TDesktop : Disseny i implementació d'un sistema gràfic tangible

Carles F. Julià i Daniel Gallardo Grassot

6 de setembre de 2007

Dedico aquest projecte a aquells capaços de
destruir-ho tot només per voler repensar el món:
A vegades és necessari caure per aixecar-se.
Carles
Els petits canvis són poderosos.
Dani

Resum

Noves eines comporten noves formes de treballar. Així es podria resumir la filosofia del nostre treball; la tecnologia apareix i molts cops, ja sigui per desconeixement o per hàbits adquirits, no se sap utilitzar correctament.

Fa un temps que s'han anat desenvolupant plataformes que permeten una interacció més directa amb l'ordinador: a través de les mans[15], dels gestos corporals[34] o dels objectes[23]. Aquestes plataformes encara no tenen definit un llenguatge, així com ja el tenim definit amb els WIMP en els ordinadors personals.

En els últims mesos ha esclatat la moda de les superfícies interactives, provocada en els grans èxits mediàtics de la ReacTable[21], la taula multitactil de Jeff Han[15], i posteriorment per la Microsoft Surface[28].

En aquest projecte intentem aprofitar aquestes noves tecnologies per apuntar cap a on podem evolucionar en la interacció entre l'home i l'ordinador. Intentem així trobar nous conceptes útils per a la relació diària prou coherents i intuïtius com per a ser usats habitualment.

En concret ens centrarem en la construcció d'un sistema d'escriptori, tant per la vessant més interna com per la part més explícita, que intenti aprofitar al màxim la plataforma de la ReacTable[21] per a aplicacions menys especialitzades.

El resultat és un sistema d'escriptori coherent, innovador i obert, que facilita el desenvolupament posterior d'aplicacions per a la plataforma.

Índex

Índex de figures

Índex de taules

Glossari

API Interfície de programació d'aplicacions: conjunt de declaracions que defineix el contracte d'un component informàtic amb qui farà ús dels seus serveis.

Aplicació de Control Per aplicació de control entenem una aplicació que gestiona àmbits bàsics que corresponen al sistema operatiu.

Assemblies Es el nom que obtenen les llibreries de C# un cop compilades a arxiu amb extensió dll.

Bytecode Fitxer de codi precompilat que es llegit per un interpret en temps d'execució.

Cursor Un cursor es el nom que rep la representació digital dels dit sobre la taula.

D-Bus Sistema de programari que proporciona una forma simple de comunicació entre diverses aplicacions, desenvolupat com a part del projecte freeDesktop.

Feedback Retorn del sistema a un event d'entrada generat per l'usuari.

Fiducial Figura que es col·loca sobre la taula. Es el nom que se li dona a la codificació gràfica que porta a la part inferior.

Framework Espai personalitzat i estructurat de treball.

GUI Graphic User Interface. Interfície gràfica d'usuari.

HTTP Protocol de transferència d'hipertext. Estableix un protocol per al intercanvi de documents d'hipertext i multimèdia a la web.

Hàptic Conjunt de sensacions no visuals ni auditives que experimenta un individu. Per exemple el tacte.

Kernel Nucli d'un sistema.

Metàfora En lingüística cognitiva, la metàfora conceptual es refereix a entendre un domini conceptual en termes d'un altre, entenen domini conceptual com a qualsevol organització coherent d'experiència. A vegades també es refereix com a analogia.

Model View Controller Patró de l'arquitectura del programari que separa les dades de l'aplicació, la interfície d'usuari i la lògica de control en tres components diferents.

Namespaces Espai de noms. S'utilitzen per a definir una estructura geràrquica dins de les llibreries.

Proxy Patró de disseny del programari que manté un representat d'un objecte.

RootSerializer Serialitzador de la classe arrel. Codifica a Xml un classe arrel i totes les seves agregades

Runtime Temps o moment d'execució.

Singleton Patró de disseny, Instància única. Garanteix l'existència d'una única instància per una classe i la creació d'un mecanisme d'accés global per aquesta instància.

TCP Transmission Control Protocol és un protocol orientat a la connexió dintre del nivell de transport del model OSI que permet l'entrega de datagrames de manera fiable.

Thread Fil d'execució

Token Ens referim amb token a un tangible o un objecte (físic o virtual) movible.

TUI Tangible User Interface. Interfície d'usuari tangible.

