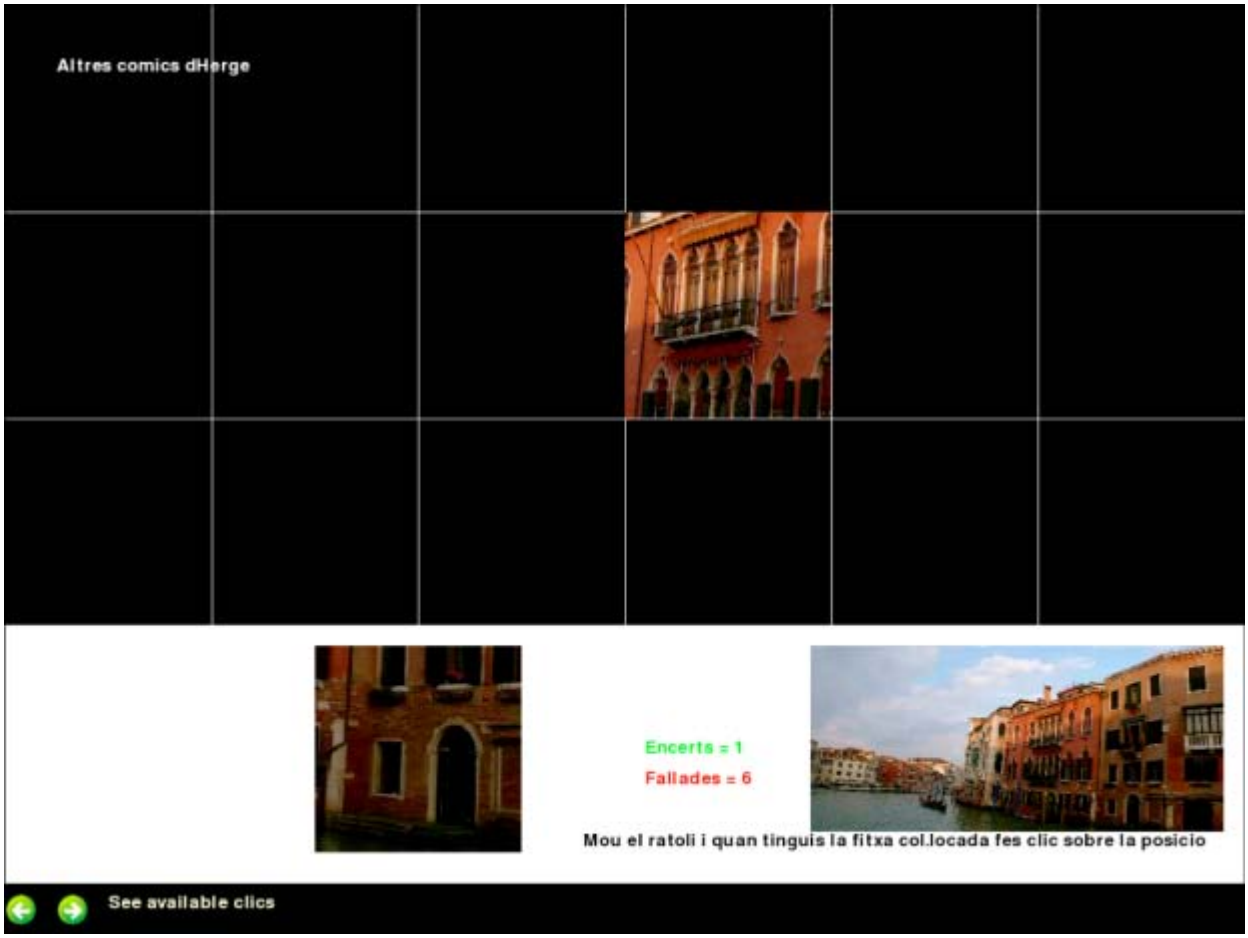


Figura 8: Activitat de prova i primera interacció entre pygame i pygtk

Això es feia amb mitjançant la classe *Controller.py,* que era el Controlador, és a dir, una classe intermitja que es comunicava amb totes les classes del PyGTK i també amb la classe sequencer. Les classes que formaven part d'aquesta arquitectura eren:

- **sequencer.py:** inicia el reproductor de clics, passa l'XML de cada una de les activitats i tracta les peticions del reproductor: canvis de pantalla, retorn a l'aplicació, etc.
- **clic_activities_handler.py**: Classe pygame que es crida indirectament des del pygtk. Aquí és on es tracta tot el relatiu a l'xml de cara al pygame, així com els esdeveniments de clics en les fletxes de següent i anterior. Aquesta funció també cridava a l'activitat de prova puzzleU.py
- **puzzleU.py**: activitat de prova de trencaclosques. Estava preparada per rebre per paràmetres la ruta concreta de la imatge, el nombre de columnes i el nombre files. Degut al posterior pas a l'arquitectura d'Activitats aportada per en Marc Benito (comentada a l'apartat següent), aquesta classe va quedar obsoleta.

Tot seguit veiem la comunicació entre el Controller i el *Reproductor de clics* mitjançant el *Sequencer*.

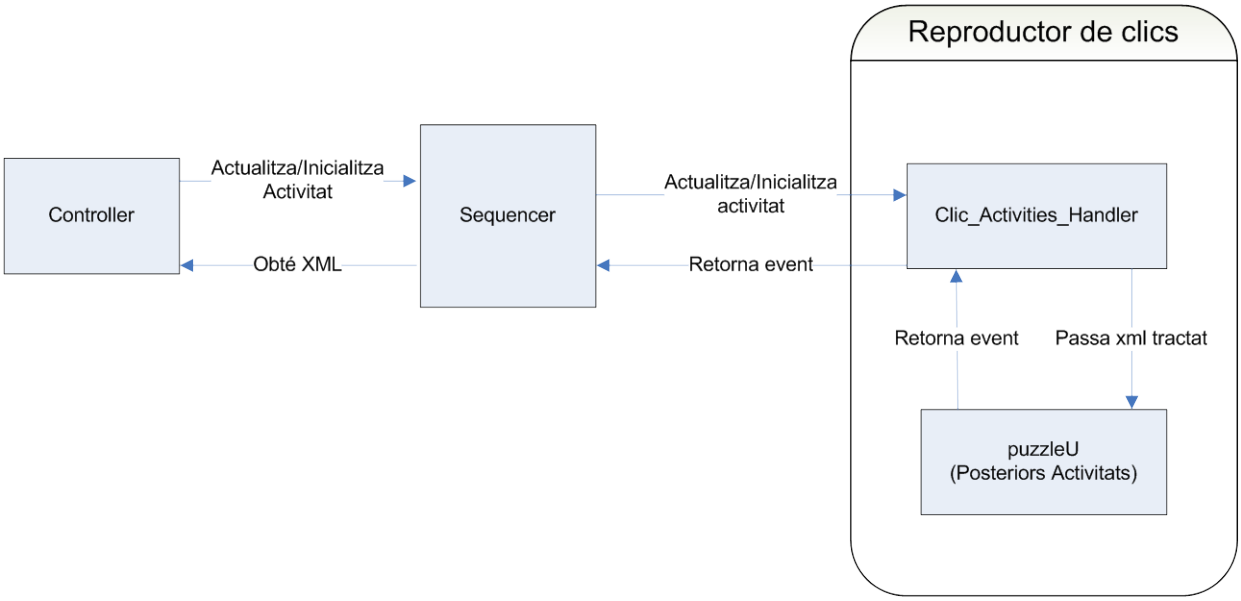


Figura 9: Comportament de les classes del Gestor amb el Reproductor de clics

També hi havia altres classes com és el cas del ws_handler.py o db_clics.py, però són classes que eren implementades exclusivament per l'altre projectista, ja que no tenien res a veure amb la nostra part del projecte.

D'aquesta manera vam començar a desenvolupar el *Reproductor de clics* i a comunicar-lo amb la interfície encarregada de gestionar els clics: el ClicManager.

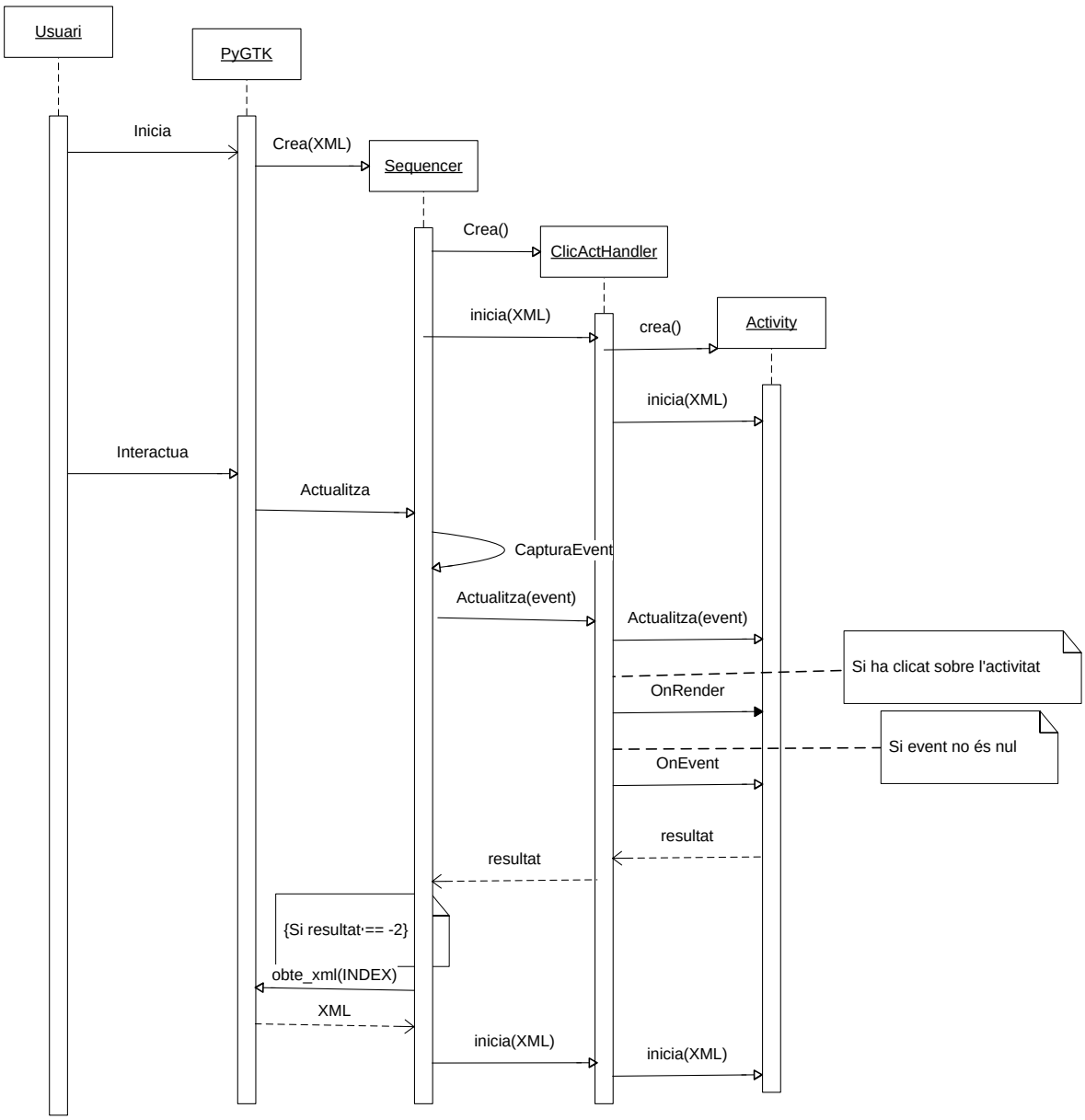
Com hem dit abans la interfície que havíem dissenyat nosaltres només arriba fins a la comunicació amb el ClicManager i la prova del tractament de l'XML de l'activitat actual que ens passa per paràmetre aquesta classe. Posteriorment, amb la incorporació d'en Marc Benito i l'aportació de la seva arquitectura d'activitats clic, ell mateix va adaptar aquesta arquitectura a la interfície feta per nosaltres.

Finalment podem dir que aquesta arquitectura es va anar perfilant i va anar evolucionant fins a tenir el que ja tenim ara: L'aplicació principal en pygtk crida la classe Sequencer, que és l'encarregada d'executar ordenadament la seqüència d'activitats parsejant l'xml de la seqüència i cridant a cada una de les activitats quan pertoqui.

La classe Sequencer crida a la classe Clic_Activities_handler, que és l'encarregada de controlar la comunicació entre el que va del pygame al pygtk i viceversa. D'aquesta manera aquesta classe és l'encarregada de iniciar activitats i també de captar si l'usuari ha apretat alguna cosa que faci que haguem de canviar a la següent activitat.

Finalment, l'última crida és la crida des del Clic_Activities_Handler a la classe Activity, superclasse de totes les subclasses relacionades amb cada un dels tipus d'activitats.

Tot seguit es mostra un diagrama de seqüència on s'inicia el reproductor d'activitats i l'usuari interactua sobre el reproductor. Cal dir que hem posat PyGTK com a nom d'una classe que en realitat és tota la interfície que gestiona aquest, del qual nosaltres no hem implementat res.



Com a exemple hem posat que es comprova si es prem sobre el botó següent, tot i que hi ha tantes comprovacions com opcions hi ha al menú del SugarClic:

- Navegació per les activitats (següent, anterior, última i primera), on també es té en compte si són el manual o l'about, casos en els quals s'haurà de tornar a la pantalla del ClicManager després de visualitzar la última activitat.
- Veure el clics
- Reiniciar activitat
- Salts en la seqüència a partir d'una cel·la

En el diagrama, la variable INDEX que és passada des del PyGTK cap al Sequencer no és res més que l'índex de la seqüència d'activitats que es vol executar.

7.2 Desenvolupament d'Activitats Clic

En aquest apartat detallarem el desenvolupament de les activitats que hem realitzat. Però per això creiem necessari primer descriure l'arquitectura que hem seguit per després procedir al disseny i implementació d'aquestes activitats.

Posteriorment mostrarem el disseny de les activitats adaptat a aquesta arquitectura i al final explicarem la implementació de cada una, passant prèviament pel comportament que tenen davant l'usuari a l'hora de la interacció en el qual ens hem fixat per desenvolupar-la.

7.2.1 Arquitectura

Amb la incorporació d'en Marc Benito al SugarClic la forma de desenvolupar activitats es va enfocar al framework de desenvolupament que havia creat. Tot seguit explicarem una mica aquest framework al qual ens hem hagut d'adaptar per programar així les nostres activitats.

Aquest framework no es basa en la traducció literal de les classes existents pel JClic de Java, ja que la quantitat de classes que java genera i l'estil de programació orientat a aplicació d'escriptori (SWING) que presentava la versió de JClic va fer que es considerés inviable poder-ho traduir amb un rendiment acceptable.

Per això en comptes d'una traducció literal el que es va fer és fixar-se en què es té en compte i què és visible en cada un dels tipus d'activitats. Per això el que cal és mirar l'XML de cada activitat JClic i el comportament que té de cara a la seva resolució per part de l'usuari.

D'aquesta manera i veient que hi havia molts elements comuns en aquestes activitats, com podrien ser que la majoria consten de una o vàries graelles, que tenen cel·les a partir d' imatges o a partir de text. Això ho podem veure a les següent imatges:

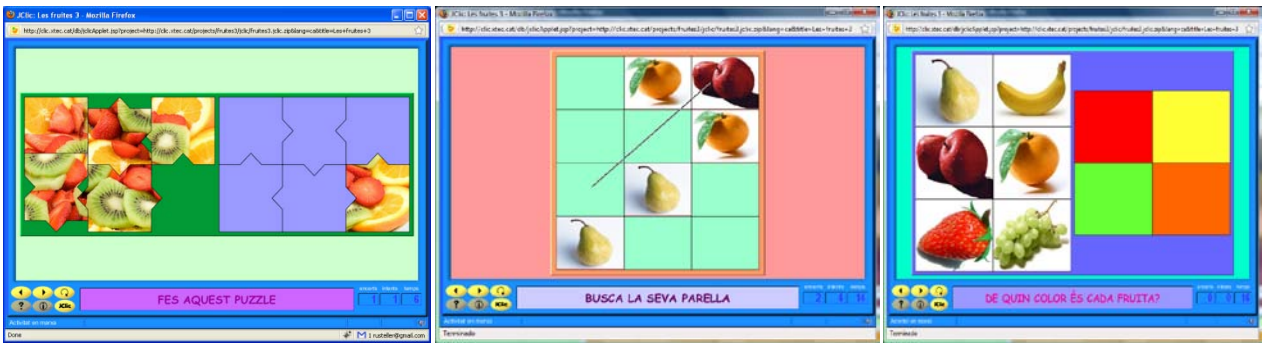


Figura 10: Activitats basades en l'arquitectura de graelles d'imatges

A continuació podem veure el diagrama de classes de l'arquitectura d'Activitats Clic:

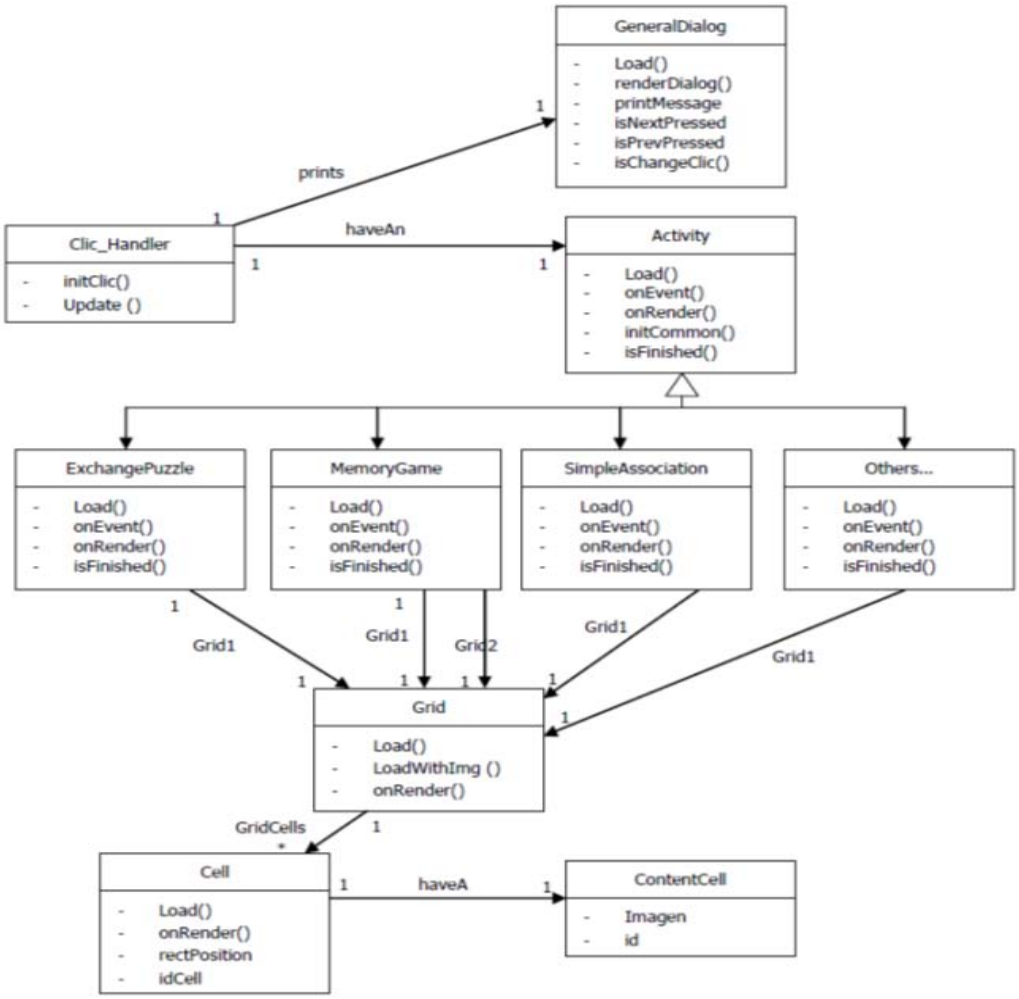


Figura 11: Diagrama de classes de l'arquitectura

El **ClicHandler** és la classe que ens uneix amb el ClicManger. Aquesta rep l'ordre d'actualitzar-se cada uns milisegons i si ha succeït algun esdeveniment, aquest és processat i realitza la acció pertinent.

ClicHandler es comunica amb el GeneralDialog i amb Activity.

El **GeneralDialog** és la classe encarregada de mostrar les parts visuals comunes i fixes en totes les activitats: els botons, els marcs, els quadres de missatges, etc.

Pel que fa a l'arquitectura desenvolupada sobre la classe **Activity:**

Una de les principals característiques d'aquesta arquitectura és la inclusió d'un patró factoria en la superclasse Activity, una classe abstracta que permet que totes les seves subclasses (que són els tipus d'activitat) siguin cridades pel model independentment del tipus de classe o activitat.

Com podem veure, Activity consta de vàries funcions definides que són les que implementen les seves subclasses:

- **Init():** Aquesta classe és una classe que no s'hereta, la tenen per defecte totes les classes del Python i en el nostre cas s'utilitza per inicialitzar els atributs comuns de l'activitat.
- **Load():** En aquesta classe es defineixen tots els paràmetres necessaris per mostrar tot el contingut de l'activitat. Les cel·les, les imatges, els sons, etc.
- **OnEvent():** En aquesta classe es tracten els esdeveniments, tant de teclat com de ratolí, de tal manera que es té en compte la intervenció de l'usuari en l'activitat.
- **isFinished():** Aquesta funció ens determinarà quan una activitat s'ha finalitzat. D'aquesta manera mostrarem els missatges d'acabament oportuns.
- **OnRender():** En aquesta funció és on s'actualitzen o es repinten els elements visuals que formen part de la pantalla en l'activitat.

Com hem dit abans, aquestes activitats segueixen totes el mateix model visual: tenen una o més graelles, que són matrius de cel·les que contenen imatges o text, en els quals l'usuari interactua dins d'aquests graelles. Per exemple el trencaclosques de forat (Hole Puzzle) és una graella on es fa desaparèixer una peça i es barregen les restants; o el joc de Memòria (Memory) on hi ha una graella amb vàries cel·les ocultes on l'usuari ha de destapar les cel·les iguals.
Així doncs, s'ha seguit aquest patró de les graelles de la següent manera:

- **Subclasse activity:** la subclasse d'Activity corresponent al tipus d'activitat i aquesta conté un o més graelles.
- **Grid**: La classe grid és la graella en sí. Aquesta classe construeix una matriu per guardar de forma ordenada tot el contingut que hi apareix.
- **Cells**: Cada una de les cel·les que conté la graella. Dins de cada cell tenim un ContentCell, referent al seu contingut
- **ContentCell**: conté informació relativa a la cel·la: la imatge associada i l'id. L'id és l'identificador que ens ajudarà a identificar la cel·la dins d'un grup perquè a l'hora de resoldre l'activitat puguem saber quan una peça o cel·la ja està al lloc que li pertoca.

Es pot dir que les activitats amb aquesta arquitectura es comporten de la següent manera:

1. L'usuari clica sobre una cel·la.
2. A la funció OnEvent es comprova quina és la cel·la que ha clicat l'usuari i tracta l'esdeveniment. Per exemple, fa constar que la cel·la clicada ha de ser seleccionada, o que ha format part de una solució i per tant s'ha de mostrar una altra imatge de cel·la on hi havia la cel·la.
3. Si la cel·la clicada tenia alguna acció a fer, això es veurà mostrat per la funció OnRender, que repintarà la pantalla tenint en compte els canvis que s'ha fet a la funció OnEvent.
4. Finalment, la funció IsFinished comprovarà si l'activitat ja ha estat solucionada, i si és així ho farà constar al ClicHandler de tal manera que farà les funcions pertinents, com per exemple fer que el missatge que es mostra a la part inferior del SugarClic canviï.

7.2.2 Arquitectura d'Activitats de text

Les activitats de text no funcionen igual que les formades per graelles. Per tant, per fer-les hem necessitat una altra arquitectura. Seguint el model anterior s'ha intentat fer una altra arquitectura, útil només per a les activitats de text, que imiti a la classe Grid però fent una espècie de graella de texts. Aquesta arquitectura l'ha realitzat el Carlos Castilla i és similar a la realitzada per en Marc Benito, és a dir, una classe TextGrid que contingui el que fa referència a certes activitats de text on s'ha pogut aprofitar que el codi és similar, com per exemple, diferenciar els textos que són informatius dels que són per resoldre, etc.

Aquesta arquitectura no ha sigut tant aprofitable com l'anterior ja que les activitats de text són molt diferents les unes de les altres i s'han hagut de fer modificacions considerables per a la realització de les activitats desenvolupades per nosaltres.

Tot seguit trobem el diagrama de classes d'aquesta arquitectura a partir de la classe Activity, que és la que hem descrit en l'apartat anterior d'arquitectura d'Activitats.

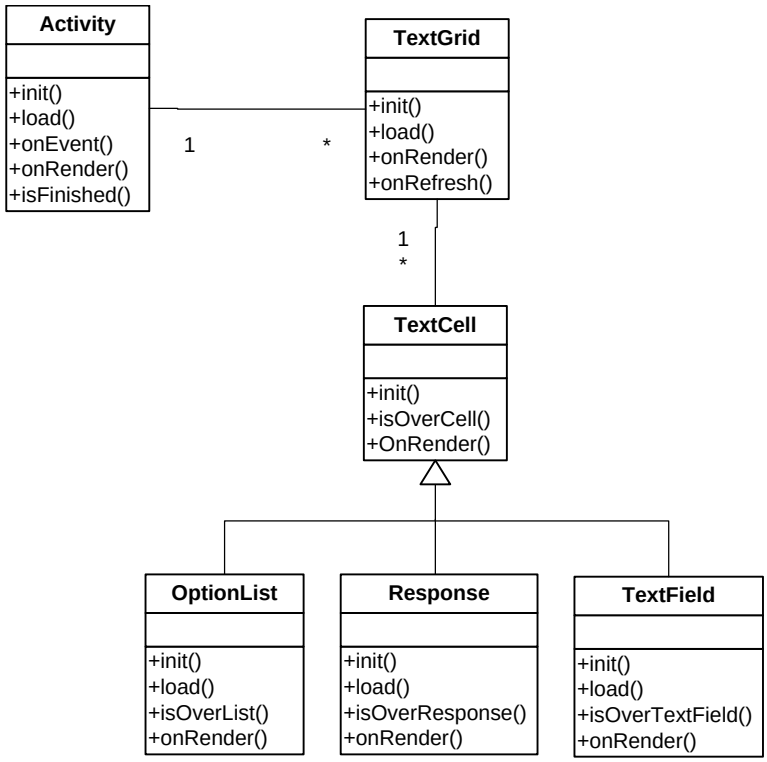


Figura 12: Diagrama de classes de l'arquitectura de text

Pel que fa a les TextGrid té les següents funcions:

- **Init:** Com totes les classes de Python, la funció que ens serveix per inicialitzar les variables de la classe.
- **Load:** funció que llegeix tot l'xml relatiu a les activitats de text cada tipus de text (sigui text normal, text de solució, casella per a seleccionar la solució, etc.)
- **Add:** Acció privada que afegeix a l'activitat i a la visualització el tipus de text citat anteriorment.
- **OnRender:** funció on es repinten els elements visuals que formen part de l'activitat.
- **OnRefresh:** és la funció OnRender desenvolupada expressament per a l'activitat OrderText, ja que en aquesta activitat, a la diferència de les altres de tipus text, s'ha de repintar de nou refrescant la posició on hi ha cada paraula.
- **changeImages, Unsort:** funcions privades per a desordenar o reordenar el TextOrder.

7.2.3 Disseny del desenvolupament d'activitats.

Pel que fa el disseny de les activitats s'ha seguit l'arquitectura citada anteriorment incorporant-hi les quatre activitats desenvolupades per nosaltres. Aquestes activitats són les de Associació Simple, Associació Complexa, Ordena Text i Resposta Escrita.

A continuació podem veure el diagrama de les classes aportades per nosaltres i comunicades amb les arquitectures d'activitats a partir de la classe Activity (la que es comunica directament amb el ClicActivityHandler).

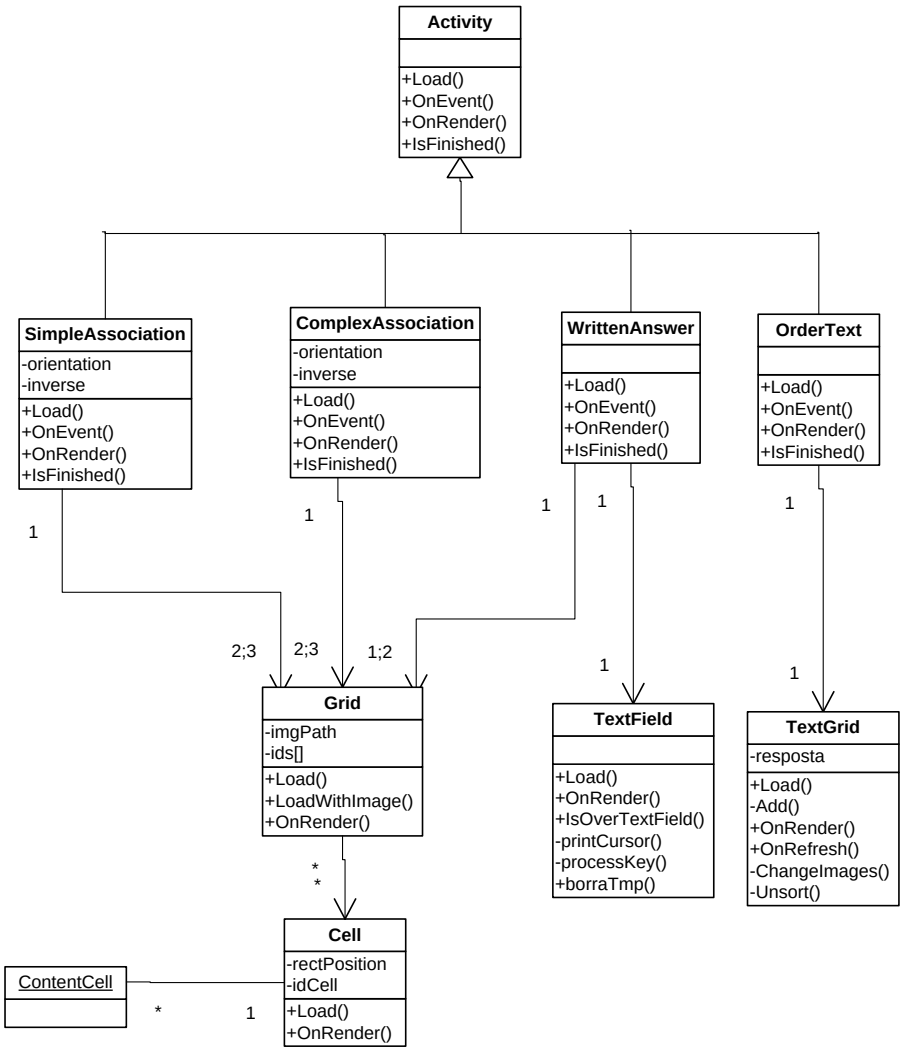


Figura 13: Diagrama de les classes aportades per nosaltres

7.2.4 Implementació de les activitats

Tot seguit explicarem la implementació de cada una de les activitats, que surten al disseny anterior. Per això explicarem de cada una el comportament que tenen envers l'usuari, és a dir, el que ja podíem observar al JClic original, i posteriorment el comportament que té el seu codi amb les funcions heretades de l'arquitectura corresponent.
Abans però, explicarem com s'ha dut a terme el parseig de l'XML de les activitats *clic* a partir de l'arxiu *.jclic*.

7.2.4.1 Parseig XML

En aquest apartat explicarem com a partir del codi XML, hem obtingut la informació relativa a les activitats. Per a veure el comportament específic observat gràcies al codi XML de cadascuna de les activitats que hem implementat caldria anar a l'ANNEX D: IMPLEMENTACIÓ DE LES ACTIVITATS A PARTIR DE L'XML DEL JCLIC (pàgina 115) .
Per això s'ha utilitzat la biblioteca del Python *minidom*, explicada ja a l'apartat de tecnologies emprades. Per tal de parsejar un document XML, només cal invocar la funció *parse* de *minidom* indicant-li la ruta fins al document que es vol processar. El resultat es guarda en una variable que a partir de llavors, contindrà la informació i a través de la qual es pot agafar. A diferència d'altres biblioteques, no calen altres elements per fer el parseig.
Per exemple:

```
xmlActivity=parse("/home/Activitat.jclic")
```

Això guardaria a la variable *xmlActivity* el resultat de parsejar Activitat.jclic i a partir d'aquí podrem navegar pel seu XML com si es tractés d'un arbre amb els seus fills o nodes.

La funció més utilitzada per poder obtenir la informació de les activitats és *getElementsByTagName("nomEtiqueta")*. Aquesta funció permet obtenir el conjunt d'elements XML que estan etiquetats amb el nom *"nomEtiqueta"* indicat com a paràmetre de la funció, en forma de vector.
Tot seguit tenim l'XML d'una activitat, concretament d'una Associació Simple

```
- <activity class="@associations.SimpleAssociation" name="plantes01" code="">
    <description/>
  + <messages></messages>
  + <settings margin="8" report="true" reportActions="false"></settings>
  + <cells rows="4" cols="1" border="false" image="planta01.jpg" id="primary"></cells>
  - <cells rows="4" cols="1" cellWidth="112.0" cellHeight="68.0" border="true" id="secondary">
    + <style borderStroke="0.7" markerStroke="5.0"></style>
      <shaper class="@Rectangular" cols="1" rows="4"/>
    + <cell txtAlign="left,middle"></cell>
    + <cell txtAlign="left,middle"></cell>
    + <cell txtAlign="left,middle"></cell>
    + <cell txtAlign="left,middle"></cell>
    </cells>
  + <cells rows="4" cols="1" border="true" image="planta01S.jpg" id="solvedPrimary"></cells>
    <scramble times="31" primary="false" secondary="true"/>
    <layout position="AB"/>
  </activity>
```

Figura 14: XML d'una Associació Simple

En la figura anterior podem veure que l'XML està format per les etiquetes <message>, <settings>, <cells>, <scramble> i <layout>.

El contingut de les etiquetes *<cells>* ens indica la informació que s'ha de mostra a la graella i per això és comú obtenir el seu contingut en cada una de les activitats.

Per tant, utilitzant la funció **Grids = xmlActivity.getElementsByTagName("cells")** obtindrem un vector on cada una de les posicions serà l'xml contingut entre les etiquetes <cells></cells>, és a dir, la variable *Grids* serà un vector que tindrà tantes posicions com etiquetes <cells> tingui l'activitat.

A partir d'aquí podem obtenir la informació continguda dins de les etiquetes de dues formes:

1. Utilitzant la funció *getAttribute("nomAtribut")*.

   Si volguéssim obtenir la imatge de la primera graella:

   ```
   + <cells rows="4" cols="1" border="false" image="planta01.jpg" id="primary"></cells>
   ```

   hauríem de fer: `Grids[0].getAttribute('image')` i això ens donaria com a resultat *planta01.jpg*.

2. Amb l'atribut nodeValue directament del mateix node.

   Agafar la informació directament continguda dins d'unes etiquetes com per exemple en el cas dels missatges:

   ```
   <p>Troba el significat d'aquestes paraules</p>
   ```

   amb l'atribut *nodeValue* , de la següent manera:

   ```
   cell.getElementsByTagName('p')[0].firstChild.nodeValue
   ```

Per navegar pel contingut de les cel·les de cada una de les graelles, com per exemple de la segona graella, caldrà recórrer el vector format per les etiquetes <cell> (Notem que és *cell*, i no *cells*) . Per això obtindrem un altre cop l'xml relatiu a aquest codi amb la funció **Grids[1].getElementsByTagName("cell").**

Aquestes són les instruccions bàsiques a tenir en compte pel parseig d'XML. Com hem dit abans, . per a veure el comportament específic observat gràcies al codi XML de cadascuna de les activitats que hem implementat caldria anar a l'ANNEX D: IMPLEMENTACIÓ DE LES ACTIVITATS A PARTIR DE L'XML DEL JCLIC (pàgina 115) .

7.2.4.2 Desenvolupament d'activitats

Tot seguit explicarem per a cada una de les activitats que hem adaptat nosaltres, el seu comportament inicial en el JClic i alguns detalls de la seva implementació.

Activitat Associació Simple (SimpleAssociation.py)

COMPORTAMENT

Es presenten dos conjunts d'informació que tenen el mateix nombre d'elements. A cada element del conjunt origen correspon un i només un element del conjunt destí.

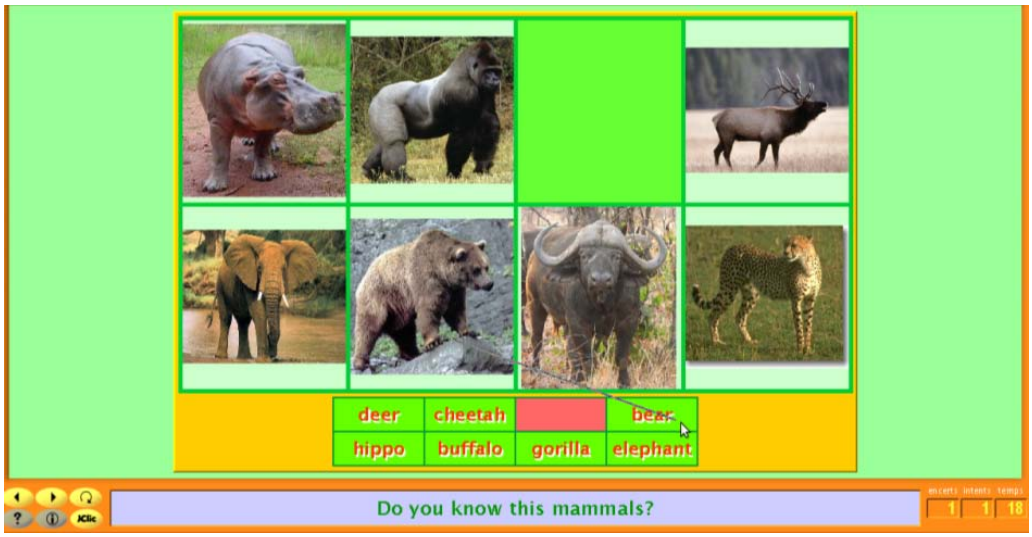


Figura 15: Activitat Associació Simple original del JClic

Com podem veure en la imatge anterior extreta del JClic original relacionem una imatge de la graella superior amb un text de la inferior.

IMPLEMENTACIÓ

Aquesta va ser la primera activitat que vam implementar i, per tant, la que més temps ens va portar al haver d'adaptar-nos a l'arquitectura citada anteriorment.

Com podem veure en la figura anterior, al clicar una de les cel·les, s'imprimeix per pantalla contantment una línia que va seguin el ratolí en el moment i que no desapareixerà fins que cliquem una de les cel·les de l'altra graella. Seguint amb el requisit de l'eficiència del que ha de gaudir l'aplicació i aprofitant l'arquitectura d'activitats ja implementada s'ha canviat aquest sistema per un

altre basat en que, al clicar una cel·la, aquesta es bordeja de color vistós de tal manera que l'usuari ja sap quina cel·la ha clicat sense que s'hagi de mostrar una línia constant amb la que hauríem de refrescar la pantalla de forma permanent.

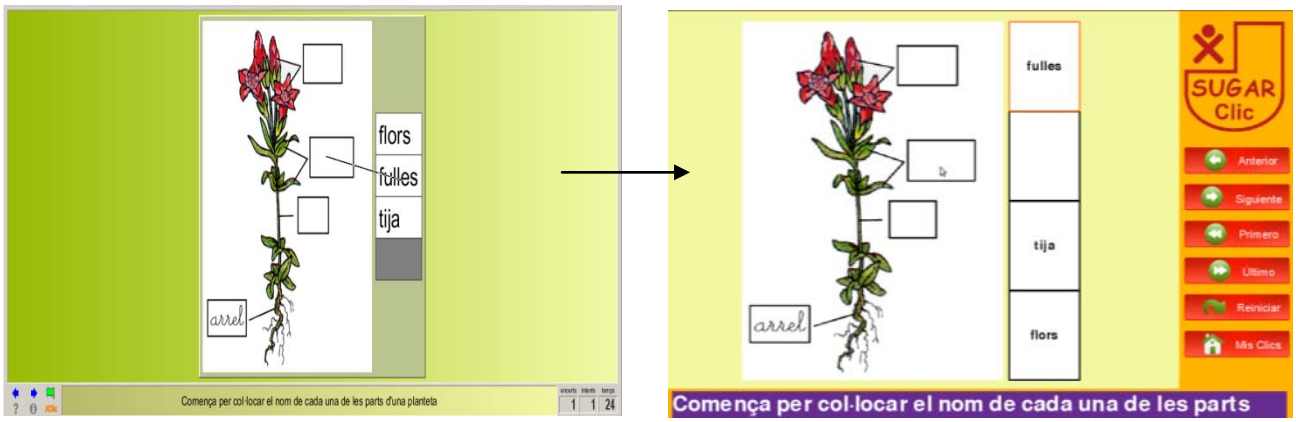


Figura 16: Comparativa de l'Associació simple del JClic (esquerra) amb la del SugarClic (dreta)

Les activitats d'Associacions Simples són molt vistoses i, com podem veure a la imatge anterior, permeten també relacionar un nom directament amb una part de la imatge, que en realitat no és res més que una graella formada per una imatge dividida en una columna i quatre files, on cada cel·la és una part d'una imatge.

De tal manera que al codi tenim les següents accions heretades de la classe Activity.py:

- **Load:**

  Carrega les graelles i mostra la 1 i la 2 per pantalla, s'assignen els ids per a cada graella.

- **OnEvent:**

  Comprova si s'ha clicat alguna de les cel·les de les dues graelles. Si és així, es faran una sèrie de comprovacions (si hi havia una de l'altra graella pressionada, si la cel·la ja formava part de les cel·les resoltes, etc.) i, si coincideixen els identificadors de la cel·la pressionada amb el de la cel·la marca anteriorment, es mostrarà la cel·la de la graella de resolució (la tercera) substituint la cel·la de la graella corresponent i l'altra s'omplirà amb una cel·la buida, traient-li les funcionalitats a totes dues cel·les.

- **OnRender:**

  S'actualitza la pantalla

- **IsFinished:**

  Si totes les cel·les estan inoperatives s'ha resolt.

Activitat Associació Complexa (ComplexAssociation.py)

COMPORTAMENT

Al igual que al SimpleAssociation, també es presenten dos conjunts d'informació, però aquests poden tenir un nombre diferent d'elements i entre ells es poden donar diversos tipus de relació: un a un, diversos a un, elements sense assignar...

Figura 17: Activitat Associació Complexa del JClic original

IMPLEMENTACIÓ

La implementació de les aplicacions bàsiques d'aquesta activitat ha sigut senzilla ja que després de fer l'Associació Simple, aquesta era el mateix però comprovant la relació una a diversos.

Però aquesta activitat és força completa ja que a part del mateix que passava amb la Associació Simple de la situació de les graelles (de costat o a dalt i a baix), també teníem el problema de donar-se diferents tipus de relació (un a varis o varis a un) i d'haver-hi caselles desactivades, és a dir, sense cap funcionalitat al fer-hi clic però importants a l'hora de visualitzar el conjunt. Això ho podem veure en la següent il·lustració:

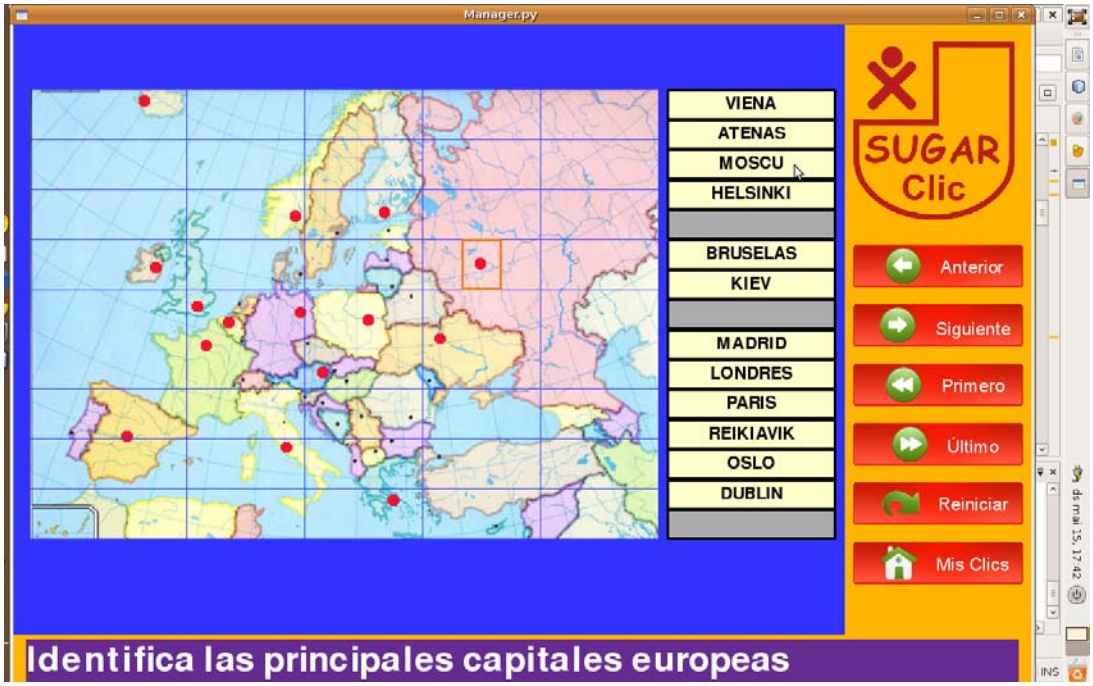


Figura 18: Activitat Associació Complexa del SugarClic

Com podem veure, la graella de l'esquerra és un mapa que en realitat és una imatge dividida en 9 files i 16 columnes, és a dir 144 cel·les, però que només tenen funcionalitat 15 d'elles. Això no s'havia tingut en compte en l'arquitectura d'activitats, per tant, hem hagut de fer algunes modificacions en la classe Activity.py.

A continuació es detallen les operacions que es fan en cada una de les funcions heretades de la classe Activity.py:

- **Load:**

  Carrega les graelles i mostra la 1 i la 2 per pantalla, s'assignen els ids per a cada graella.

- **OnEvent:**

  Comprova si s'ha clicat alguna de les cel·les de les dues graelles. Si és així, es faran una sèrie de comprovacions (si hi havia una de l'altra graella pressionada, si la cel·la ja formava part de les cel·les resoltes, etc.) i, si coincideixen els identificadors de la cel·la pressionada amb el de la cel·la marca anteriorment, es mostrarà la cel·la de la graella de resolució (la tercera) en una de les graelles traient-li també les funcionalitats que tenia i l'altra es mantindrà igual.

- **OnRender:**

  S'actualitza la pantalla.

- **IsFinished:**

  Si totes les cel·les de la graella corresponent estan inoperatives, l'activitat s'ha resolt.

Activitat Ordena Text (OrderText.py)

COMPORTAMENT

En el moment de dissenyar l'activitat se seleccionen en el text algunes paraules o paràgrafs que es barrejaran entre si. L'usuari ha de tornar a posar-ho en ordre.