WIMP Acrònim de Window Icon Menu Pointing device. Es refereix al sistema interactiu dels escriptoris virtuals de l'actualitat.

XML Llenguatge d'etiquetatge extensible.

1. Introducció

Per Carles F. Julià i Daniel Gallardo

1.1. Sobre la memòria

Donat que aquest projecte ha estat elaborat per dues persones, hem intentat des d'un principi dividir-lo en parts i repartir-les equitativament. Aquestes parts eren:

- Creació d'un micronucli per a una interfície tangible(Daniel Gallardo Grassot).
- Estudi, implementació, i avaluació de diferents metàfores en la interacció amb menús en interfícies tangibles(Carles Fernàndez Julià).

Però finalment, les dues parts s'han anat trepitjant fins que cada membre del grup ha acabat participant-hi en ambdues.

Al llarg d'aquesta memòria, podreu trobar a l'inici de cada capítol el nom de l'autor que hi ha contribuït més, que és qui ha escrit principalment aquell tros. Tot i així cal tenir en compte que la feina ha estat molta i repartida.

Tant la part de *Context i Conceptes* com la de *Estat de l'art* són resums o es basen en el paper *New Metaphors in Tangible Desktops[22]* que vam fer en motiu de l'assignatura de Taller de Sistemes Interactius.

1.2. Context i Conceptes

1.2.1. L'escriptori com a metàfora

1.2.1.1. Escriptoris tradicionals

Abans de tot parlarem sobre els escriptoris tradicionals, de tota la vida, que es poden trobar en qualsevol casa o oficina, fets de fusta o vidre i on es treballa. Normalment, els escriptoris estan formats per les mateixes parts.

- La primera, l'àrea de treball on hi han totes les eines i on es duen a terme totes les accions.

- La segona, les carpetes, calaixos i altres mètodes d'emmagatzematge de documents.

Les eines en els escriptoris tradicionals són: llapis, bolígrafs, telefon, regle, goma, rellotge, etc .. dissenyades per escriure, dibuixar o comunicar-se.

1.2.1.2. Escriptoris virtuals

Amb l'arribada de les noves tecnologies, apareix l'oportunitat de portar aquests escriptoris tradicionals (físics) dins els ordinadors, convertint-se en virtuals. Com els tradicionals, els escriptoris virtuals estan formats per una àrea especifica de treball, on hi ha totes les carpetes, arxius i documents[9, 29].

També varies eines avançades es troben a l'abast dels usuaris, que poden realitzar un vast ventall d'accions. Les accions són les mateixes que en els seus predecessors però més desenvolupades. Els ordinadors milloren aquestes accions fent-les més eficients i rapides.

1.2.1.3. De tradicionals a virtuals

Primer de tot ens hauríem de preguntar que ens aporta aquest canvi. Be, aquesta evolució ens dona noves possibilitats, mantenint la metàfora del escriptori dins l'ordinador. El sistema de carpetes i fitxers millora la organització dels documents, fent que sigui més fàcil i rapida la seva cerca i manipulació, ja que en el sistema tradicional els usuaris poden perdre els documents quan n'hi ha un excés. En els virtuals gràcies a l'ordre estàtic en el qual estan soluciona aquest tipus de problema. A part de disposar d'un conjunt d'eines noves més avançades i útils. Com per exemple, editors de text, de so, eines avançades de comunicació, impressores, etc..

Aquest canvi va ser estudiat per Jakob Nielsen[30]: segons els seus estudis, primer en les interfícies basades en comandes, havíem de definir objectes i funcions (comandes) en una sintaxi nom/verb, com Ms-Dos o UNIX. Això canvia quan arriben les primeres GUI (Interfícies gràfiques d'usuari) on trobem les finestres, les icones, els menús, els apuntadors (WIMP), i on els objectes estan representats per icones i les comandes en menús.

Si parlem d'interacció, les GUI's i les WIMP no són del tot eficients en aquest camp. Per exemple, algunes tècniques d'interacció estan basades en complicades seqüències (clic, seleccionar, executar, deseleccionar) també algunes WIMP no tenen quasi feedback hàptic. El ratolí pot ser un exemple clar de dispositiu sense eficiència interactiva.

Nielsen també va estudiar com hauria de ser el futur de les Interfícies d'Usuari Tangibles (TUI), va dir que aquestes interfícies haurien de ser lliures de context i les accions i els objectes unificats en un sol token.

També va dir que les TUI serien molt més eficients perquè d'un sol cop, l'usuari podria agafar-manipular, en comptes de la complicada seqüència de les WIMP, que les feia més lentes i molt més feixugues.

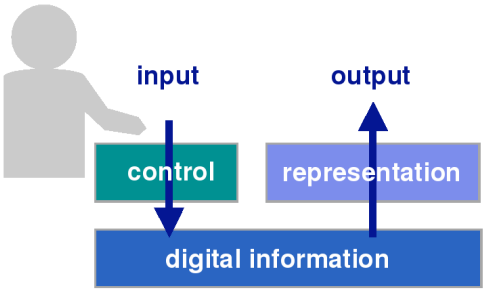


Figura 1.1.: MVC: Interfícies d'usuari.[16]

1.2.2. Repàs dels diferents esquemes Model view Controller

1.2.2.1. Interfícies d'usuari

Des de sempre, els usuaris han interactuat amb les màquines utilitzant dos canals bàsics de comunicació, un orientat a l'entrada de dades (anomenat Control) i un altre orientat a la sortida (anomenat representació). Aquests components sempre han anat fortament separats, de fet poden trobar-se en medis i localitzacions diferents[1].

1.2.2.2. Interfícies gràfiques d'usuari (GUI)

En el cas de les GUI, tornem a tenir dos canals de comunicació separats, però que posseeixen certes peculiaritats. Aquestes interfícies gràfiques, utilitzen la metàfora de l'escriptori[2] però no manipulen els objectes directament sinó que per fer-ho es serveixen d'eines externes que actuen a mode de control remot del "escriptori"[19]. Per tant el feedback tangible que aportin aquestes eines, serà nul o en el millor dels casos insuficient.

Eina externa(Acció)	Reacció
Teclat	Eina d'escriptura, la manera més directa per a transmetre text a l'ordinador.
Ratolí	Una nansa, un llapis, és la representació de la mà al escriptori virtual.
Joystiks, pads, tauletes digitalitzadores,...	Tornen a ser com el ratolí, eines de representació de moviments reals dins d'entorns virtuals.
Volants, Joystiks especials , eines electromecàniques,...	Tot i que aquestes eines poden oferir un retorn hàptic, ara per ara no hi han aplicacions que les utilitzin dins del camp de les GUI.

Taula 1.1.: Eines per al control remot de l'escriptori.

[1] Per exemple un interruptor, quan l'acciones, s'encén un llum que no té necessàriament perquè sortir del mateix interruptor.

[2] Divideixen l'espai en zones de treball, utilitzen eines, carpetes, fitxers,....

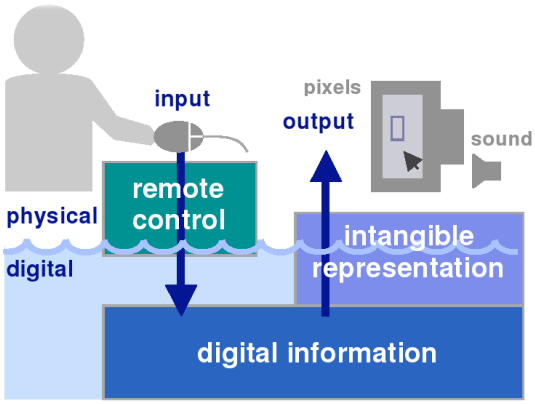


Figura 1.2.: MVC: GUI.[16]

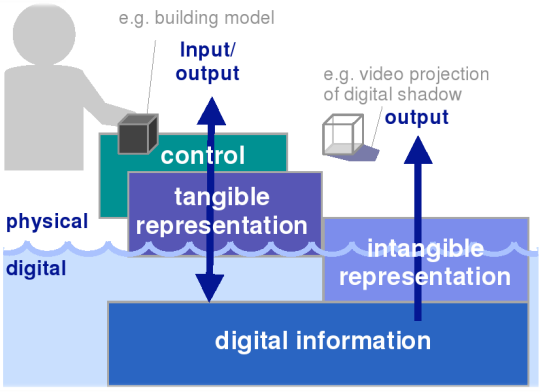


Figura 1.3.: MVC: TUI.[16]

1.2.2.3. Interfícies tangibles d'usuari (TUI)

Els canvis en aquest model, apareixen amb la fusió dels dos canals (control i representació) i amb això la metàfora de "Tu pots tocar la informació" . Ara per ara, les interfícies TUI, són les que s'apropen més a la forma d'interactuar que tenim amb les coses reals.