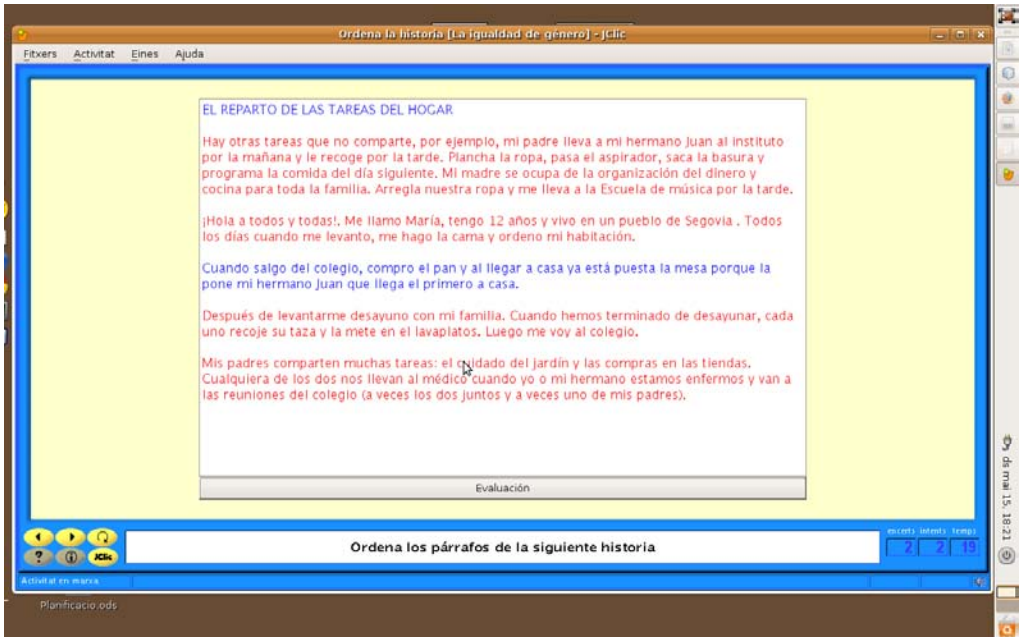


Figura 19: Activitat Ordena Text del JClic original

Com podem veure es tracta de ordenar paraules o grups de paraules dins d'un text i quan aquest estigui resolt s'apreta el botó d'Evaluació per veure si està correcte.

IMPLEMENTACIÓ

En el canvi de JClic a SugarClic s'ha tret el botó d'Evaluació per agilitzar l'activitat i hem mostrat directament les paraules correctament ordenades resaltant-les de color verd.

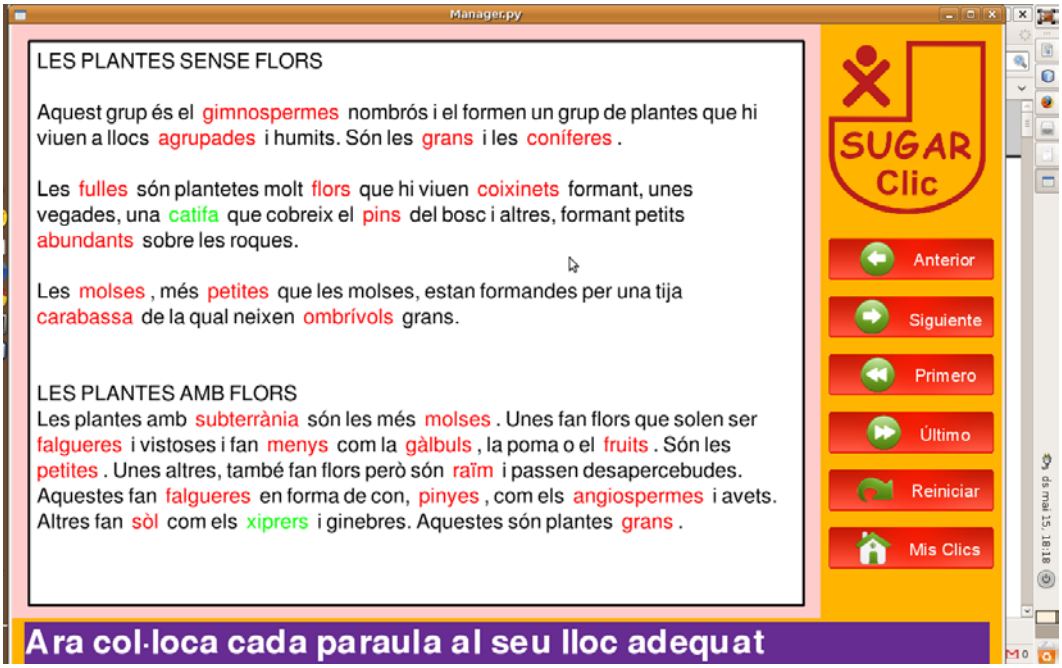


Figura 20: Activitat Ordena Text del SugarClic

Per aquesta activitat s'ha utilitzat l'arquitectura de les activitats de Text: el TextGrid. D'aquesta arquitectura només hem aprofitant la correcta visualització del text en vàries línies ja que s'ha hagut d'adaptar tota la funció de OnRender, fent-ne una de nova: OnRefresh. Això és degut a que per aquesta activitat, a diferència de les altres de tipus text, s'ha de repintar de nou refrescant la posició on hi ha cada paraula al canviar-les de lloc, és a dir, que per exemple en la figura anterior, si canvien la paraula *"gimnospermes"* per la paraula "*sòl*", no ocupen el mateix espai, i per tant les paraules canviaran el tamany de l'espai que ocupen, redreçant així tot el paràgraf.

També s'ha hagut d'afegir al TextGrid les funcions changeImage i unsort, basant-nos en l'arquitectura dels Grids, per a desordenar el texts de les solucions entre ells.

Les funcions en aquesta activitat són les següents:

- **Load:**

  Carrega el text.

- **OnEvent:**

  Indica un canvi d'ordre en les solucions si l'usuari ha clicat dos textos solució anteriorment.

- **OnRender:**

  S'actualitza la pantalla, delega tota la responsabilitat a la funció OnRefresh de la TextGrid.

- **OnFinished:**

  Si coincideixen de longitud el vector de solucions correctes amb el vector de solucions totals, l'activitat s'haurà acabat.

Activitat Resposta Escrita (WrittenAnswer.py)

COMPORTAMENT:

Es mostra un conjunt d'informació i, per a cadascun dels seus elements, cal escriure el text corresponent.

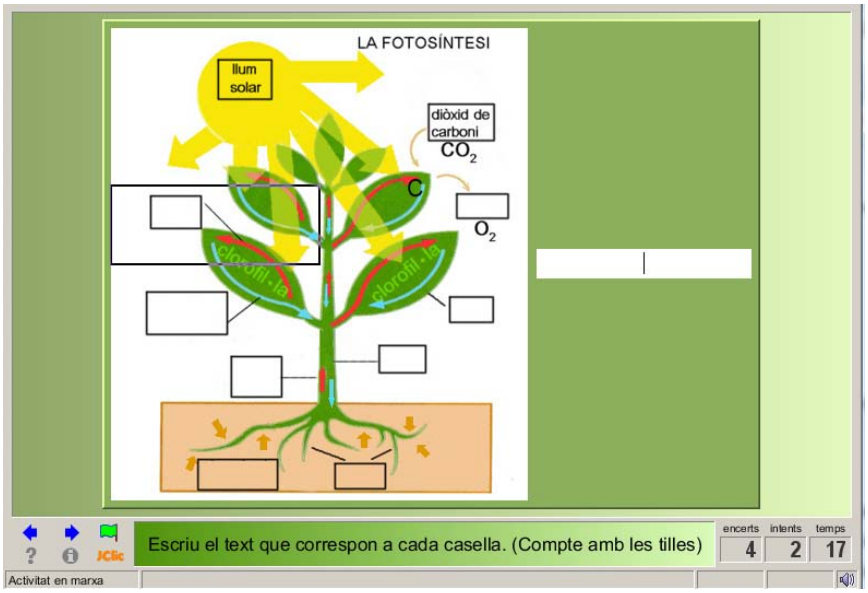


Figura 21: Activitat Resposta Escrita del JClic original

IMPLEMENTACIÓ:

Tot i que és una activitat que intervé el text, té un funcionament similar al que pugui tenir l'Associació Complexa, però en comptes de relacionar una graella amb l'altra, només n'hi ha una a la qual hem d'escriure el nom que correspongui. Aquesta també té la opció de tenir una graella per la solució.

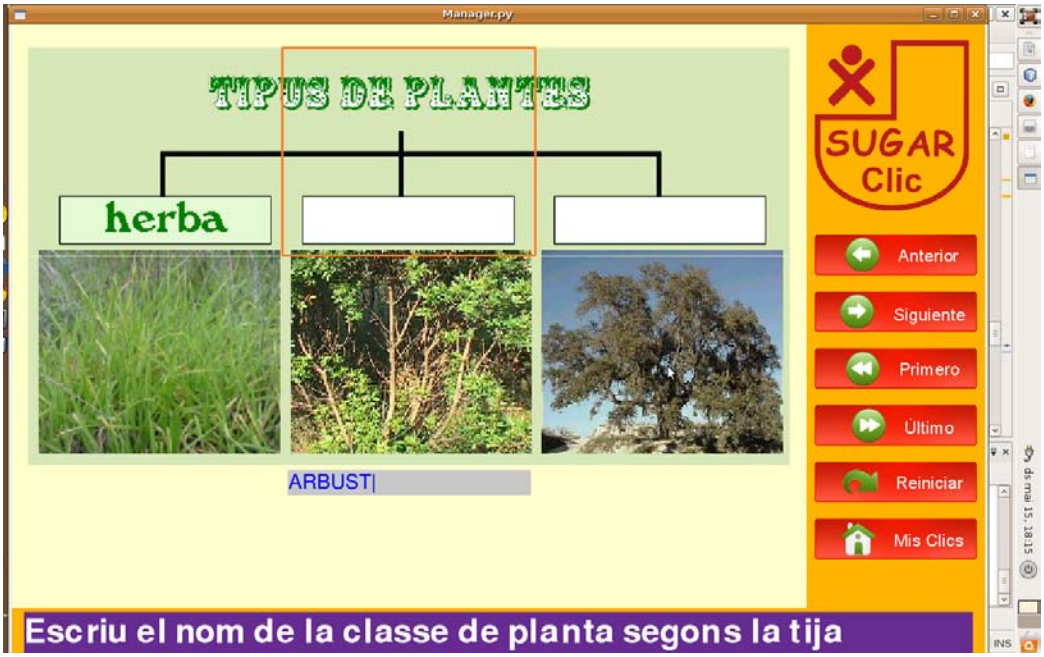


Figura 22: Activitat Resposta Escrita del SugarClic

S'ha tingut també en compte que l'usuari no hagi d'anar clicant sobre la cel·la que vol resoldre obligatòriament, sinó que, al igual que passa amb l'activitat original, que l'usuari pressiona l'Enter.

Per aquesta activitat s'ha aprofitat la classe TextField, classe que en principi era per les activitats de text, i s'ha adaptat per obtenir la solució.

Aquest activitat ha estat de les que hem fet una adaptació més fidel respecte a la del JClic original.

Les funcions utilitzades aquí són les següents:

- **Load:**

  Carrega els Grids amb les seves solucions i el quadre de text per escriure la solució (la classe TextField) .

- **OnEvent:**

  Aquesta funció només controla els esdeveniments de ratolí, és a dir, que només controlarem si l'usuari ha clicat sobre

- **OnKeyEvent:**

  Aquí es processa la tecla passada al TextField i es comprova si la solució escrita correspon al

- **NextPiece:**

  Funció privada que després de que l'usuari encerti pressionant Enter sobre una casella, aquesta busca

7.3 Millora del Projecte

Tot i tenir actualment totes les activitats creades, durant i després de la realització d'aquestes activitats hem fet un procés de millora i correcció del projecte.

La metodologia que hem seguit, tot i la millora en quan a eficiència, cal dir que al ser més una adaptació que una traducció literal, ha fet que haguéssim de considerar i abarcar quantitats considerables de casos possibles pel què fa al comportament del JClic i de les seves activitats. I és per això que el nostre projecte ens hem dedicat en gran part al testing i millora de les activitats.

Per a realitzar aquest procés de proves i millora ens hem basat en la metodologia de Bug and Feature Hunting.

7.3.1 Metodologia: Bug and feature hunting

El Bug Hunting és una activitat que barreja la revisió per parelles amb el Testing Exploratori. Més concretament, el Bug Hunting són exercicis informals de test, basant-nos en fer proves per realitzar un objectiu clar i sense perdre el temps. Aquest objectiu principal és trobar *Bugs* o errors, millorar la qualitat del Software que estem desenvolupant amb l'afegit de sortir una mica de la rutina i integrar l'equip com un tot, eliminant aquest límits entre els diferents rols dins d'una estructura de projecte.

El bug hunting fomenta els següents punts:

- Trobar els bugs o errors que a vegades no són trobats en els casos de prova.
- Enfortir el treball en equip.
- Relacionar-se amb gent amb la qual no es treballa habitualment en equip.
- Facilitar el coneixement de l'aplicació com un tot. Ja que hi ha gent especialitzada en certs mòduls i funcionalitats però que desconeixen el funcionament de l'aplicació com un tot.
- Detecció d'errors intervenint en l'aplicació com si fóssim usuaris.
- Millorar l'ànim de l'equip al sortir una mica de la rutina del projecte.

7.3.2 Utilització del Google Code

Hem fet una adaptació de la metodologia mencionada en l'apartat anterior combinada amb l'eina que ens ha aportat Google Code.

Tot i explicar l'eina en l'apartat de tecnologies emprades, aquí detallarem com hem utilitzat els avantatges que ens ha aportat.

Cada projectista realitzava proves de diferents amb el SugarClic de diferents clics que s'hagués pogut descarregat d'Internet (per exemple de la pàgina de la xtec: http://clic.xtec.cat/db/listact_ca.jsp).

Per fer les proves hem trobat una incompatibilitat amb el nostre projecte. Aquesta incompatibilitat és que al descarregar-nos les activitats clic, el SugarClic no codifica els accents que poden tenir els arxius adjunts a aquestes activitats. Per això hem d'adaptar aquelles activitats que puguin tenir accents en els seus fitxers, com és el cas de les imatges, ja que sinó la imatge relacionada no es veurà a la graella.

Si el projectista veia algun error pel que fa al seu funcionament o a la visualització, aquest error era reportat per ell a la pàgina web de Google Code en forma d'incidència. Com que cada projectista s'encarregava més o menys d'una part de codi, aquesta incidència l'assigna la persona que ha trobat l'error al projectista responsable del codi o qui ell consideri adequat per a resoldre aquesta incidència.

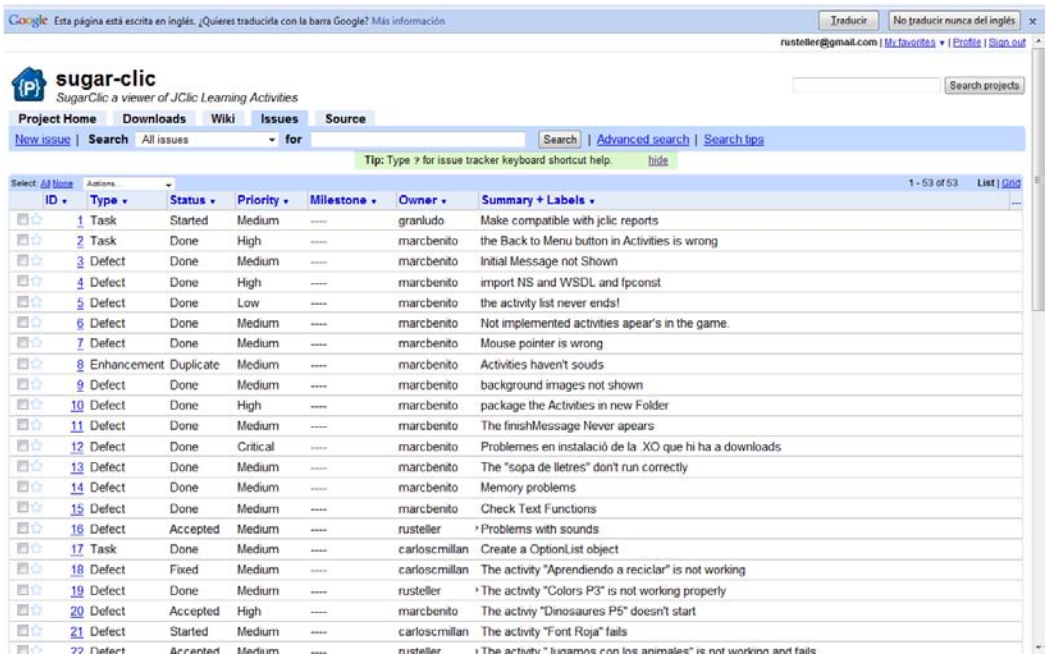


Figura 23: Pàgina de Google Code on es reporten les incidències

Aquestes incidències es poden anar afegint comentaris mentre es vagin fent observacions o correccions i un cop resolta, el projectista que l'ha resolt ho reporta i es tanca la incidència.

7.3.3 Aportacions a la millora del projecte.

Tot seguit detallarem cada un dels punts que hem solucionat nosaltres pel que fa a la millora o correcció del SugarClic. Descriurem tant els punts que se'ns hagin assignat d'altres projectistes o que els haguem trobat nosaltres directament. Cal dir que fent les proves també hem trobat errors que s'han assignat o notificat a altres desenvolupadors.

- **Primera versió de l'escriptura de text en vàries línies.**

  La primera versió del visualitzador de clic del SugarClic no permetia veure més d'una línia en una cel·la, de tal manera que les activitats que contenien cel·les amb texts llargs eren il·legibles.

  La nostra aportació va ser la creació d'una funció simple que permetia que el text es pogués visualitzar en múltiples línies comparant la longitud del rectangle del text amb la longitud de la cel·la que el comprenia. Posteriorment aquesta funció va ser depurada i millorada per un altre projectista i reaprofitada en les activitats de text dins del TextGrid.

- **Substitució del cursor per l'original del OLPC.**

  El pygame utilitza com a cursor per defecte l'habitual de la fletxa blanca. Però l'OLPC utilitza un cursor més gros i més visible per als nens.

  Per una correcta visualització vam integrar aquest cursor i el d'una mà fent el símbol de la "V", semblant al que té el JClic per defecte quan es realitza correctament una activitat.

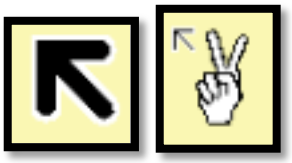


Figura 24: Cursors nous del SugarClic

- **Posar identificadors per un Grid**

  Una altra millora o correcció d'error ha sigut un tema que no es tenia en compte en l'arquitectura d'activitats i que feia que activitats com el Panell d'Exploració o les Associacions tinguessin problemes de visualització:

  Tot i que les cel·les per sí mateixes poden tenir el seu identificador (id), també és possible que tots els ids estiguin inclosos en la graella. Tot seguit mostrem un tros de codi xml del *.jclic* on això passa:

  ```
  <cells rows="4" cols="12" border="false" image="prueba1.jpg" id="primary">
  <style/>
  <shaper class="@Rectangular" cols="12" rows="4"/>
  <ids>
  -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 3 -1 -1 -1 -1 -1 -1 -1 -1 0 -1
  -1 -1 2 5 4 -1 -1 -1 -1
  </ids>
  </cells>
  ```

  Com podem veure en el codi XML anterior hi ha una sèrie de ids separats per un espai. Això s'ha hagut de tractar i també fer que les cel·les amb un *-1* no siguin operatives, cosa que tampoc es tenia en compte.

- **Millora dels sons (encara no compatible amb l'OLPC).**

  La nostra aportació en el tema dels sons va ser fer la funció que els reprodueix i tractar quan els ha de reproduir en les activitats. Es va fer la funció general de reproduir i es va implementar la seva crida en 3 de les activitats veient que funcionava en el Sistema Operatiu Ubuntu, tot i que posteriorment, al fer les proves amb l'OLPC es va veure que es produïa un error greu que feia que el programa deixés de funcionar.

- **Salts entre activitats mitjançant clics a cel·les concretes.**

A alguns dels conjunts d'activitats JClic tenim una activitat a l'inici de tipus *Panell d'Informació* que actua de menú i que ens permet saltar a una activitat concreta de tal manera que podem anar directament a un conjunt d'activitats.

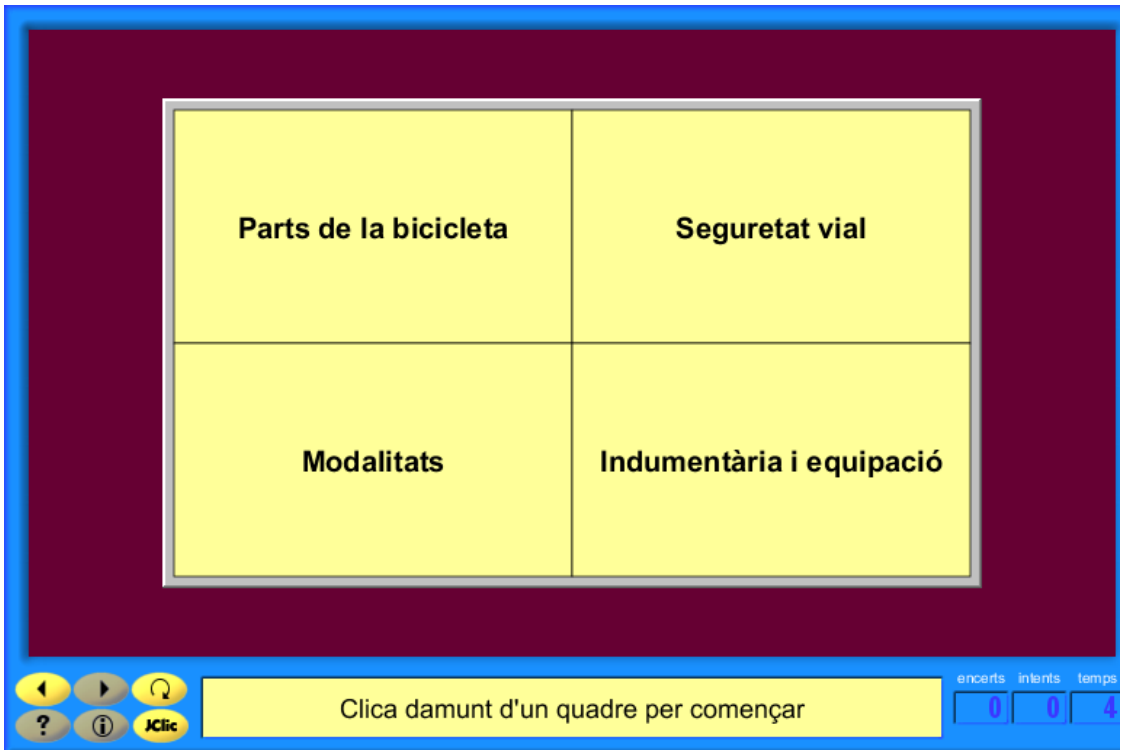


Figura 25: Activitat del JClic original on podem saltar a diferents llocs

Això s'ha realitzat afegint un nova classe anomenada SequenceElement que, associada a la Sequencer, controla aquests salts entre activitats.

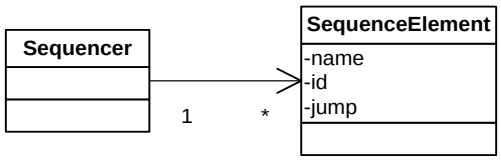


Figura 26: Nova claca afegida SequenceElement

En la part de la seqüència del *.jclic* tenim el següent codi per identificar l'element, un id:

```
<item id="parts" name="parts bici">
<jump action="RETURN" id="back"/>
</item>
<item name="joc de memòria"/>
```

<item name="puzzle doble"/>

Al fer clic sobre un element el l'activitat Panell d'Identificació, aquesta controla si la cel·la té el codi següent:

<cell>
<media type="RUN_CLIC_PACKAGE" level="1" file="parts"/>
* Parts de la bicicleta *
</cell>

- **Modificació del disseny visual del SugarClic.**

El disseny del SugarClic fet fins a la data era senzill: un parell de botons i una barra per mostrar el missatge. Aquest disseny però es va modificar per tal que hi haguessin més botons per ser més intuïtiu i fos més vistós.