Els objectes, esdevenen la porta entre el mon real (tangible) i el món virtual. Però en la majoria dels casos, Els objectes es poden transformar en simples controls remots dels objectes virtuals, cosa que ens retorna a paradigma de les GUI.

Què fer per evitar-ho? ara per ara no hi ha cap norma preestablerta del que es pot fer i del que no, fet que obre una porta gegant a fer experiments d'interacció amb aquest tipus d'interfícies tangibles. Però cal remarcar que un abús excessiu d'objectes posa més entrebancs que facilitats a la comunicació amb la màquina[33, 31].

1.3. Estat de l'art

Per explicar com està el tema en el panorama de les interfícies tangibles i , en especial, els escriptoris tangibles[3], ho podem dividir en aquestes mateixes dues categories.

1.3.1. Aplicacions tangibles:

Avui en dia han saltat a l'actualitat múltiples interfícies tangibles (sembla realment una moda). Des de la primera aparició amb el *marble answering machine[3]* han sorgit moltes aplicacions específiques controlades amb interfícies tangibles[3, 10].

En la actualitat tenim per exemple la ReacTable[21] que bàsicament és un sintetitzador modular tangible amb moltes utilitats capaç de crear música en directe.

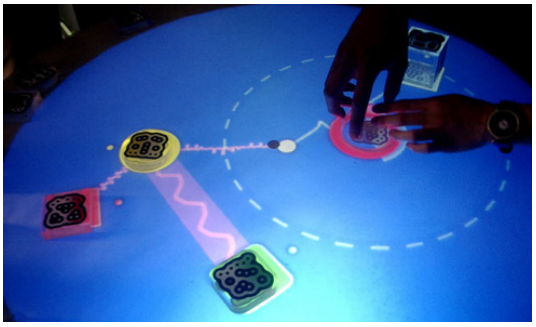


Figura 1.4.: ReacTable.

Un altre intent de instrument musical , anterior, és el Audiopad[32] pionera en la seva matèria.

Un exemple famós , però és en el camp de les interfícies multi-tàctils és el del Jefferson Y. Han[11], tota una eminència en aquest tema.

1.3.2. Escriptoris tangibles

Al començar el projecte, el més semblant a un escriptori tangible era per una banda *Panasonic interactive table* i per l'altra *mediaBlocks[36]*.

Panasonic interactive table intenta integrar WIMP en una taula interactiva tangible fent servir una mena de "metàfora de la peixera" on els programes (menjar) són executats per peixos i les seves comunicacions es fan a través de bombolles. Sembla ser que els desenvolupadors intenten aplicar el model WIMP, que sabem que és un model que funciona, combinant-lo amb una metàfora nova que es posa sobre del Wimp. La combinació sembla bastant un intent desesperat de combinar diferents tecnologies sense parar-se a pensar en el llenguatge interactiu que en pot resultar.

[3] També es treballa en extendre els esctiptoris virtuals[20, 25, 14], i al marge dels escriptoris[13].

Figura 1.5.: Audiopad.

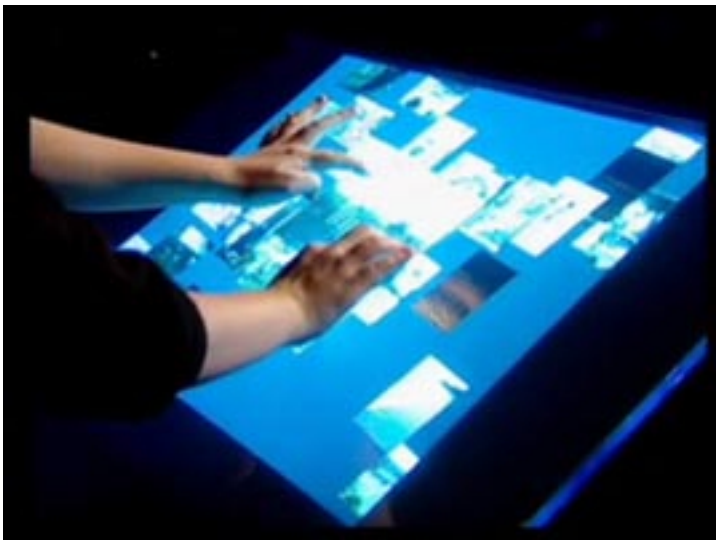


Figura 1.6.: Jeff Han's multi-touch interface.

mediaBlocks[36] intenta implementar el "fitxer tangible". En cert sentit té lògica pensar el la transformació física dels arxius i carpetes del sistema operatiu. Tot i això hem de pensar que no és cap innovació: els disquets han existit des de fa molts anys i els llapis USB a l'actualitat compleixen la mateixa funció.

A mitjans del projecte van sortir a la llum els plans de Microsoft en aquest camp. Aproximadament suposem que intentava fer quelcom molt semblant al nostre projecte denominat *Microsoft Surface.*

Microsoft Surface[28] es presenta com la taula multotouch multimèdia que reconeix els objectes que hi poses a sobre. Sembla que pretén implementar les funcions bàsiques del sistema operatiu i per tant de l'escriptori com la gestió de fitxers o les aplicacions multimèdia.

Per tot el que hem pogut veure fins ara tan sols hi trobem un intent de crear una sèrie d'aplicacions tangibles a mode de demostració molt separades entre si. El que han presentat fins ara semblen demostracions de la tecnologia i d'alguns conceptes avançats.

Pel que fa al les novetats, sembla que a algunes demos intenten desmarcar-se de les Wimp i a altres les util·litzen en tota regla. Bàsicament ho resumiríem com una sèrie de demostracions la potència del seu maquinari, però haurem d'esperar a que ho presentin amb més detall.

Figura 1.7.: Panasonic interactive table.

Figura 1.8.: mediaBlocks.

Figura 1.9.: Microsoft Surface

2. Planificació i anàlisi

2.1. Organització dins el grup

Donades les característiques d'aquest projecte, i que l'hem dut a terme entre dues persones, hem hagut d'utilitzar un seguit d'eines per a mantenir canals de comunicació tant per al desenvolupament, com per al Disseny de la interacció.

Inicialment, varem obrir una wiki de discussió[1] on es va plantejar tota l'estructura que hauria de tenir un servidor d'aplicacions tangibles, quines serien les formes d'interacció principals, i com s'aniria repartint la feina.

El fet d'utilitzar una wiki com a eina de discussió, va ser una gran avantatge, ja no teníem que quedar sovint per a fer reunions que gairebé mai concretaven res i posar-nos d'acord. La wiki havia solucionat el problema de la comunicació, tot i així un cop cada dues o tres setmanes establíem reunions entre nosaltres i el tutor del projecte per a fer un seguiment del procés global.

Més endavant, quan varem començar a tenir codi, varem migrar de la wiki a un servidor TRAC[2], un entorn que a part de tenir wiki incorporada, ens oferia eines de control de versions per al codi i fitxers relacionats amb el projecte.

El tret característic més important de TRAC, és la seva organització del projecte en fites que ens permetien mitjançant els tiquets[3] veure l'estat global del projecte en un parell de gràfiques.

2.1.1. Branques

Com comentem a la secció 1.1, el projecte consta de dues branques, una orientada a la implementació d'un sistema d'escriptori tangible i l'altre a l'estudi sobre les possibilitats dels sistemes TUI i de l'escriptori tangible mateix.

En aquesta memòria podreu trobar-les als capítols 3 i 4 respectivament, així com altra informació sobre les eines emprades, treball futur i annexes relacionats.

[1] http://chaosct.no-ip.org/wiki
[2] http://bestsheep.no-ip.org:1234/trac
[3] Etiquetes d'assignació de fragments de codi.

2.2. Anàlisi de requeriments

2.2.1. Requeriments de la interacció

Tot i que en un projecte com el nostre on la paraula *experimentació* és important, sembla que un no hagi de tenir objectius concrets sinó simplement experimentar, almenys una mínima reflexió prèvia de requeriments és necessària.

En el nostre cas es fa una llista de requeriments, pel que fa a la experiència d'usuari, més aviat oberta i general. Al llarg del projecte s'ha anat ampliant, confirmant en alguns punts i relaxant en alguns altres, fins a arribar a un llistat final, però no per això definitiu.