Tot seguit tenim la comparació dels dos dissenys:

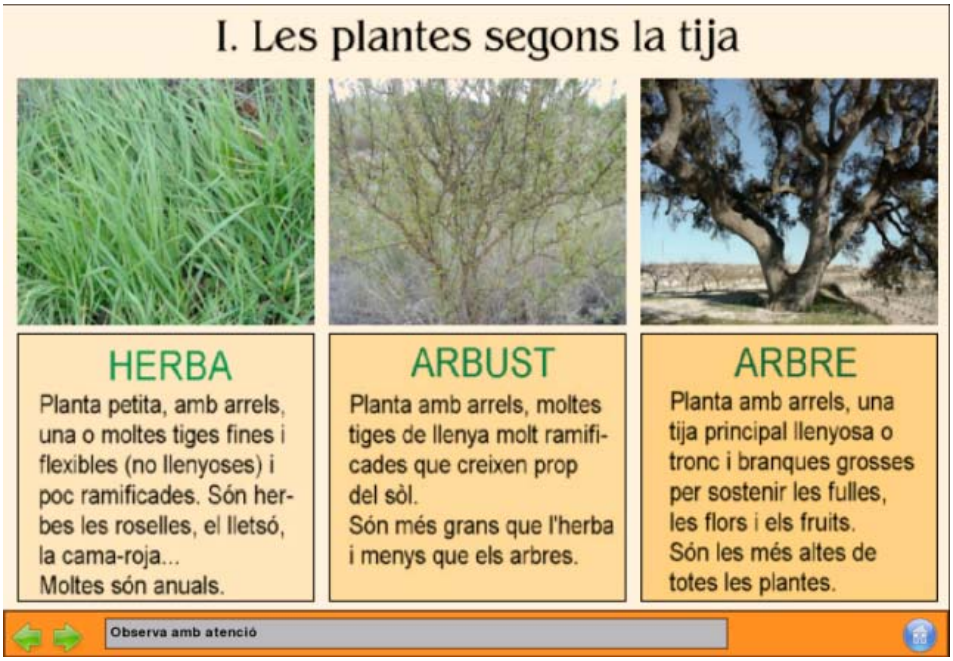


Figura 27:Dissenya anterior del SugarClic

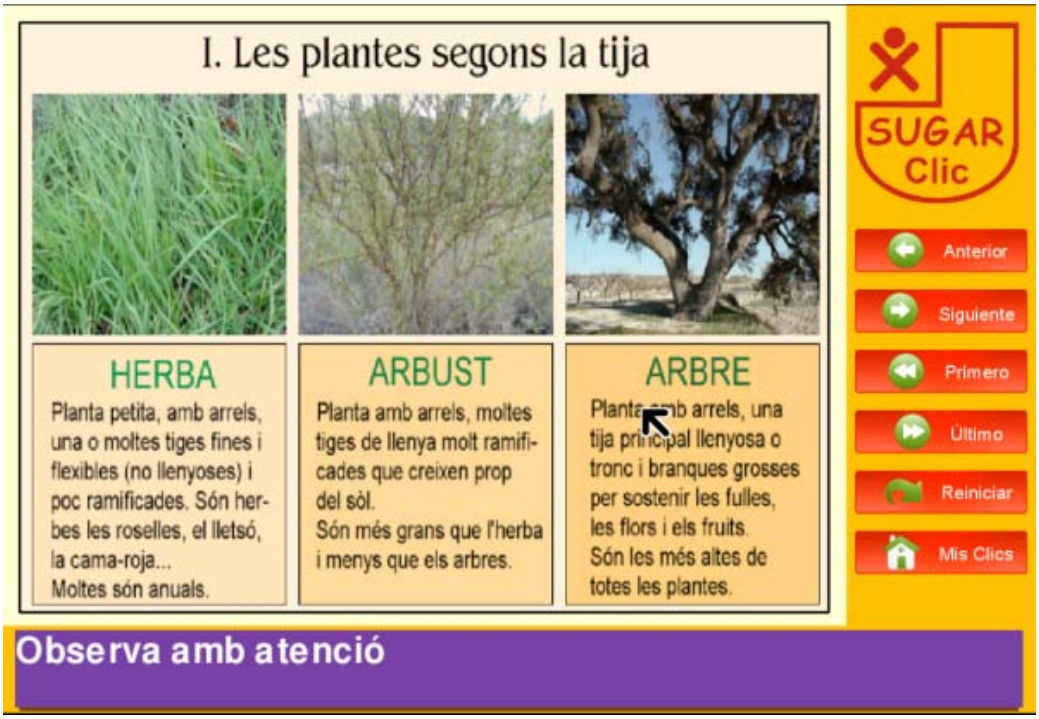


Figura 28: Disseny actual del SugarClic

Com podem veure a la imatge es va posar una barra per a l'escriptura del missatge més gran de tal manera que les lletres del missatge fossin més grans o fins i tot escriure en dues línies (aprofitant la funció d'escriptura per vàries línies).

S'ha afegit un logotip del SugarClic proporcionat per un col·laborador (Fernando Pelillo) i s'han canviat els botons per uns més vistosos amb les lletres corresponents a la funció dels botons a la seva dreta.

Es van afegir també les funcionalitats d'anar a la primera activitat de la seqüència, a la última i reiniciar l'activitat actual.

- **Realització de dues activitats (el manual i l'about) per a facilitar l'ús i de les eines del SugarClic i saber la donar a conèixer el nostre projecte.**

  Realització de dos clics utilitzant l'eina JClic Author que intenten fer una breu introducció del SugarClic per facilitar el seu ús a l'usuari en el cas del *Manual*; i un *About* per donar a conèixer qui som, què fem, quina llicència utilitzem i quina es la nostre web principal.

  Aquest dos clics són clics senzills composats per Panells d'Informació tot i que tenen la diferència de que al acabar s'ha de retornar a la pàgina principal del SugarClic, cosa que no fan

la resta d'activitats. S'han hagut de tractar aquests casos afegint un petit text XML i tenint-ho en compte en el codi (en el Sequencer) perquè es tractin com a tal.

Figura 29: Primera pantalla del Manual

- **Millora de la visualització en els tamanys de les imatges.**

  En l'aplicació anterior, les imatges eren redimensionades al tamany que ocupava la cel·la totalment i, si hi havia text, aquest es superposava amb la imatge, de tal manera que a vegades el text era il·legible.

  Per això, al fer les proves i observant el codi es va trobar un codi associat a les cel·les dins de l'XML del fitxer *.jclic* que ens indica si el text i la imatge han d'estar sobreposats o no.

Figura 30: Cas on el text i les imatges no es sobreposen

També tenim el codi del .jclic que ens indica a on ha d'estar una imatge respecte la cel·la, i en aquests casos es va deixar de maximitzar la imatge per tal de situar-la a on el codi ens indiqui.

- **Resolució de problemes amb les vores de les cel·les**

Abans les vores de les cel·les és visualitzaven sempre. Hem afegit a la classe Cell, un atribut *hasBorder* que comprovarà l'atribut "border" que tenen les cel·les al *.jclic* i que ens indica si la cel·la ha de portar vores o no.

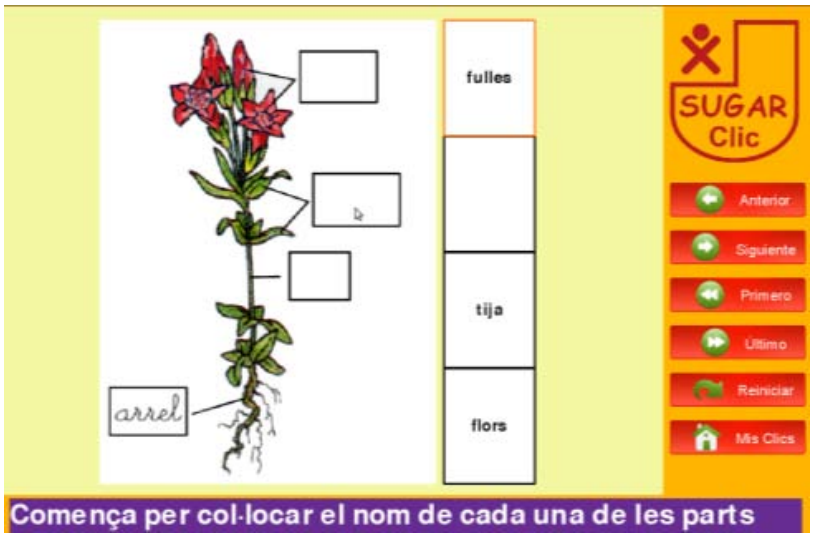


Figura 31: Imatge sense bores

A l'anterior figura mostrem una activitat que no ha de portar bores per defecte. La imatge de l'esquerra són 4 cel·les que tenen l'atribut border=False

8. Planificació i valoració econòmica

En aquest apartat s'explica la planificació seguida en el projecte.

Al ser el SugarClic un projecte amb varis membres, en general ha sigut un projecte canviant i la nostra aportació en ell s'ha vist modificada a mitjans d'aquesta planificació degut a que ens hem hagut d'adaptar a les incorporacions d'aquests membres.

Primerament farem una breu descripció de les etapes del projecte, després desglossarem les tasques que s'han dut a terme durant la realització d'aquest, i posteriorment explicarem veurem la duració planificada i real de cadascuna d'aquestes tasques.

Per acabar comentarem i compararem l'estimació i la duració real d'aquesta planificació.

8.1 Descripció de les etapes

Tot seguit explicarem les etapes més significatives del nostre projecte:

1. **Aprenentatge de l'entorn**

   Com la majoria de projectes, aquest ha tingut una etapa d'aprenentatge de l'entorn. En aquesta etapa d'aprenentatge de les tecnologies del OLPC i del JClic ens hem centrat a realitzar una activitat de prova per l'XO, que no és res més que un trencaclosques.

2. **Separació i coordinació d'interfícies**

   La segona etapa ha sigut la coordinació amb un altre dels projectistes del SugarClic, el Jose Camallonga, per fer la separació de les parts i interfícies que posteriorment gestionaria i duria a terme cadascú per separat. Per fer aquesta separació es va realitzar el disseny de les classes separades i la implementació del protocol d'intercanvi d'informació entre les dues interfícies. A més a més, en aquesta etapa també es va començar a realitzar la interfície en pygame a partir de l'activitat de prova que teníem fet per aprendre l'entorn de l'OLPC.

3. **Aportació de l'arquitectura i realització d'activitats**

   La tercera etapa vindria marcada per la incorporació d'en Marc Benito i en Carlos Castilla dins del projecte. En Marc, que havia estat treballant en la realització d'activitats clic per l'OLPC, va aportar i incorporar una arquitectura i un nucli de desenvolupament d'activitats molt atractiu, el qual ens vam adaptar amb en Carlos per seguir les desenvolupant les activitats que quedaven basades en aquesta arquitectura.

4. **Testing i millora del SugarClic**

A finals del passat any hi havia comunitats que estaven a l'espera que treiéssim una versió operativa del SugarClic. Des d'aleshores vam començar tots els membres del SugarClic a efectuar tasques de testing i millora corregint els possibles errors i realitzant millores gràfiques. Primerament, quan l'aplicació no estava del tot finalitzada això es feia amb el seguiment del nostre director de projecte i distribuint la feina entre els projectistes que tenien menys treball. Però posteriorment aquestes tasques de millora s'han anat realitzant fins la finalització del projecte amb un procés continu on es provaven els clics disponibles i s'assignaven les tasques que es creien oportunes entre els mateixos projectistes de tal manera que ja al final s'ha tingut una versió força aproximada al JClic original.

En aquesta etapa també hi ha la presentació d'en Marc Benito del seu Projecte, això va fer que també féssim tasques de documentació com podria ser el manual, que en aquest projecte és l'11. ANNEXOS

ANNEX A: MANUAL D'ÚS DEL SUGARCLIC situat a la pàgina 89.

5. **Realització de les activitats de text.**

   Després de la presentació d'en Marc Benito, ja havíem realitzat totes les activitats relatives a l'arquitectura disponible. És per això que amb en Carlos Castilla vam realitzar la resta d'activitats que quedaven, les de text.

8.2 Planificació

En aquest apartat podem veure la planificació del projecte, però primerament mostrarem les tasques que s'han planificat i dut a terme durant el projecte. Aquestes són les següents:

- Definició del projecte: Tasques de definició i planificació dels objectius
- Aprenentatge de l'entorn: Les tasques relatives a la recopilació d'informació i l'aprenentatge.
  - Aprenentatge Pygame-OLPC: Per entrar en el món de l'OLPC i del pygame es va realitzar un petit trencaclosques per aquesta plataforma.
  - Aprenentatge Activitats .JClic: Per veure i estudiar el funcionament, així com el codi XML que conté cada activitat a realitzar es va haver de realitzar un període d'aprenentatge previ.

  - Aprenentatge arquitectura SugarClic: Hi va haver una etapa de coneixement de l'arquitectura d'activitats aportada per un dels projectistes del SugarClic

- Implementació i disseny
  - Interfície Pygame i comunicació amb PyGTK: Definició del protocol entre els dos llenguatges, aquesta tasca la vam fer conjuntament amb un altre projectista
  - Iniciar el reproductor d'activitats: Aquesta és una tasca que va lligada amb l'anterior i que surt de l'aprenentatge del Pygame i l'OLPC, ja que des de l'activitat de prova i la interfície de Pygame-Pygtk es va començar a construir el reproductor, que posteriorment passaria a mans d'un altre projectista que acabaria la tasca.
  - Realització d'activitats: Disseny i realització de les activitats adaptades pel SugarClic.
    - Activitat Associació simple
    - Activitat Associació complexa
    - Activitat Ordena Text
    - Activitat Resposta escrita
- Tasques de testing i millora de l'aplicació: Etapa contínua en la que s'han realitzat les proves i s'han implementat les millores corresponents.
- Documentació: Realització de la memòria i la presentació del projecte

A continuació mostrarem els diagrames de Gantt relatius a la planificació real i al temps real invertit relatius al projecte.

8.2.1 Planificació inicial

Tot seguit mostrarem en un diagrama de Gantt la planificació inicial del projecte, és a dir, el temps que havíem estimat invertir per desenvolupar cada tasca.

La planificació va des de l'estiu del passat 2009 fins a principis de Juny d'aquest any. S'han agrupat els mesos de Juliol i Agost ja que la feina planificada s'ha anat fent durant aquests mesos d'estiu.

Per facilitar la visió del diagrama s'ha fet en dues parts, l'una a sota de l'altra, de tal manera que la de baix segueix a la de dalt.

També podem veure l'estimació de la dedicació en hores en cadascuna de les setmanes i la dedicació de cada tasca en concret. Considerem que hem treballat al projecte a temps parcial, per això hi ha setmanes de 10 hores o de 20 hores i algunes de vacances.

	Dedicació	Juliol-Agost	Setembre	Octubre	Novembre	Desembre
Definició del projecte	10 hores					
Aprenentatge						
Aprenentatge OLPC-pygame	70 hores					
Aprenentatge Jclic	10 hores					
Aprenentatge arquitectura	10 hores					
Disseny i Implementació						
Interfície pygame-pygtk	20 hores					
Iniciar interfície	30 hores					
Activitat Associació simple	40 hores					
Activitat Associació complexa	50 hores					
Testing i millora	140 hores					

	Dedicació	Gener	Febrer	Març	Abril	Maig	Juny
Disseny i Implementació							
Activitat Ordena text	50 hores						
Activitat Resposta escrita	40 hores						
Testing i millora	140 hores						
Documentació	60 hores						

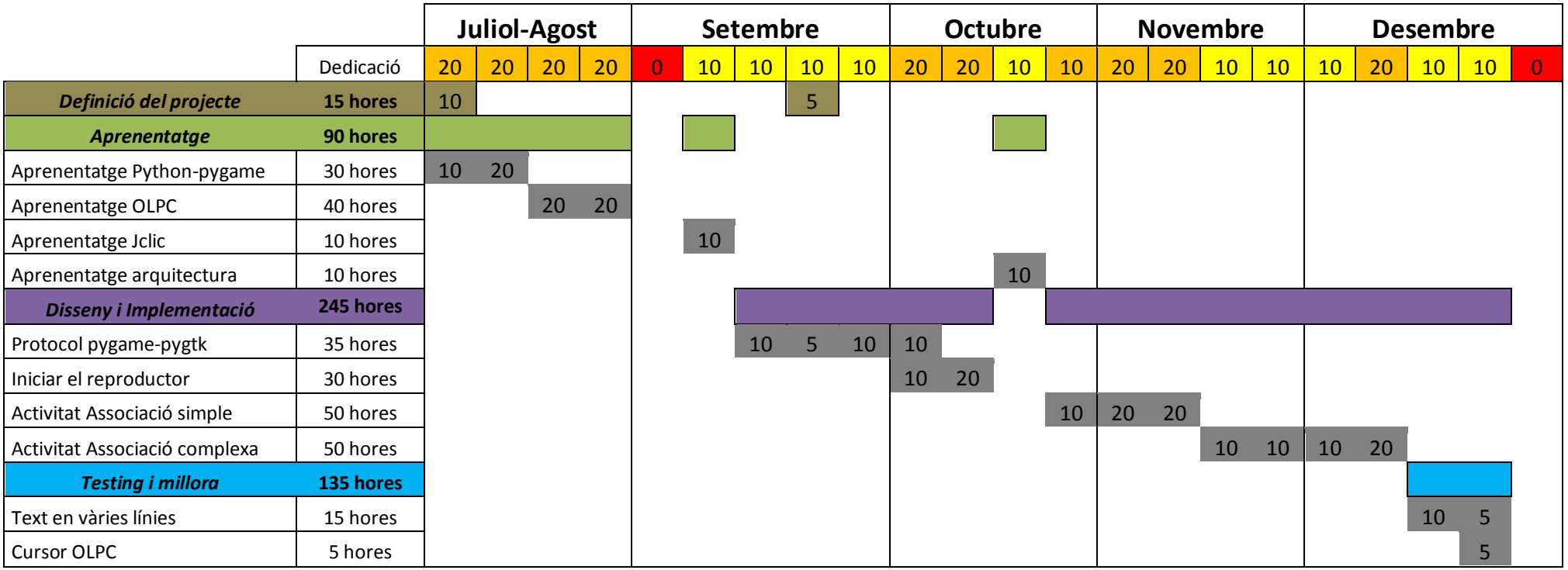
Vacances
10 hores a la setmana
20 hores a la setmana

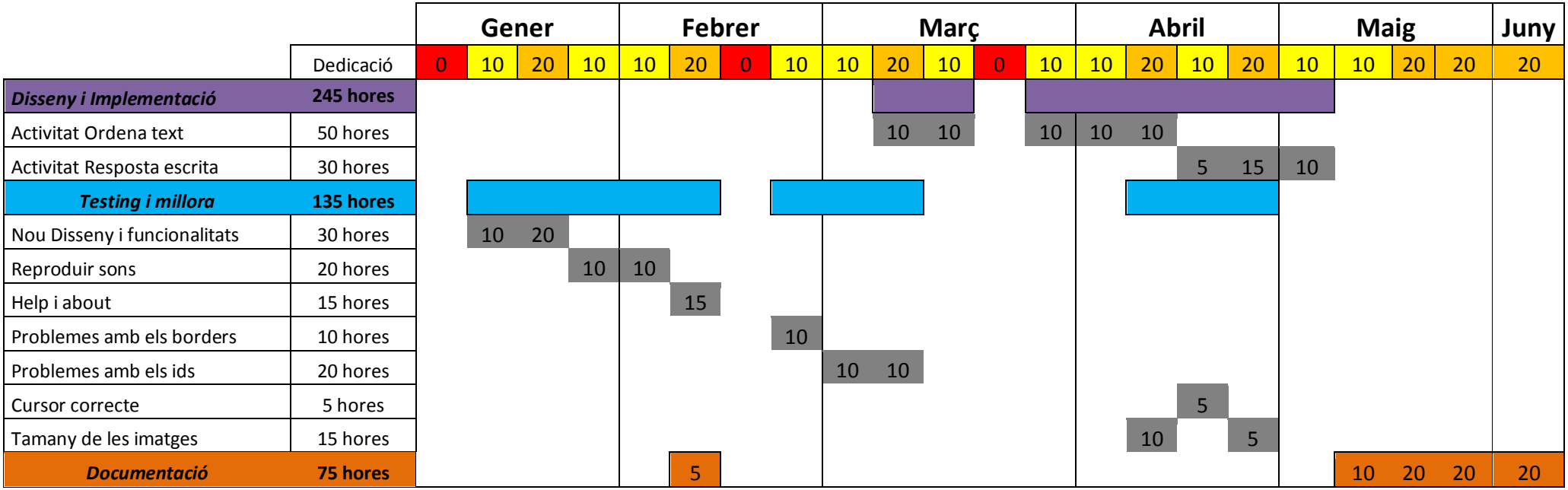
Si sumem la dedicació de les hores veurem que hem estimat un total de 530 dedicades al projecte. Contem que el testing i millora és una activitat constant i que, tot i que és variable segons el volum de feina i a les tasques que ens assignen altres projectistes, hem estimat unes 10 hores a la setmana. Com ja hem dit anteriorment ha estat un projecte dinàmic, i per tant ens hi ha hagut moments que ens hem hagut d'adaptar a les circumstàncies, de tal manera que a la planificació inicial no contàvem en que ja tindríem tota una arquitectura per realitzar les activitats i ens hauríem d'adaptar a ella. Això fa que aquesta planificació inicial s'hagi fet de mica en mica i adaptant-la abans de realitzar cada una de les tasques.

8.2.2 Planificació real

El següent Gantt mostra el treball real aproximat que hem realitzat durant el projecte. És similar a l'anterior, tot i que hem detallat una mica més les etapes i les hores, i hem desglossat les tasques de testing i millora.

Tot i posar les hores que s'han dedicat a cada subetapa del testing, s'ha de tenir en compte que les hores que hem invertit per buscar i trobar els errors estan incloses en el còmput de la setmana. També cal dir que amb el testing hem trobat errors que posteriorment es poden haver assignat a altres projectistes, clar exemple de tasques de testing que no desemboquen a cap activitat feta per nosaltres.


Juliol-Agost
Setembre
Octubre
Novembre
Desembre
Dedicació
20 20 20 20 0 10 10 10 10 20 20 10 10 20 20 10 10 10 20 10 10 0
Definició del projecte 15 hores
10 5
Aprenentatge 90 hores
Aprenentatge Python-pygame 30 hores
10 20
Aprenentatge OLPC 40 hores
20 20
Aprenentatge Jclic 10 hores
10
Aprenentatge arquitectura 10 hores
10
Disseny i Implementació 245 hores
Protocol pygame-pygtk 35 hores
10 5 10 10
Iniciar el reproductor 30 hores
10 20
Activitat Associació simple 50 hores
10 20 20
Activitat Associació complexa 50 hores
10 10 10 20
Testing i millora 135 hores
Text en vàries línies 15 hores
10 5
Cursor OLPC 5 hores
5


Gener
Febrer
Març
Abril
Maig
Juny
Dedicació
0 10 20 10 10 20 0 10 10 20 10 0 10 10 20 10 20 10 10 20 20 20
Disseny i Implementació 245 hores
Activitat Ordena text 50 hores
Activitat Resposta escrita 30 hores
Testing i millora 135 hores
Nou Disseny i funcionalitats 30 hores
Reproduir sons 20 hores
Help i about 15 hores
Problemes amb els borders 10 hores
Problemes amb els ids 20 hores
Cursor correcte 5 hores
Tamany de les imatges 15 hores
Documentació 75 hores
10 10 10 10 10
5 15 10
10 20
10 10
15
10
10 10
5
10 5
5
10 20 20 20

8.2.3 Seguiment de les tasques

En aquest apartat expliquem la nostra dedicació a les tasques realitzades, i les que ens han reportat un volum de feina diferent a l'esperat, comparem el volum d'hores o setmanes de l'estimació al temps real invertit :

- Definició del projecte: Tasques de definició i planificació dels objectius.

  Temps esperat: 2 setmanes

  Temps real: 3 setmanes, en un total de 15 hores. Podem dir que hem estat 10 hores a l'estiu i 5 mes després al setembre.

  S'ha afegit una setmana més en la planificació ja que, com hem dit abans, els objectius s'han hagut de reajustar al avançar el projecte.

- Aprenentatge de l'entorn:
  - Aprenentatge Pygame-OLPC:

    Aquesta tasca estava planificada per fer-la durant l'estiu. S'ha fet irregularment durant els mesos de juliol i agost però s'ha arribat al resultat esperat. Al no saber res de l'entorn de l'OLPC i del llenguatge de programació de Python i pygame, aquest aprenentatge ha estat una tasca difícil. Hi hem invertit un total de 70 hores.

  - Aprenentatge Activitats JClic

    Vam estar el temps esperat, 10 hores.

  - Aprenentatge arquitectura SugarClic:

    Pel que fa a l'aprenentatge de l'arquitectura d'activitats era de fàcil comprensió i adaptació, per això s'ha tardat poc. Aquesta tasca ens ha reportat un volum de feina de 10 hores.

- Disseny i implementació
  - Interfície Pygame i comunicació amb PyGTK

    Temps esperat: 20 hores

    Temps real: 35 hores

A priori, semblava una tasca fàcil haver d'adaptar les dues interfícies, però es va esdevenir més complicat de l'esperat degut a la poca compatibilitat entre els llenguatges. Tot i que s'ha desenvolupat amb un altre projectista, aquesta ha sigut el que més volum de feina ha tingut, nosaltres hem comptabilitzat que, per la nostra part, hem aportat un total de 35 hores en aquesta tasca entre disseny i implementació en front de les 20 hores que havíem estimat.

- Iniciar el reproductor d'activitats

  Aquesta és una tasca que va lligada molt amb l'anterior i que surt de l'aprenentatge del Pygame i l'OLPC, ja que des de l'activitat de prova es va començar a construir el reproductor, que posteriorment passaria a mans d'un altre projectista que acabaria la tasca. Abans que un dels altres projectistes finalitzés aquesta tasca nosaltres li vam dedicar unes 30 hores.

- Realització d'activitats:

  - Activitat Associació simple

    Temps estimat: 40 hores

    Temps real: 50 hores.

    Tot i ser més simple que l'associació complexa, al ser la primera activitat que hem desenvolupat, ens ha costat més de dissenyar i implementar: un total de 50 hores.

  - Activitat Associació complexa

    Hem estat el temps esperat, 50 hores.

  - Activitat Ordena Text

    L'activitat Ordena Text ens ha ocupat 50 hores del nostre temps degut a que també hem ens hem hagut d'adaptar i hem hagut de modificar l'arquitectura de les activitats de text.

  - Activitat Resposta escrita

    Temps esperat: 40 hores

    Temps invertit: 30 hores

El temps invertit en la realització d'aquesta activitat ha estat menor del que esperàvem degut a les seves similituds amb altres activitats i a que era més senzill del que preveiem en un principi.

- Tasques de testing i millora de l'aplicatiu

  Aquesta tasca ha estat contínua, és a dir, que des de la finalització de les primeres activitats de Associació Simple i Associació Complexa s'han estat realitzant aquests tipus de tasques. Això si, ens hem adaptat al volum de feina que teníem, per tant la dedicació ha variat al llarg de les setmanes. Això sí, en total hem computat unes 135 hores dedicades realitzant les correccions i millores, però també fent les proves corresponents.

- Documentació: Realització de la memòria i la presentació del projecte.

  Temps esperat: 60 hores

  Temps invertit: 75 hores

  No esperàvem que la documentació ens donés tant volum de feina i per això hem hagut d'invertir-hi més hores, un total de 75 hores.

La dedicació del projecte ha estat de 20 setmanes de 10 hores (un total de 200 hores) i 18 setmanes de 20 hores (un total de 360 hores) , la resta les hem comptabilitzat com a festives. Per tant, la dedicació total ha estat de 200 + 360 = **560 hores**.

Per tant, ens hem desviat unes 30 hores del plantejament inicial. Hem hagut d'afegir respecte la planificació inicial 15 hores de la interfície de pygame i pygtk, 10 hores de l'activitat Associació Simple, 5 hores més de Definició del Projecte, 15 hores més de Documentació, i per altra banda treure 10 hores menys de l'activitat Resposta Escrita i 5 hores menys de Testing.

8.3 VALORACIÓ ECONÒMICA

Per valorar el treball humà d'aquest projecte hem dividit en el treball que s'ha fet en quatre rols.

- **El cap de projecte:** L'encarregat de comprovar que es compleixen tots els requisits que s'han exigit, i controlar que l'aplicació avanci de manera correcta. Aquest rol l'ha dut a

terme el nostre director del Projecte. Hem estimat aproximadament un 10% de les hores que hem fet nosaltres, per tant, unes 56 hores.

Els següents tres rols, que els hem dut a terme nosaltres, són:

- **L'analista:** L'encarregat de fer l'etapa d'especificació i disseny del sistema.
- **El programador**: L'encarregat de la implementació de l'aplicació.
- **El dissenyador:** L'encarregat de decidir i analitzar els temes relatius amb la correcta visualització de la interfície gràfica.

Rol	Hores	Preu/Hora (€/h)	Cost (€)
Cap de projecte	56	55	3.080,00 €
Analista	180	35	6.300,00 €
Dissenyador	50	20	1.000,00 €
Programador	330	25	8.250,00 €
Total	**616**		**18.630,00 €**

En el testing hi intervindrien l'analista, el programador i el dissenyador. Pel que fa a la documentació del projecte principalment la realitzaria l'analista, ja és l'encarregat de plasmar el disseny de l'aplicació tot i que els programadors participarien en algunes parts.

Pel que fa el cost del hardware i el software, podem dir que no ens ha costat res, ja que ja disposàvem d'un equip adequat per realitzar el projecte i, tot i que hem utilitzat un portàtil XO per fet proves, aquest ens l'ha deixat en Martin Langhoff, és a dir que ha tingut un cost de 0€ per a nosaltres.

Per tant, el cost total del projecte l'hem calculat en **18.630 €.**

9. Conclusions

9.1 Objectius assolits

El plantejament inicial del projecte global tenia un objectiu comú clar: mitigar la falta de continguts disponibles per a OLPC adaptant-ne el projecte educatiu JClic. Aquest objectiu ha estat completat amb les aportacions de tots els membres. Tenim totes les activitats possibles del JClic traduïdes i funcionant correctament.

Pel que fa als nostres objectius dins d'aquest projecte, podem dir que els hem completat satisfactòriament:

- Contribuir a la realització d'una interfície de comunicació entre dos llenguatges com són Python i pygame.
- Desenvolupar activitats per al SugarClic, més concretament fer l'adaptació de les activitats clic fetes per al JClic següents:
    - Associació Simple
    - Associació Complexa
    - Ordenar Text
    - Resposta escrita
- Solucionar possibles problemes que pugui tenir el projecte del SugarClic i dotar-lo de millores que puguin influir en la seva correcta visualització.

Aquest últim objectiu, tot i sembla des d'un principi un pèl subjectiu, podem arribar a la conclusió que amb el considerable nombre de contribucions fetes tant individualment com a nivell de grup hem aportat millores significatives pel que fa a la visualització.

9.2 Treball per fer

Encara que el treball ha adquirit els objectius que es van marcar al principi, encara faltaria una tasca important: la de prova per part dels usuaris finals del SugarClic i un manteniment per tal de que aquests usuaris poguessin comunicar els errors trobats i acabar de solucionar els errors i millores que encara faltarien.

Ens han quedat un alguns punts pendents del projecte que no hem pogut solucionar del tot:

- Millorar el rendiment en diferents parts del SugarClic

- La reproducció de sons. Tot i deixar-ho preparat en els casos d'encert i fallada, amb l'OLPC això no funciona del tot bé, ja que si ho provem amb l'Ubuntu això funciona correctament.
- Una cosa que també s'hauria de mirar en un futur seria una adaptació directa de les activitats .jclic. Primerament considerar el tamany dels fitxer que aquest conté, ja que l'aplicació és per a un entorn on existeix una velocitat de connexió limitada. Un altre punt relatiu a aquesta adaptació seria la traducció directa dels fitxers *.jclic*, ja que aquestes s'han hagut d'adaptar una a una per errors en cas d'accents.

9.3 Valoració personal

Tot i que els objectius s'han vist completats per tots els membres del grup, ara toca una part important: la utilització d'aquest projecte per part dels nens i nenes que gaudeixin d'un OLPC.
Crec que parlo per tots els que formem part d'aquest projecte quan dic que sincerament tenim moltes esperances que aquest projecte s'integri el més aviat possible i esperem que contribueixi en la mesura del que sigui possible en l'educació d'aquests nens i nenes.

Gràcies a aquest projecte, a part dels objectius, també he pogut satisfer les meves motivacions personals:

- Ser l'integrant d'un projecte que busca fer una aportació en el món de l'OLPC i aportar així una aplicació o eina educativa per a nens i nenes de països en desenvolupament.
- Formar part d'un projecte dinàmic de codi obert i ubicat a Internet.

Per això vull donar les gràcies tant al director del projecte, en Marc Alier, com a tots els projectistes amb els que he coincidit i que m'han ajudat quan ho necessitava: en Jose Camallonga, en Carlos Castilla i en Marc Benito.

Aquest projecte ha sigut un repte tecnològic força gran per a mi, ja que en un principi no coneixia les tecnologies que envolten tot el món de l'OLPC i el JClic ni la seva envergadura com a projectes. L'únic que havia vist una mica d'aquest projecte és la tecnologia de l'XML, tota la resta (el sistema Sugar, els llenguatges de programació de python i pygame, etc.) han reportat una tasca de recerca d'informació i aprenentatge.

També ha resultat una feina costosa però gratificant posar-nos d'acord en la primera part del projecte per fer la tasca del protocol d'intercanvi d'informació entre les interfícies. Per això s'han hagut de comunicar dos llenguatges de programació que aparentment eren similars però que després hem vist que ha dut més treball del que esperàvem. D'aquesta manera cadascú ha hagut d'aportar part de la feina per a posteriorment avançar de forma més eficient en el projecte.

Com a conclusió final, considero que el projecte de final de carrera és una assignatura essencial que aporta una experiència molt valuosa per a qualsevol enginyer, per l'esforç i dedicació que això suposa i pel fet enriquidor de veure com aplicant els coneixements que has rebut durant tota la carrera per fi els pots aplicar per fer quelcom on podem observar resultats tangibles.

10. Bibliografia

- Sobre l'OLPC
  - www.es.wikipedia.org
  - http://laptop.org/en/
  - wiki.laptop.org/go/
  - http://www-static.laptop.org/es//index.shtml

- Sobre el JClic:
  - clic.xtec.cat/es/jclic
  - http://projectes.lafarga.cat/projects/jclic
  - www.argenclic.com.ar

- Sobre python i pygame:
  - http://www.python.com
  - http://www.pygame.org

11. ANNEXOS

ANNEX A: MANUAL D'ÚS DEL SUGARCLIC

En aquest annex s'explica com interactuar amb el SugarClic. Aquest annex no és res més que una guia per l'usuari que vulgui utilitzar l'aplicació.

Després d'instal·lar el SugarClic tindrem al nostre Escriptori un botó nou amb la forma d'un ratolí que si cliquem iniciarem el SugarClic. La primera pantalla que ens sortirà és la següent:

Figura 32: Pantalla d'inici

Aquí podrem escollir que desitgem fer, tenim 5 opcions:

- MANUAL: És un clic senzill perquè els usuaris aprenguin a utilitzar el SugarClic
- ABOUT SUGARCLIC: Petit clic on hi ha la informació relativa al projecte SugarClic
- DOWNLOAD CLICS: On podrem descarregar-nos els clics per internet al nostre OLPC.
- MY CLICS: On podrem veure els clics que tenim instal·lats al nostre OLPC.
- USB: On podrem manipular la informació relativa als clics de l'OLPC per tal de passar-los a una unitat d'emmagatzematge via USB.

A continuació descriurem com hem de descarregar els Clics i jugar amb ells:

Si a la pantalla d'inici cliquem al botó de DOWNLOAD CLICS, s'obrirà la següent pantalla:

Figura 33: Pantalla d'inici de DOWNLOAD CLICS

En aquesta pantalla podem veure el menú de sobre on tenim els botons de HOME PAGE, que ens portarà a la pantalla d'inici del DOWNLOAD CLICS (és a dir, a la que estem ara mateix) i el botó MAIN MENU, que ens portarà a la pantalla d'inici principal (Figura 31).
A la resta de pantalles podem veure una espècie de navegador al que hi ha la pàgina web on podrem descarregar-nos els clics o baixar-nos les últimes versions del SugarClic.

Al menú de la dreta podem clicar sobre diferents vincles:

- Inicio: per tornar a la pantalla principal (figura 2)
- Galeria de clics: para veure els clics disponibles y descarregar-los
- Download: per veure les últimes versions del SugarClic.

Per veure els Clic haurem de prémer sobre el vincle del menú de la dreta amb el text “Galeria de Clics”.

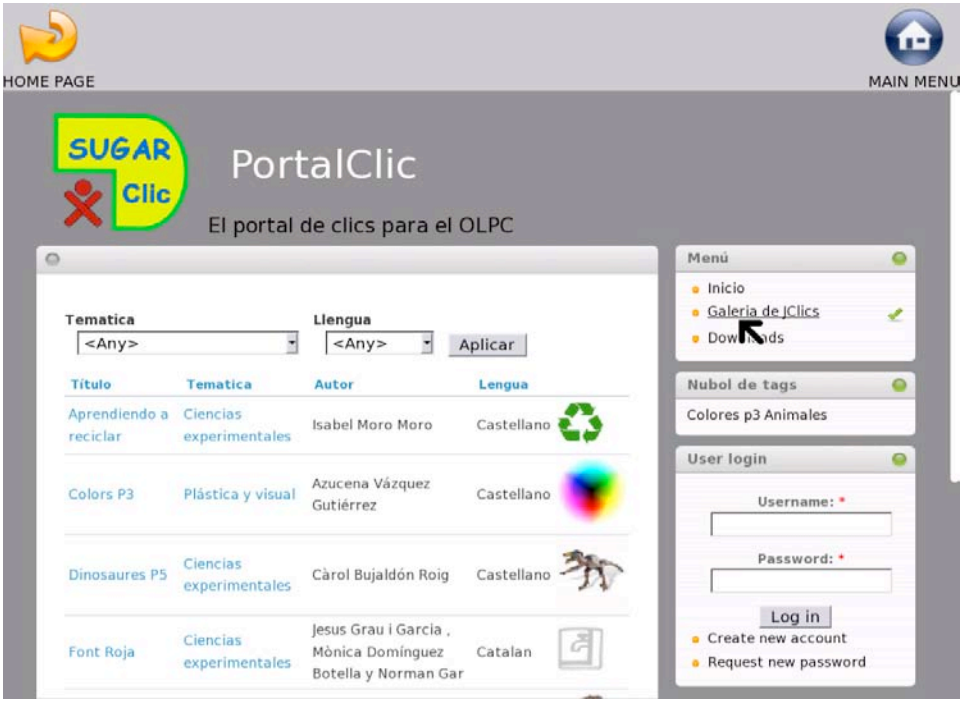


Figura 34: Pantalla de Galeria de clics

Aquí podem veure tots els clics o filtrar-los per les opcions que volguem: tipus de temàtica i tipus de llengua.

Per veure les característiques del qualsevol Clic i, si volem descarregar-lo posteriorment, haurem de clicar sobre el nom i ens sortirà una pantalla com la següent.

Figura 35: Pantalla de descàrrega de clics.

Si baixem la barra de la dreta podrem veure més detalls del clic i podem descarregar-lo clicant sobre el vincle de Descargar.

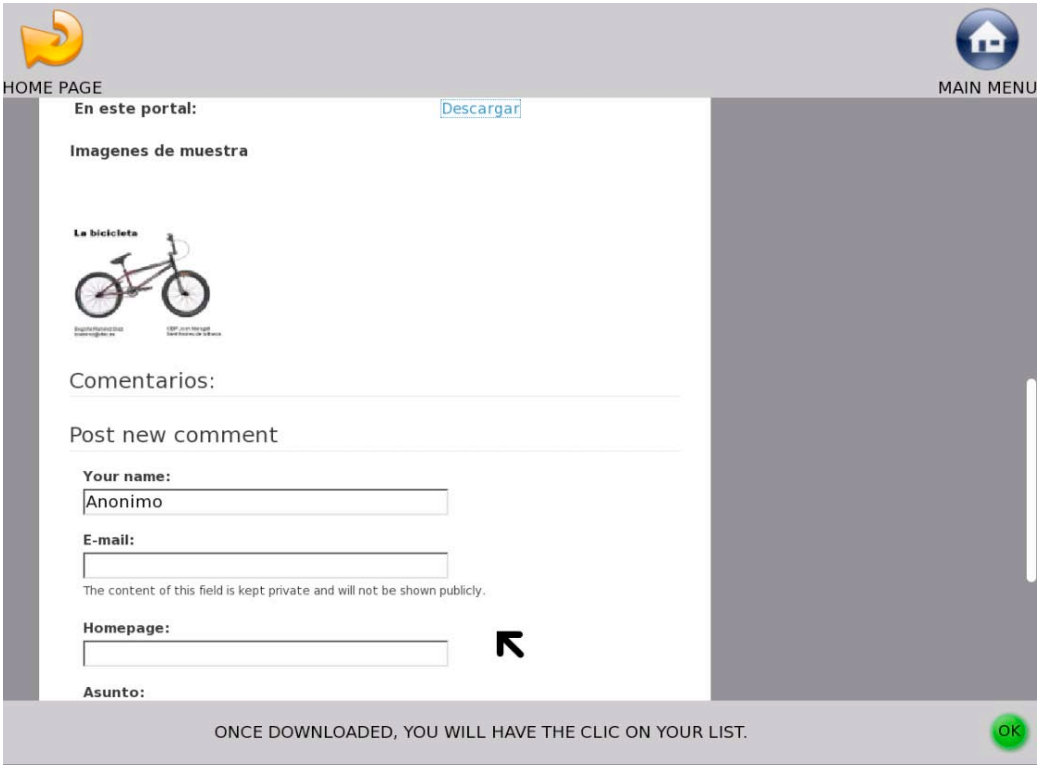


Figura 36: Pantalla de descarga del clic 2

Quan s'hagi descarregat ens avisarà a una barra a sota de la pantalla.

També podem també fer comentaris del Clic.

Per veure les últimes versions del SugarClic haurem de pressionar sobre el menú de la dreta al vincle de Download.

Figura 37: Pantalla de Downloads

Per descarregar la versió haurem de clicar sobre el vincle de la dreta amb l'extensió .xo.

Un cop haguem descarregat l'activitat podrem veure-la al menú de MY CLICS, al qual podem accedir des del menú principal (Figura 31: Pantalla d'inici) al botó del cub de Rubik. La pantalla que veurem és aquesta:

Figura 38: Pantalla de My Clics

Al menú de dalt podrem esborrar els clics o anar al menú principal (figura 1).

Per jugar a un clic farem doble clic sobre el clic que volguem visualitzar.

ACTIVITATS.

Cada un dels diferents clics està format per diferents activitats de diferents tipus. Un exemple és el següent:

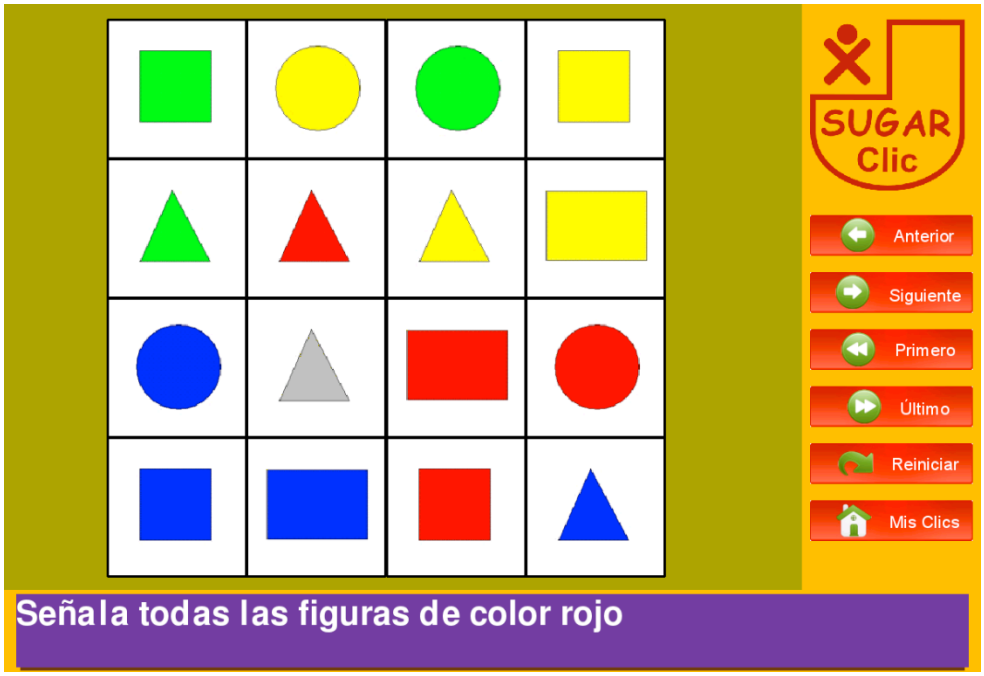


Figura 39: Pantalla de clic 1

Podem veure tres parts: el menú de la dreta, l'activitat en sí i les instruccions de sota.

En el menú de la dreta tenim les opcions dins del Clic:

- Anterior: ens portarà a la activitat anterior.
- Siguin-te: ens portarà a la següent activitat.
- Primer: ens portarà a la primera activitat.
- Último: ens portarà a la última activitat.
- Reiniciar: tornarem a començar l'activitat actual
- Mis Clics: anirem a la pantalla de MY CLICS (Figura 37).

A sota podem veure les instruccions de l'activitat actual, i a la part central l'activitat en sí.
A l'exemple actual haurem de pressionar les figures de color vermell. Després de cada clic que fem a cada imatge de figures de color vermell ho notarem perquè la imatge canviarà, en aquest cas es tornarà gris, i quan haurem resolt del tot l'activitat el missatge de sota canviarà indicant-nos que hem finalitzat l'activitat. El resultat serà el següent:

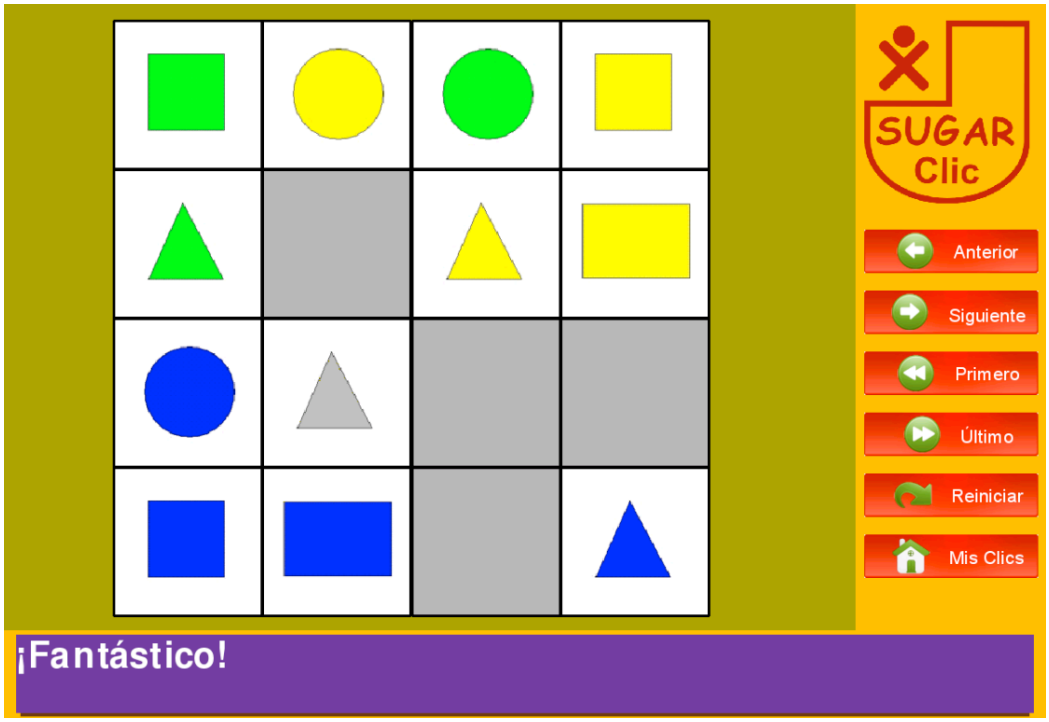


Figura 40: Pantalla de clic 2

Com hem dit anteriorment, hi ha diferents tipus d'activitat. L'exemple anterior és una activitat d'Exploració, però també hi ha altres exemples com l'activitat següent, on haurem de resoldre un trencaclosques.

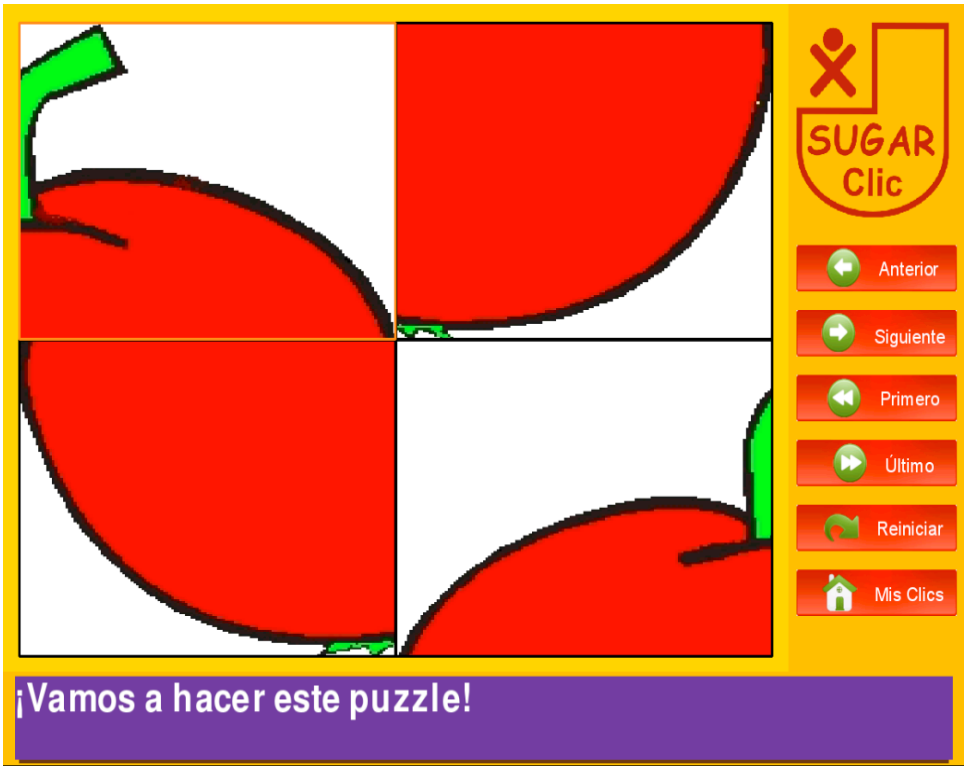


Figura 41: Pantalla de clic 3

Per resoldre'l haurem de pressionar sobre les cel·les que vulguem moure i després a la seva posició correcta.

ANNEX B: Activitats JClic

El JClic permet realitzar set tipus bàsics d'activitats, tot i que alguns d'aquests tipus presenten diverses modalitats, donant lloc a 16 possibilitats diferents. Tot seguit s'expliquen que fa cada tipus d'activitat.

Associació

Associació simple

Es presenten dos conjunts d'informació que tenen el mateix nombre d'elements. A cada element del conjunt origen correspon un i només un element del conjunt imatge.

Figura 42: Associació simple

Associació complexa

També es presenten dos conjunts d'informació, però aquests poden tenir un nombre diferent d'elements i entre ells es poden donar diversos tipus de relació: un a un, diversos a un, elements sense assignar...

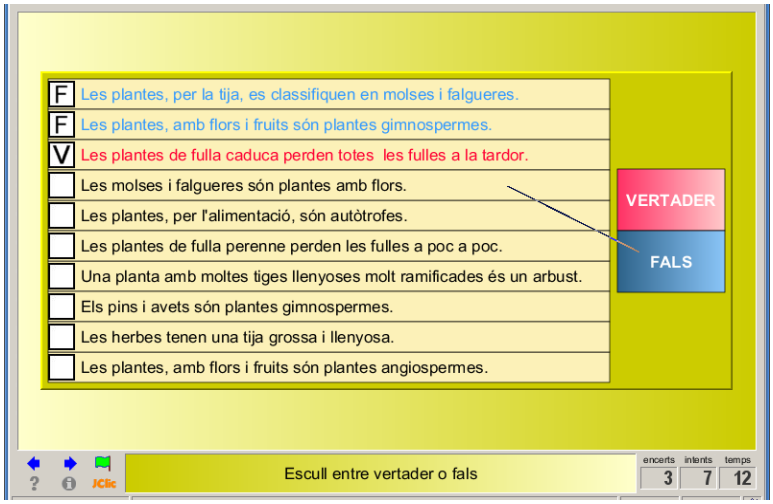


Figura 43: Associació complexa

Joc de memòria

Aquest tipus d'activitats consisteix a descobrir parelles d'elements entre un conjunt de caselles inicialment amagades. Les parelles poden estar formades per dues peces idèntiques, o per dos elements relacionats. A cada intent es destapen dues peces, que es tornen a amagar si no formen parella. L'objectiu és destapar tots els elements del plafó.

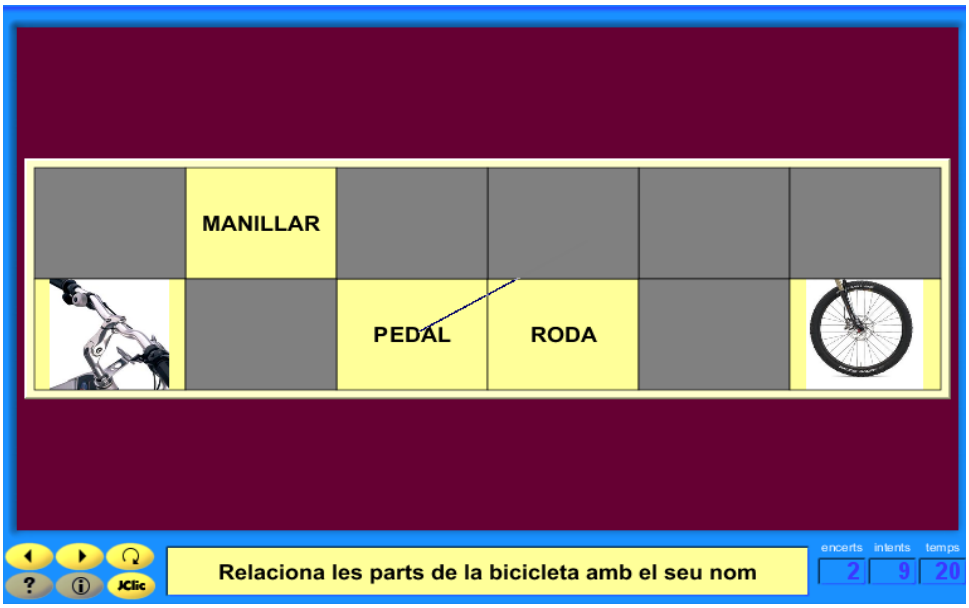


Figura 44: Joc de memòria

Activitat d'exploració

Es mostra una informació inicial i en fer clic damunt seu es mostra, per a cada element, una determinada peça d'informació.

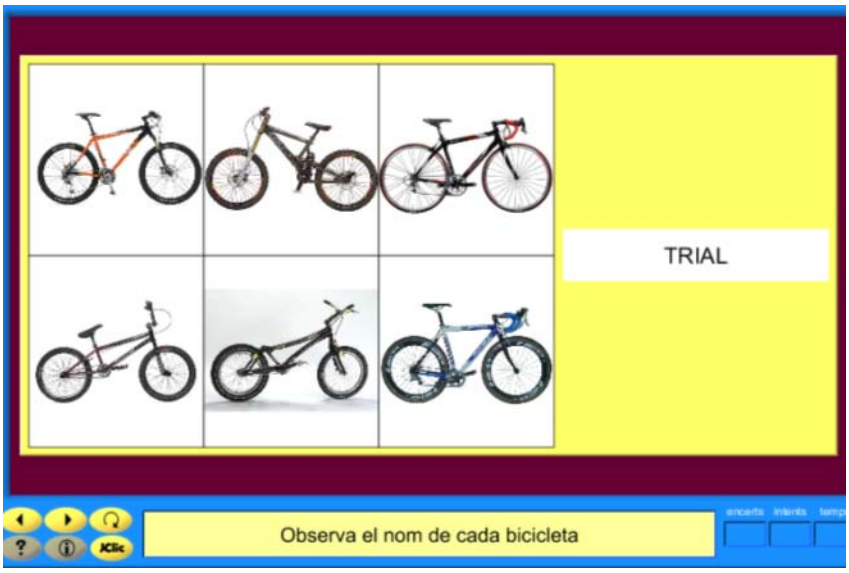


Figura 45: Activitat d'exploració

Activitat d'identificació

Es presenta només un conjunt d'informació i cal fer clic damunt d'aquells elements que acompleixin una determinada condició.

Figura 46: Activitat d'identificació

Activitat d'informació

Es mostra un conjunt d'informació i, opcionalment, s'ofereix la possibilitat d'activar el contingut multimèdia que porti cada element.

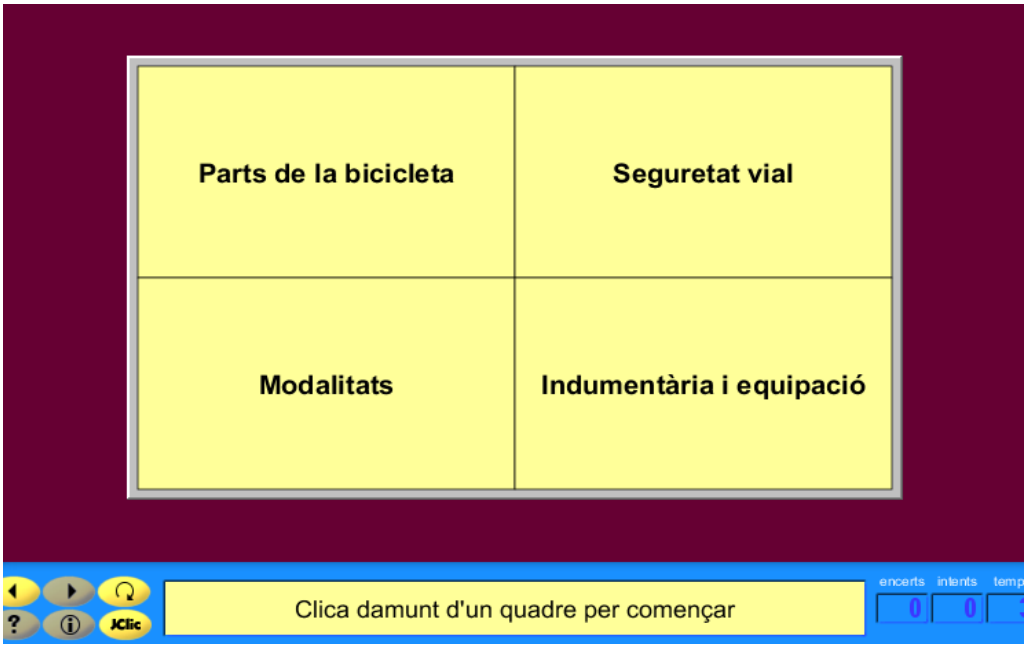


Figura 47: Activitat d'informació

Trencaclosques

Trencaclosques Doble

Es mostren dues graelles. En una hi ha la informació desordenada i l'altra està buida. Cal reconstruir l'objecte en la graella buida portant-hi les peces una per una.

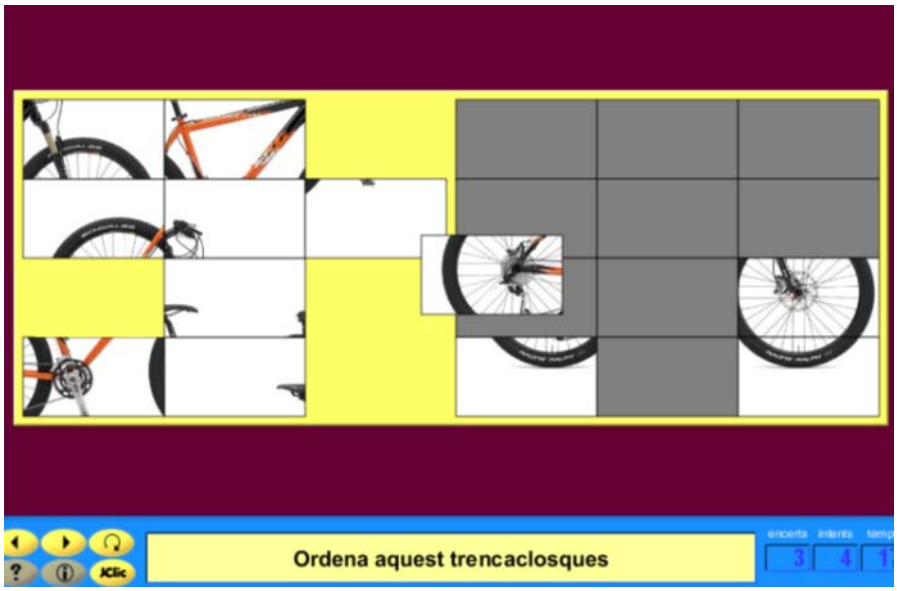


Figura 48: Trencaclosques doble

Trencaclosques d'intercanvi

En una única graella es barreja la informació. En cada tirada es commuten les posicions de dues peces fins a ordenar l'objecte.

Figura 49: Trencaclosques d'intercanvi

Trencaclosques de forat

En una única graella es fa desaparèixer una peça i es barregen les restants. A cada intent es pot desplaçar una de les peces veïnes al forat, fins tenir-les totes en l'ordre original.

Figura 50: Trencaclosques de forat

Activitat de resposta escrita

Es mostra un conjunt d'informació i, per a cadascun dels seus elements, cal escriure el text corresponent.

Figura 51: Resposta escrita

Activitats de text

Completar text

En un text es fan desaparèixer determinades parts (lletres, paraules, signes de puntuació, frases) i l'usuari ha de completar-lo.

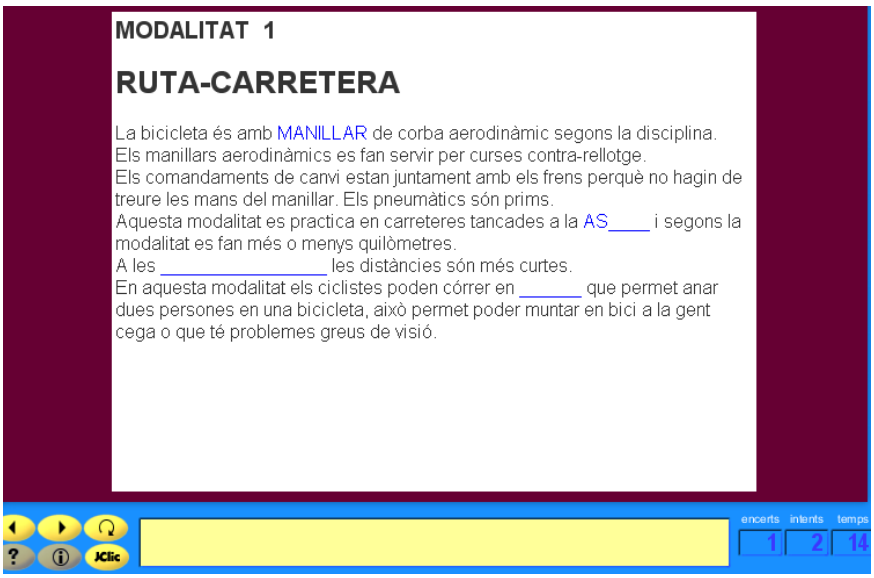


Figura 52: Completa text

Omplir forats

En un text se seleccionen determinades paraules, lletres i frases que s'amaguen o es camuflen, i l'usuari ha de completar-lo. La resolució de cadascun dels elements amagats es pot plantejar de maneres diferents: escrivint en un espai buit, corregint una expressió que conté errades, o seleccionant diverses respostes possibles d'una llista.

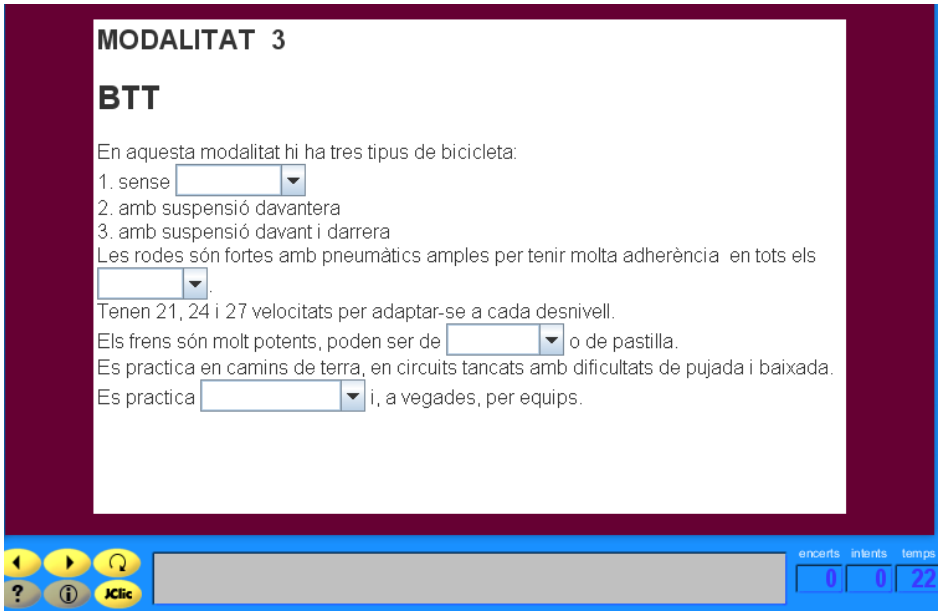


Figura 53: Omplir forats

Identificar elements

L'usuari ha d'assenyalar amb un clic de ratolí determinades paraules, lletres, xifres, símbols o signes de puntuació.

Figura 54: Identificar elements

Ordenar elements

En el moment de dissenyar l'activitat se seleccionen en el text algunes paraules o paràgrafs que es barrejaran entre si. L'usuari ha de tornar a posar-ho en ordre.

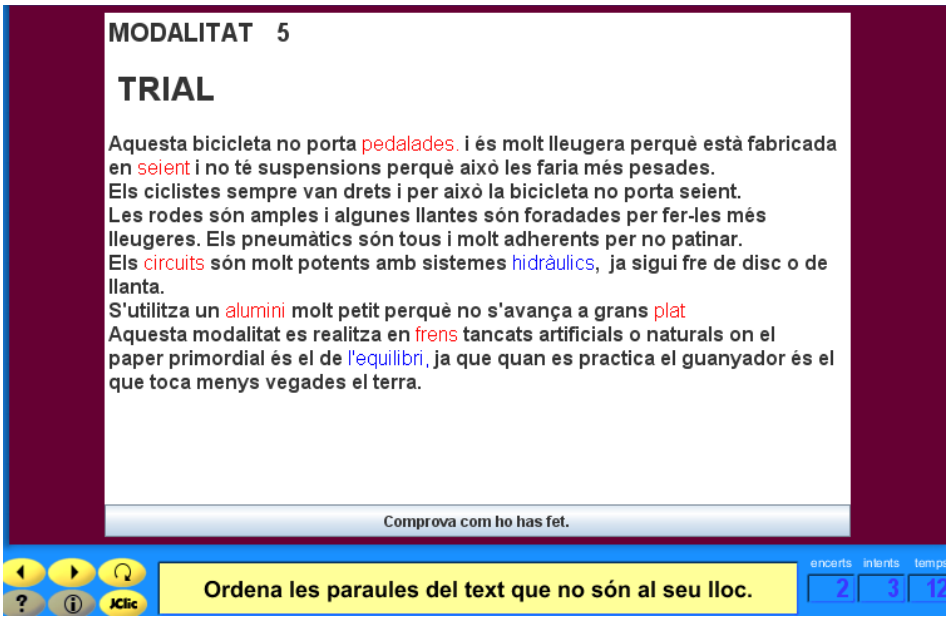


Figura 55: Ordena elements

Sopa de lletres

Cal trobar les paraules amagades en una graella de lletres. Les caselles neutres de la graella (aquelles que no pertanyen a cap paraula) s'omplen amb caràcters seleccionats a l'atzar en cada jugada. Pot ser amb contingut associat. En aquest cas s'anirà desvetllant un element d'un conjunt d'informació (text, sons, imatges o animacions) cada vegada que es localitza una paraula nova.

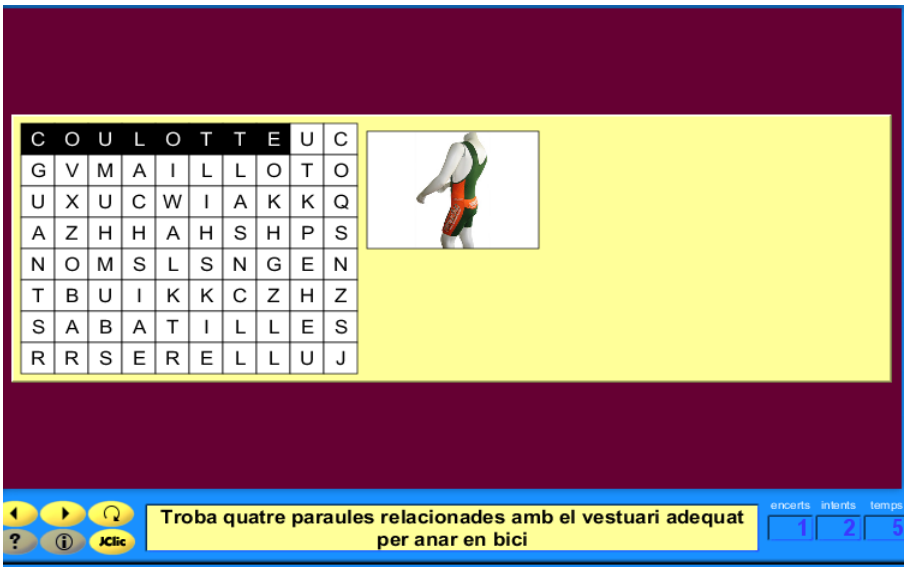


Figura 56: Sopa de lletres

Mots encreuats

Cal anar omplint el tauler de paraules a partir de les seves definicions. Les definicions poden ser textuals, gràfiques o sonores. El programa mostra automàticament les definicions de les dues paraules que es creuen en la posició on es trobi el cursor en cada moment.

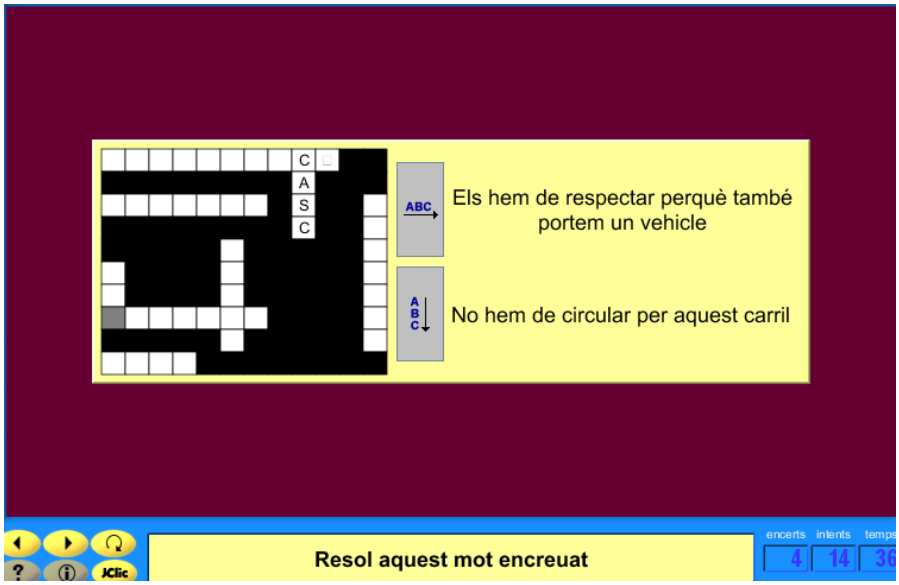


Figura 57: Mots encreuats

ANNEX C: 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 a l'educació i es va posar a treballar per assolir aquesta meta.

El primer exemple d'iniciativa d'utilització deis 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, Way 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 ordinadors 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 que fa a l'accés a les TIC rep el nom d'escletxa digital (en anglès *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 11 a estudiants de Oakar. 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,* és a dir, un ordinador portàtil, 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, Red Hat (creador del Sugar) es manifesten a favor del projecte. Diversos països com Brasil, Tailàndia, 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 OLPCs (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 gaudeixen 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 rep l'ordinador o *laptop)* esta 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 ¡que comparteixin experiències. També permet compartir coneixements entre el/s 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 a l'educació formal els jocs encara són considerats com una activitat poc seria, poden arribar a ser de gran utilitat en l'aprenentatge dels nens. Els jocs representen un gran estímul per a ells, ja que mentre juguen estan aprenent alhora.

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

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

Una educació de qualitat i un sistema d'aprenentatge per tothom són basics per aconseguir una societat justa, equitable, econòmica i socialment viable. Això esta relacionat amb els "*Objectius del*

Mil·lenni 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 busca arribar als racons més inhòspits del planeta, per tal que tot nen pugui arribar a conèixer el món de la informàtica. Aporta un granet de sorra 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. Però mentre no baixi el preu dels ordinadors portables aquest potencial l'aprenentatge queda reduït a una minoria.

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 l'escletxa 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 potencia, rapidesa o prestacions, es caracteritza per la facilitat d'interacció que té amb l'usuari, pel disseny atractiu i robust, i sobretot per la seva facilitat d'ús. Està fet així ja que és un producte enfocat totalment per als nens.

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 anterior, 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 en l'apartat de durabilitat), on els nens poguessin gaudir 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.

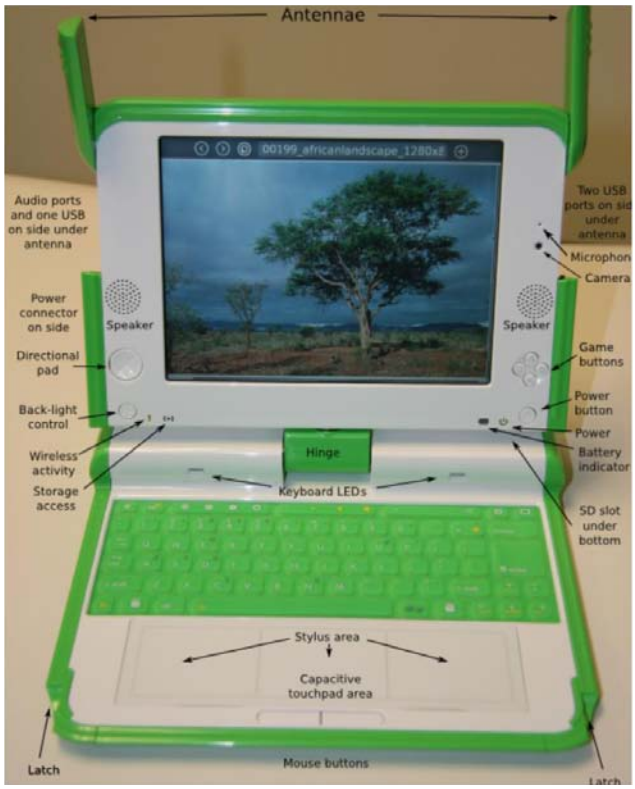


Figura 58: L'XO

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

Pel que fa al maquinari, trobem una CPU AMD Geode a 433 MHZ compatible amb les instruccions de l'Athlon i el controlador gràfic esta 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 7’5 polzades 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 802.11 b/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 carrega/descarrega) o a la corrent, 3 connectors USB i una ranura per targeta MMG/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 pot realitzar les seves pròpies aplicacions i executar-les a 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:

1. Python, amb el qual esta construïda la interfície gràfica i els diferents models d'activitats.
2. JavaScript per tal de poder executar scripts al navegador.
3. Csound, un ambient de so i audio programable.
4. Squeak, una petita versió de Smalltalk.
5. Logo, un lIenguatge d'alt nivell de molt fàcil aprenentatge, raó per la qual s'utilitza per treballar amb nens.
6. També disposa d'algun nivell de suport per Java i Flash.

Les aplicacions natives de les que disposa, enumerades de forma ràpida, son l'Xulrunner, que és el navegador d'lnternet, 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 plugins 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 la mida 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 èmfasi 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 qualitat molt destacable 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 (que traduït seria el diari), que consisteix bàsicament en un extracte de totes les activitats executades 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 algunes captures de pantalla 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 59: Pantalla principal de l'XO

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

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

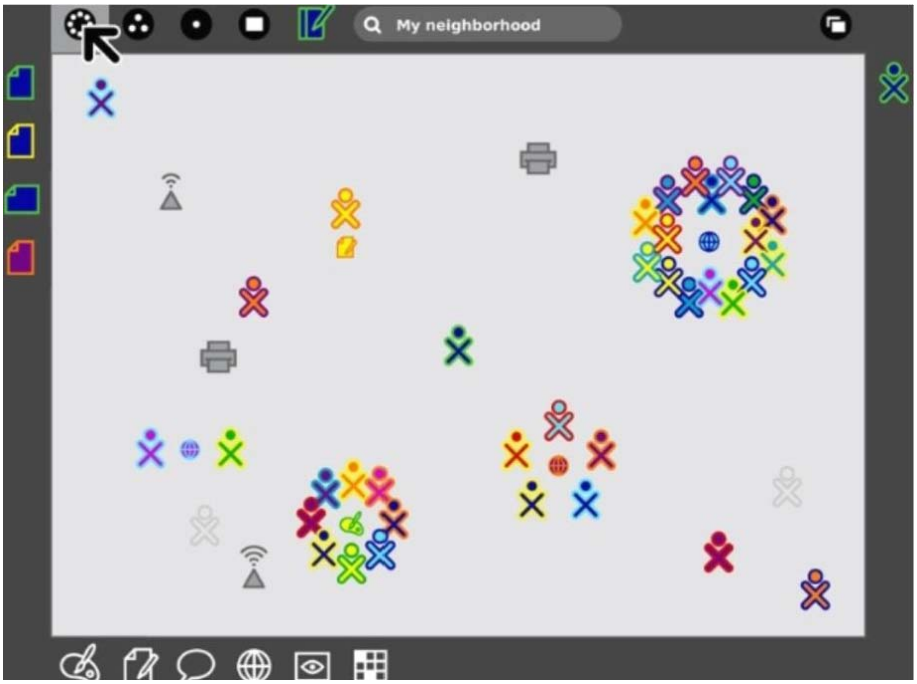


Figura 60: Pantalla de xarxa de l'XO

En la pròxima 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 navegador, ha fet fotos amb la càmera i ha xatejat amb un altre usuari.

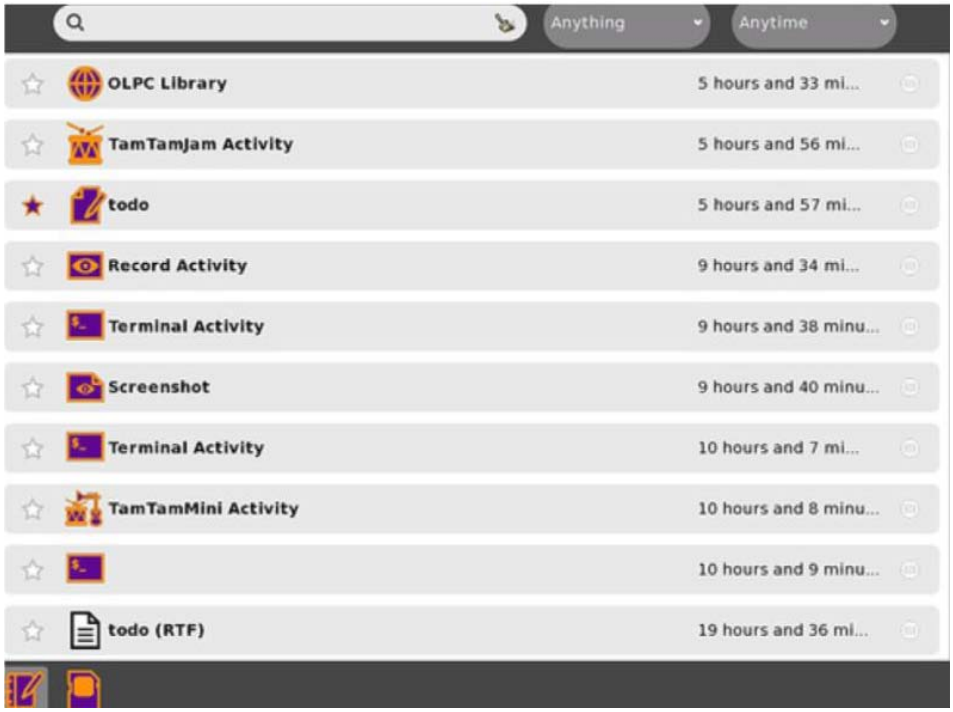


Figura 61: Journal de l'XO

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.

6. Futur

Cada dia el món avança més ràpid, i en especial el món de les tecnologies de la informació. És per això que cada cop ens trobem més nous productes i més noves tecnologies i funcionalitats que podran ser aprofitades pel projecte OLPC.

És per això que ja s'està pensant en la nova versió de l'OLPC. Com ve sent la tendència, s'està prescindint del teclat en molts dispositius per incorporar pantalles tàctils. Primerament van ser els mòbils els que van iniciar aquesta nova tendència i últimament també s'ha presentat el nou IPad d'Apple. Doncs l'OLPC també vol seguir aquesta tendència i s'està pensant en un disseny totalment renovat d'aquest portàtil i més semblant a un IPad o una llibreta que a un PC.

Figura 62: Nou prototip de XO

ANNEX D: IMPLEMENTACIÓ DE LES ACTIVITATS A PARTIR DE L'XML DEL JCLIC

Tot seguit explicarem el desenvolupament d'activitats fent èmfasi com ho hem fet i com hem utilitzat el codi que ens proporcionava l'xml del .jclic de cada activitat.

Primerament hem de modificar la classe ClicActivityHandler de tal manera que accepti les activitats que volem.

Com hem observat les activitats es defineixen entre les etiquetes <activity></activity> i s'associen a un tipus concret, com per exemple les d'Associació Simple*:*

*<activity class="**@associations.SimpleAssociation**" name="plantes01" code="">*

(codi activitat)

</activity>

Per això hem de permetre que s'executin aquest tipus d'activitats a la classe ***clicActivitiesHandler.py***, primer on es fan les comprovacions de les classes i després llançant-la de la següent manera:

```
def executeActivity(self, node, media, settings):
            if   node.getAttribute('class')  =='@associations.SimpleAssociation':
                        return SimpleAssociation(node, media, settings)
```

Posteriorment observem el comportament de l'activitat i el seu codi XML dins del JClic. Tot seguit es desglossarà per a cada una de les activitats.

Activitat Associació Simple

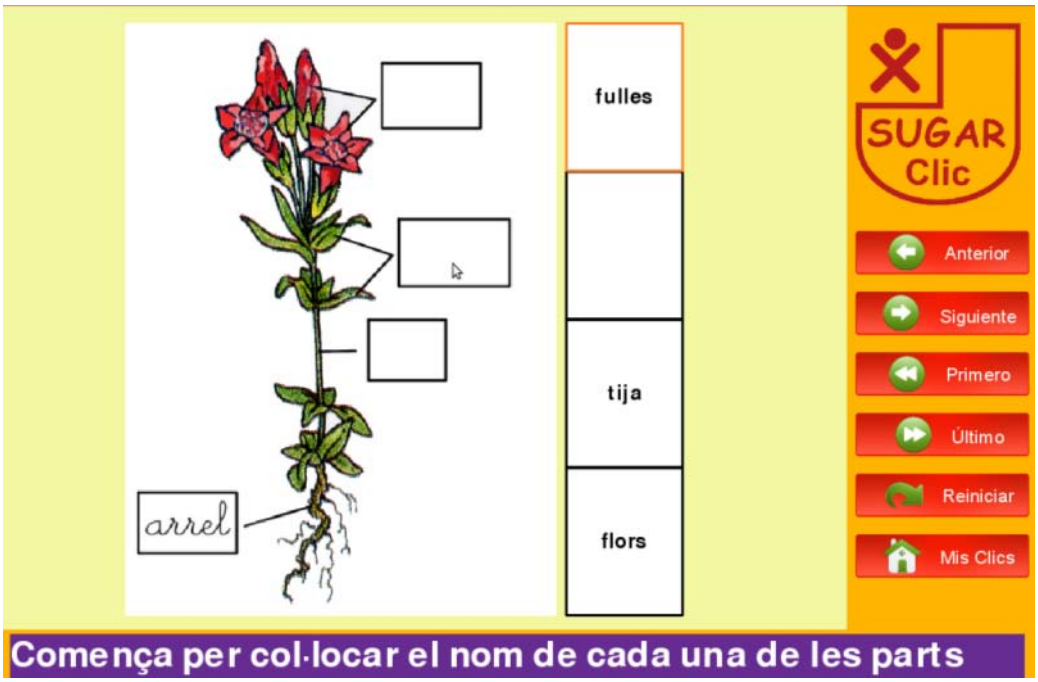


Figura 63: Activitat Associació Simple del SugarClic

Observant el codi font del .jclic trobem el següent:

Una primera graella delimitada per les etiquetes *<cells></cells>* de 4 files i 1 columna amb una imatge associada:

*<cells **rows="4" cols="1"** border="false" **image="planta01.jpg"** id="primary">*

<style/>

<shaper class="@Rectangular" cols="1" rows="4"/>

</cells>

Una segona graella també de 4 files i 1 columna amb un estil concret i amb el contingut detallat de cada cel·la, un text:

<cells rows="4" cols="1" cellWidth="112.0" cellHeight="68.0" border="true" id="secondary">

<style borderStroke="0.7" markerStroke="5.0">

<font family="Dialog" size="36"/>

<color background="0xFFFFFF"/>

</style>

<shaper class="@Rectangular" cols="1" rows="4"/>

```
<cell txtAlign="left,middle">
<p>flors</p>
</cell>
<cell txtAlign="left,middle">
<p>fulles</p>
</cell>
<cell txtAlign="left,middle">
<p>tija</p>
</cell>
<cell txtAlign="left,middle">
<p>arrel</p>
</cell>
</cells>
```

Com podem veure les cel·les anteriors no tenen cap identificador assignat, això vol dir que se li assignaran identificadors automàticament del 0 al 3 en aquest cas.

Una tercera graella de 4 files i 1 columna amb una imatge associada i amb id "*solvedPrimary*":

```
<cells rows="4" cols="1" border="true" image="planta01S.jpg" id="solvedPrimary">
<style/>
<shaper class="@Rectangular" cols="1" rows="4"/>
</cells>
```

Finalment també ens hem de fixar en aquest fragment de text al final de l'activitat:

```
<layout position="AB"/>
```

Això ens informarà si les graelles estan una a la dreta de l'altra o una a sota de l'altra.

Així doncs, segons l'XML del *.jclic* tenim dues graelles de 4 files i 1 columna, una amb la imatge *planta01.jpg* associada i l'altra amb les paraules *flors, fulles, tija i arrel* dins les seves respectives

cel·les. Quan associem correctament la part de la imatge corresponent amb la cel·la, la imatge es veurà substituïda per una altra imatge: *planta01S.jpg*, que és la mateixa imatge però amb el nom:

Figura 64: Imatges de l'activitat anterior

Activitat Associació Complexa

Com podem veure, la graella de l'esquerra és un mapa que en realitat és una imatge dividida en 9 files i 16 columnes, és a dir 144 cel·les, però que només tenen funcionalitat 15 d'elles. Això no s'havia tingut en compte en l'arquitectura d'activitats, per tant, hem hagut de fer algunes modificacions en la classe Activity.py.

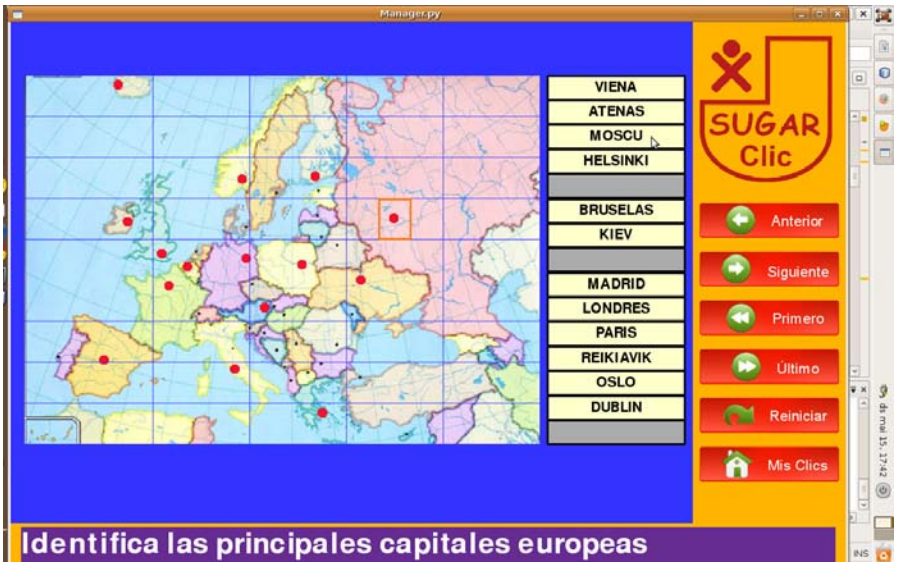


Figura 65: Activitat Associació Complexa del SugarClic

En aquesta activitat tenim el següent codi XML en el .jclic:

Primerament una graella de 9 files i 16 columnes (per evitar la còpia de tot el text hem posat "..."):

```
<cells rows="9" cols="16" border="false" image="capitales.jpg" id="primary">
<style />
<shaper class="@Rectangular" cols="16" rows="9" />
<cell />
<cell />
<cell id="4" />
<cell />
<cell />
      ......
<cell />
<cell />
<cell id="5" />
<cell />
<cell />
<cell id="6" />
<cell />
<cell />
      ...
<cell />
</cells>
```

Posteriorment una altra graella de 15 files i 1 columna amb textos on els identificadors els afegim de forma automàtica:

```
<cells rows="15" cols="1" cellWidth="134.0" cellHeight="23.0" border="true" id="secondary">
<style />
<shaper class="@Rectangular" cols="1" rows="15" />
```

```
<cell border="true">
 <style borderStroke="0.7" markerStroke="5.0">
  <color background="0xFFFFCC" inactive="0xCCFFFF" alternative="0xCCCCFF" />
 </style>
 <p>MADRID</p>
</cell>
<cell>
 <style borderStroke="0.7" markerStroke="5.0">
  <color background="0xFFFFCC" inactive="0xCCFFFF" alternative="0xCCCCFF" />
 </style>
 <p>DUBLÍN</p>
</cell>
<cell>
 <style>
  <color background="0xFFFFCC" inactive="0xCCFFFF" alternative="0xCCCCCC" />
 </style>
 <p>PARÍS</p>
</cell>
      ....
</cells>
```

També s'ha de tenir en compte l'atribut ***inverse*** de la capçalera de l'activitat. Aquest atribut ens diu quina de les dues graelles hem d'omplir amb buits i quins no:

```
<activity class="@associations.ComplexAssociation" name="ac_riosnorte" code="" inverse="true">
```

Així doncs, tenim dues graelles que hem de relacionar comprovant els ids, aquí al no haver-hi

tercera graella, omplim els les cel·les resoltes amb quadrats buits.

Activitat Ordena Text

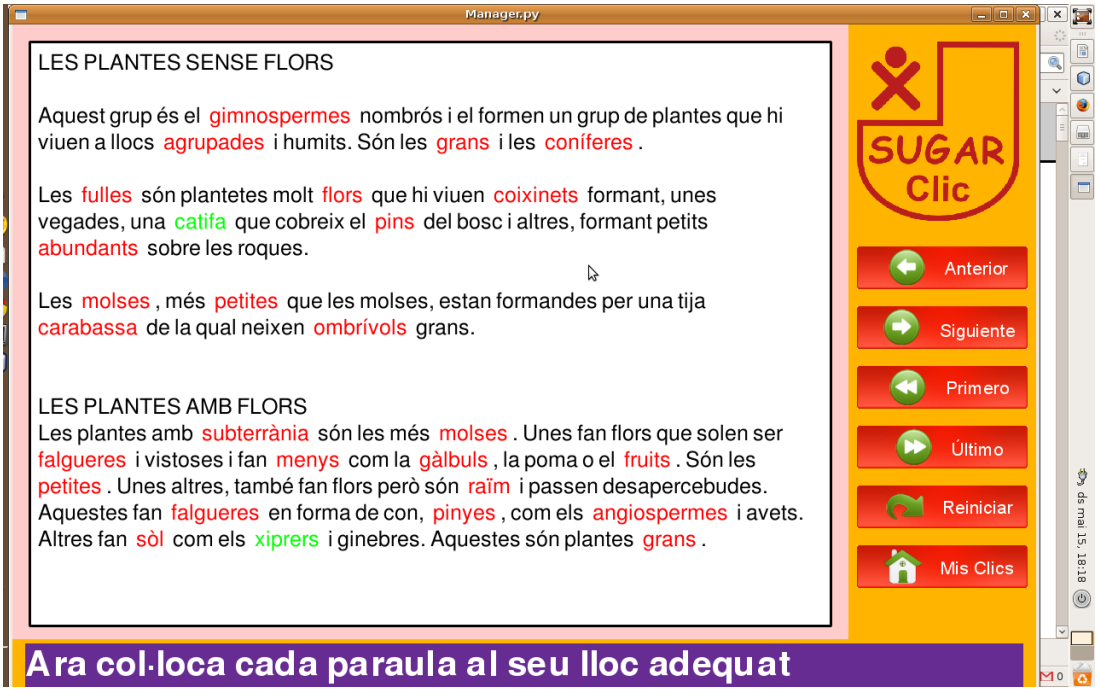


Figura 66: Activitat Ordena Text del SugarClic

Pel que fa el codi XML del .jclic, tenim el següent:

```
<document>
<style name="default" bold="false" italic="false" background="0xFFFFFF" tabWidth="12" size="14" family="Arial" />
<style name="target" base="default" foreground="0x0000FF" target="true" />
<style name="targetError" base="target" foreground="0xFF0000" />
<section>
<p background="0xFFFFFF" Alignment="1" family="Arial">
<text bold="true" foreground="0x339900" size="16">LES PLANTES SENSE FLORS</text>
</p>
<p background="0xFFFFFF" family="Arial" />
<p background="0xFFFFFF" family="Arial">
<text size="14">Aquest grup és el </text>
<target>menys</target>
<text size="14"> nombrós i el formen un grup de plantes que hi viuen a llocs </text>
```

```
    <target>ombrívols</target>
    <text size="14"> i humits. Són les </text>
    <target>molses</target>
    <text size="14"> i les </text>
    <target>falgueres</target>
    <text size="14">.</text>
   </p>
...
  </document>
```

El document en sí està delimitat per les etiquetes *<document></document>*, dins del document hi ha paràgrafs delimitats per les etiquetes * * i dins d'aquests paràgrafs cal diferenciar entre el text normal, amb les etiquetes*<text></text>* i les solucions, delimitades amb les etiquetes *<target></target>*.

Cal tenir en compte que el text poden ser tan paraules, com grups de paraules o fins i tot, paràgrafs sencers.

Activitat Resposta Escrita (WrittenAnswer)

Figura 67: Activitat Resposta Escrita del SugarClic

Similarment a les associacions, primerament una graella de 2 files i 3 columnes amb unes ids associades:

```
<cells rows="2" cols="3" border="false" image="Classif01.3.jpg" id="primary">
  <style />
  <shaper class="@Rectangular" cols="3" rows="2" />
  <ids>0 1 2 -1 -1 -1</ids>
 </cells>
```

Després una sèrie de cel·les que seran les que ens diran la solució de cada cel·la de la graella anterior. Les possibles solucions dins d'una mateixa cel·la estan separades per una barra vertical (|).

```
 <cells rows="1" cols="3" cellWidth="170.0" cellHeight="30.0" border="true" id="answers">
  <style>
   <color background="0xEAFBBF" />
  </style>
  <shaper class="@Rectangular" cols="3" rows="1" />
  <cell txtAlign="left,middle">
   <p>herba|herbes</p>
  </cell>
  <cell>
   <p>arbust|arbustos</p>
  </cell>
  <cell txtAlign="left,middle">
   <p>arbre|arbres</p>
  </cell>
 </cells>
 <cells rows="2" cols="3" border="true" image="classif01.3S.jpg" id="solvedPrimary">
  <style />
```

```
<shaper class="@Rectangular" cols="3" rows="2" />
</cells>
```

ANNEX E: Manual de desenvolupament d'activitats SugarClic

Introducció

Aquest manual pretén ser una guia per a desenvolupadors que vulguin col·laborar en el projecte SugarClic, concretament en la creació i manteniment de les activitats. Es defineixen els requisits del sistema per poder treballar i la estructura que tenen i la rel·lació amb els objectes que les composen.

SugarClic és la traducció de l'aplicació JClic que està codificada en Java, al llenguatge de programació Python fent ús de diversos mòduls. S'ha creat aquesta versió per poder ser usada a les màquines de la plataforma OLPC, les quals disposen d'una potència limitada i, per tant, és molt important l'optimització de l'eficiència.

Els projectes d'activitats jclic, anomenats clic, estan composats d'un fitxer .jclic amb format XML que conté la informació de les activitats i del projecte en sí. A més, conté tots els elements multimèdia necessaris per reproduïr els clics.

Entorn de treball

Per desenvolupar aplicacions es pot treballar amb qualsevol sistema operatiu que tingui instal·lat els programes i biblioteques que s'especifiquen a continuació, però hi ha alguns amb el que es recomana treballar per ser més fàcil la preparació de l'entorn.

La millor opció és Fedora versió 10 o 11. Aquesta distribució Linux és en la que està basat el sistema operatiu Sugar, l'usat en les màquines XO. En Fedora la majoria de mòduls ja venen instal·lats i sinó, és molt fàcil fer-ho.

Una altra opció és usar Ubuntu. Es recomana, però, la versió 8, ja que versions posteriors utilitzen versions de Python que no són les de Sugar i pot donar problemes.

També es pot desenvolupar en altres sistemes operatius, però per les activitats sempre ha de tenir instal·lats els següents mòduls:

- Python 2.5
- PyGTK
- PyGobject
- Pygame
- Sugar
- Glade 3 (herramienta de desarrollo visual de interficies graficas para GTK/GNOME)
- Hulahop (widget de PyGTK que permite incrustar un navegador Mozilla en la aplicacion)
- Gettext (librerias para crear mensajes en varios idiomas)

Tot i poder fer proves amb Ubuntu o Fedora, es recomana, però, fer-les directament en un entorn Sugar, ja sigui instal·lat de forma nativa en una màquina, a través d'un LiveCD o Sugar on a Stick (que és el mateix que un LiveCD però des d'un dispositiu USB) o emulant-lo en una màquina virtual com ara VMware, Qemu o VirtualBox [1]. D'aquesta manera s'eviten problemes de compatiblitat o falta de mòduls i a més la visualització també serà millor. S'ha de tenir en compte que no és una aplicació multiplataforma, encara que es pugui executar en altres entorns, està dissenayda per usar-la en Sugar, fent ús de funcionalitats concretes del sistema.

Estructura de classes

Tots els tipus d'activitats existents són subclasses de la classe pare Activity. Això és així perquè gran quantitat del codi necessari per implementar les activitats és comú a totes. A la classe pare es

[1] Es poden trobar instruccions de com virtualitzar Sugar al wiki de One Laptop per Child:

on s'implementen funcions d'inicialització, per mostrar text, imatges, entre altres funcionalitats que sempre són necessàries. Després cada classe que implementa una activitat crea els seus objectes concrets per representar la informació.

Hi ha una classe que pertany al reprductor en si, però que és important conèixer a l'hora d'implementar activitats perquè és la que fa de pont entre ambdues parts: és el ClicActivitiesHandler. Aquí és on es fa la importació de les classes que implementen les activitats i és des d'on es crida i la que manté la comunicació amb l'activitat: propaga els events, rep informació d'on s'ha fet clic en cas de que hagi de passar a una altra activitat o sortir, etc. En cas de crear activitats noves, s'ha d'afegir a les importacions i a la llista d'activitats disponibles (funció canExecuteActivity) i indicar la classe que l'implementa (a la funció executeActivity).

També comentar que les activitats es reprodueixen dins d'un marc que conté els botons de navegabilitat de les activitats i un espai per les instruccions de com resoldre l'activitat implementat en la classe GeneralDialog.

Una vegada sabem això, veiem el diagrama de classes i les rel·lacions que les uneixen, per saber millor com s'estructuren les activitats i els objectes que usen. Després es farà una petita descripció dels elements per saber quina funció tenen.

Activity

Aquesta és una classe pare, amb unes funcions que després totes les activitats hereden i implementen segons les accions que cada activitat hagi de fer. A més, la classe pare té algunes funcions que són comunes i útils a totes les activitats, com ara funcions per mostrar text, carregar imatges… Les subclasses que hereden de la classe Activity són totes les activitats existents. En el diagrama es mostren algunes a mode d'exemple.

http://wiki.laptop.org/

1. init(): en aquesta funció s'inicialitzen els elements genèrics de les activitats, com ara el color de fons, la informació de l'activitat i el contingut multimèdia.
2. load(): aquesta funció s'implementa en cada activitat i és l'encarregada d'inicialitzar els elements específics necessaris per representar l'activitat: grids, imatges, variables i resta d'aspectes es preparen aquí.
3. onEvent() i onKeyEvent(): són les funcions encarregades de gestionar les accions que han d'haver quan es fa clic amb el botó o bé es pica alguna tecla. Rep el punt de la pantalla on s'ha clicat o la tecla picada i respon segons això.
4. onRender(): és la funció que s'encarrega de pintar tots els elements que formen l'activitat: cel·les, textos… Aquesta funció és cridada després de cada event per plasmar els canvis que hagi ocasionat l'acció.
5. isGameFinished(): indica si el joc ha finalitzat. Aquesta funció es consultada cada vegada que hi ha un event des de clicActivitiesHandler.

Grid és un objecte que consta d'un conjunt de Cells. És una graella amb un número de files i columnes determinades que ordena la informació de les cel·les en un vector. Té operacions d'inicialització, càrrega i pintar.

Un Grid es pot inicialitzar buit i després assignar el valor de cada cel·la, o bé amb una imatge per tot el Grid. En aquest cas es divideix la imatge en el número de Cells que el composen assignant-li a cada una el seu tros d'imatge corresponent.

Cell és la unitat d'informació d'una activitat. Pot contenir text, imatge, audio o una combinació d'aquests. Concretament, aquesta informació la conté un objecte ContentCell associat a la Cell. Té funcions d'inicialització, pintar i per comprovar si s'ha clicat a sobre de la Cell.

ContentCell és un objecte sense funcions específiques que conté la informació concreta d'una Cell. La informació que es pot emmagatzemar és: un identificador, una imatge, una lletra o text i una ruta a un arxiu d'audio.

TextGrid és una graella semblant al Grid però per tractar text. Es composa d'un vector ordenat de TextCells que forma l'activitat. Les funcions que conté funcionen de forma anàloga al Grid: inicialitza, carrega i pinta els elements que el composen.

TextCell és la unitat d'informació en una activitat de tipus text. Igual que les Cells té un contingut associat, que pot ser de tipus ContentCell en el cas que sigui només text, o bé de tipus OptionList, Response o TextField.

OptionList és l'equivalent a un Select d'altres llenguatges. És una llista desplegable amb diferents opcions i que permet triar una.

Response és semblant a un camp per introduïr text però amb la diferència de que s'indiquen el número d'espais buits a omplir i pot ser que mostri la primera lletra com a pista.

TextField és un camp de text. En aquest cas no s'especifiquen els espais buits, sinó que s'ha d'omplir sense cap tipus de pista.

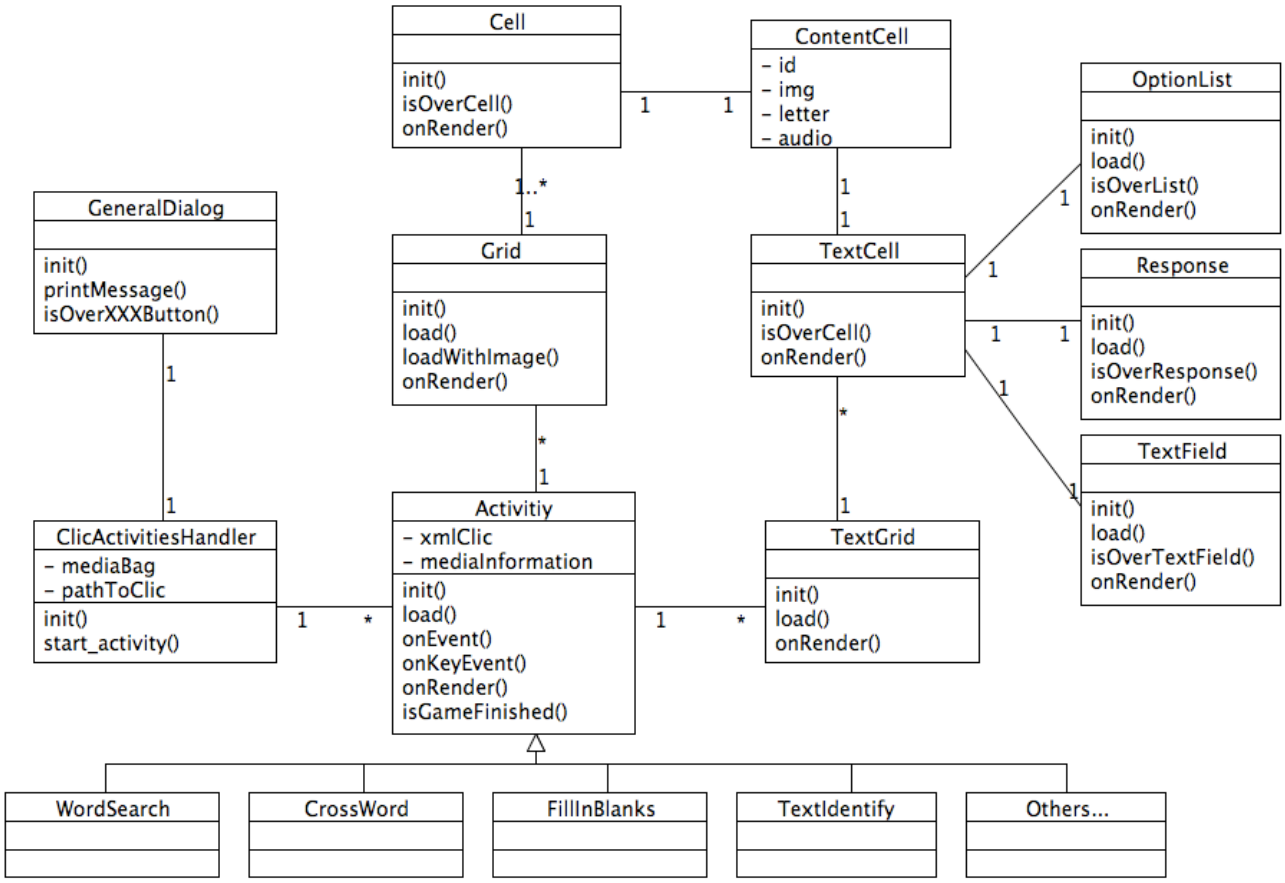


Figura 68: Arquitectura d'activitats

Les bases per implementar activitats

A continuació es farà una descripció de les accions i elements més importants sobre les activitats per tenir una guia de com implementar-les. És molt genèric ja que cada tipus d'activitat té les seves particularitats, però existeixen parts comunes.

Quan es fa la inicialització d'una activitat es rep el tros d'XML pertanyent a l'activitat que es guarda en una variable anomenada xmlActivity, i la informació del mediaBag que, mitjançant una funció cridada en el mateix procés d'inicialització, crea un diccionari amb parells (fitxer, ubicació) de manera que si accedeixes a la "posició" del diccionari "fitxer", obtens la ubicació d'aquest fitxer. En la següent imatge es pot veure un exemple d'accés al mediaBag.

```
220             pathImage =xmlcell2.getAttribute('image')
221
222             pathImage = self.mediaInformation[pathImage]
223             imagePath = self.pathToMedia+'/'+pathImage
224
225             newImg = pygame.image.load(imagePath).convert_alpha()
```

Figura 69: Exemple d'accés al mediabag

Una vegada sabem com es fa la inicialització de la informació genèrica de l'activitat, hem de tractar les operacions pròpies de cada activitat:

- Funció Load, que es on es fa la càrrega dels objectes de l'activitat. L'entorn està dissenyat de manera que tots els elements necessaris per representar l'activitat es carreguin i preparin quan en aquesta funció, de manera que després només es tracti amb Cells, ContentCells i altres objectes creats, codificant purament en Python sense necessitat d'instruccions Pygame. Potser que en algun cas, això no es compleixi al 100% per comoditat, però només en situacions esporàdiques.
    - Es rep per paràmetre la superficie de l'activitat i el primer que hem de fer és preparar-la cridant a la funció setBgColor implementada a la classe pare. Aquesta funció pinta i emmagatzema la superfície que ens caldrà per poder pintar tots els elements de l'activitat sobre ella.
    - S'ha de crear els elements necessaris per representar l'activitat. Normalment les activitats consten d'un o dos Grids amb un número de Cells determinat per la informació de l'XML. A vegades és útil tenir un Grid ocult per ajudar-nos a tenir informació que en principi, no es mostra.
- A la funció onRender s'han de pintar tots els elements que s'hagin de mostrar. Serà suficient amb cridar a la funció onRender corresponent a l'objecte a pintar, normalment sobre els Grids creats, però no s'ha d'oblidar pintar la resta d'objectes que s'hagin de mostrar, si existeixen. Si s'ha creat algun objecte ocult, lògicament sobre aquest no s'haurà de fer l'onRender.

- Funció onEvent és l'encarregada de gestionar events de ratolí. Quan el sistema rep un event "clic" de ratolí, agafa el punt on s'ha produït i el propaga a l'activitat. En aquesta funció és on es gestiona la lògica del joc. El més normal és recorrer el Grid o TextGrid per saber on s'ha fet el clic i, segons quina sigui la Cell o objecte pressionat, s'haura de fer una acció o una altra segons les regles del joc.

- Funció onKeyEvent gestiona els events de teclat. De la mateixa manera que amb els events de ratolí quan es pica una tecla, es comprova quina és i arriva a l'activitat. Normalment s'haurà de mostrar la lletra picada en un element prèviament seleccionat amb el ratolí però tenint en compte que aquesta acció pot tenir altres conseqüències.

- Funció isGameFinished és on es defineix la condició de finalització del joc. S'ha de comprovar si es compleixen els requisits per donar el joc per finalitzat, perquè el ClicActivitiesHandler s'assabenti i faci les accions pertinents.

S'ha de tenir en compte que la classe pare Activity implementa funcions útils per les activitats. Una de les més importants és la funció printxmlCellinCell. A partir d'una informació XML pertanyent a una cel·la i la Cell que es vol processar, aquesta funció intenta mostrar sobre la cel·la una imatge, un text o ambdues coses segons la informació obtinguda. A més, si la cel·la té audio, emmagatzema la informació necessària i pot ser reproduït cridant a la funció també pública a totes les activitats play_sound.

ANNEX F: Ubicació del software desenvolupat

Tot el codi desenvolupat per l'aplicació es troba a la següent direcció:

http://code.google.com/p/sugar-clic/source/checkout

Per poder accedir al codi, un cop estem en aquesta direcció hem de seleccionar la pestanya ***Browse*** i es mostrarà els següent contingut:

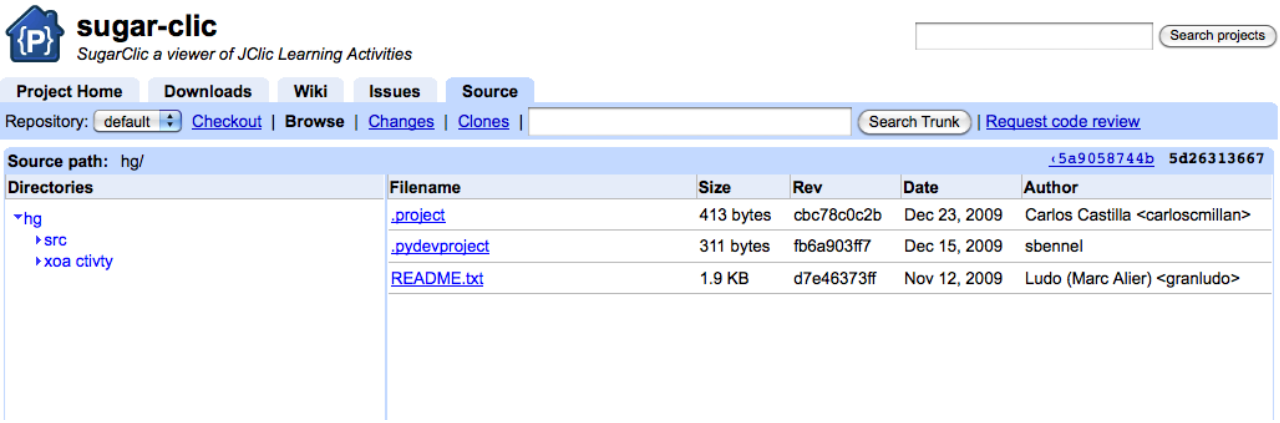


Figura 70: Navegador del codi desenvolupat

Un cop estem a l'arbre de directoris, hem de seleccionar el directori ***src*** que és on es troben tots els mòduls de l'aplicació:

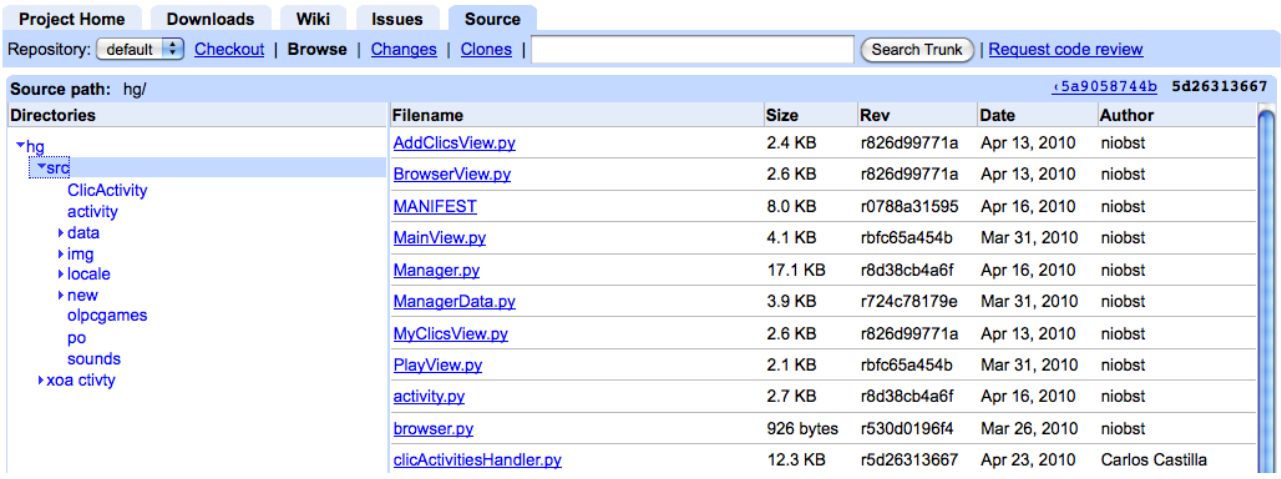
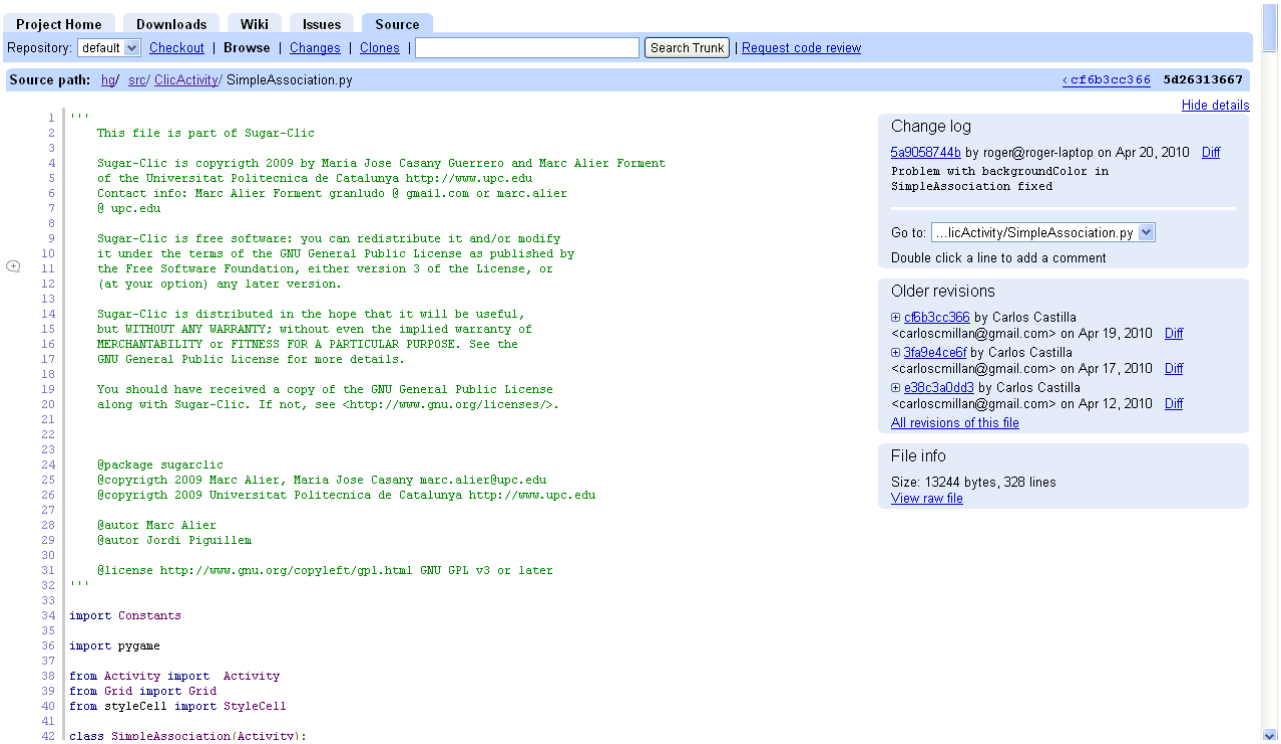


Figura 71: Mòduls de l'aplicació

Per poder veure el codi d'un mòdul, l'únic que cal fer és seleccionar un dels mòduls que apareix a la dreta de la figura anterior. El resultat és el següent:




```
'''
   This file is part of Sugar-Clic

   Sugar-Clic is copyrigth 2009 by Maria Jose Casany Guerrero and Marc Alier Forment
   of the Universitat Politecnica de Catalunya http://www.upc.edu
   Contact info: Marc Alier Forment granludo @ gmail.com or marc.alier
   @ upc.edu

   Sugar-Clic is free software: you can redistribute it and/or modify
   it under the terms of the GNU General Public License as published by
   the Free Software Foundation, either version 3 of the License, or
   (at your option) any later version.

   Sugar-Clic is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
   GNU General Public License for more details.

   You should have received a copy of the GNU General Public License
   along with Sugar-Clic. If not, see <http://www.gnu.org/licenses/>.



   @package sugarclic
   @copyrigth 2009 Marc Alier, Maria Jose Casany marc.alier@upc.edu
   @copyrigth 2009 Universitat Politecnica de Catalunya http://www.upc.edu

   @autor Marc Alier
   @autor Jordi Piguillem

   @license http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
'''

import Constants

import pygame

from Activity import  Activity
from Grid import Grid
from styleCell import StyleCell

class SimpleAssociation(Activity):
```

Figura 72: Codi de l'Associació Simple