- El Sistema[4] s'ha de utilitzar només amb l'ajuda de les mans i els tangibles.

  És important establir aquest tipus de límits bàsics i aparentment evidents des del principi, sinó intentar-los imposar més endavant és tot un problema. Amb aquest en concret entenem que el sistema d'entrada ha de ser el mencionat, mans i tangibles, deixant de banda teclats accessoris, reconeixement de veu, reconeixement gestual tipus eye-toy, etc.. El disseny de les aplicacions en depèn enormement, així que és molt necessari.

- El Sistema ha de ser intuïtiu.

  Amb això volem dir que un potencial usuari ha de entendre moltes coses visualment i per analogia amb altres sistemes d'interacció que ja coneix. Per aconseguir això cal treballar a fons les metàfores emprades, la visualització de la informació i la coherència (punt següent).

- El Sistema ha de ser coherent.

  D'aquesta qüestió en fem un punt particular ja que és necessari si no es vol complicar infinitament el sistema. Les coses que signifiquen coses semblants funcionen de forma semblant: aquesta afirmació, que per fer un sistema funcional no cal que sigui certa, per a fer-ne un de intuïtiu i agradable és fonamental.

- El Sistema ha de ser multi-usuari[12].

  Volem que el TDesktop permeti la interacció de més d'una persona en la mateixa aplicació o aplicacions diferents alhora. Aquesta és una constant en casi tots els dissenys de taules interactives degut a la metàfora inherent ineludible de la taula com a lloc de reunió i interacció social.

- El Sistema no ha de tenir una direcció definida.

  Donat que volem que el sistema es pugui usar en taules rodones, i al ser multi-usuari, hem de tenir en compte que no podem restringir el sentit de les coses que apareixen a la pantalla. Això pot significar per exemple minimitzar el text o adaptar-lo (veure 4.2.1.4 a la pàgina 75).

[4] Aquí com a Sistema entenem tot el TDesktop: El servidor + les aplicacions.

- El Sistema ha de ser còmode.

  No hi pot haver programes que requereixin de l'adopció d'una postura incòmoda o de la utilització de mans molt distants. Sobre aquest tipus de restriccions en discutim més a la la subsecció 4.5 a la pàgina 85. També es pot aplicar a que la interacció ha de ser senzilla en el sentit de no requerir un diàleg interactiu massa estès per a fer accions senzilles.

També a part d'aquests requeriments, referents a la interacció i al procés de decisió del llenguatge interactiu particular de cada aplicació en concret, en vàrem fer d'altres:

- No s'han de descartar metàfores ni mètodes diferents d'entrada.

  És molt important considerar totes les opcions i fins i tot estar disposat a canviar aspectes fonamentals si hi ha raons per pensar que hi ha altres formes millors, o més adequades de fer.

- Cal defugir el sistema establert.

  Els sistemes interactius actuals, molts cops molt basats en els sistemes WIMP, no són bons perquè siguin habituals. Aquests poden arrossegar moltes contradiccions i deficiències de l'antiguitat per la seva incapacitat de inventar sistemes nous. Això no vol dir que ignorem totes les metàfores WIMP, però cal posar-les en quarantena i entendre'n el motiu de la seva creació abans de adoptar-les.

- Cal separar la interacció del processat.

  Cal deixar en mans de la aplicació la creació del llenguatge específic, posant-la a la capa més superior possible. Seguint el model Model-View-Controller, cal estar disposat a canviar la interacció en qualsevol moment sense afectar-ne tot el sistema a més baix nivell.

2.2.2. Requeriments del Sistema

Atès que el nostre sistema[5] esta enfocat a la experimentació amb interfícies tangibles i no a desenvolupar una aplicació en concret, els requeriments d'aquest s'han anat modificant durant tot el procés de creació. Però val a dir que hi han uns requeriments bàsics que hem seguit des d'un inici per tal de satisfer les característiques dels sistemes interactius tangibles en general[17] per així no tancar-nos cap porta:

- El sistema ha de ser capaç d'interpretar els missatges provinents del ReacTIVIsion i retransmetre'ls a qui li pugui interessar.

- El sistema ha de ser capaç de calibrar la sortida gràfica mitjançant algun mètode visual.

[5] En el nostre cas el sistema seria l'aplicació TServer.

- El sistema ha de ser capaç de suportar més d'una aplicació.
  - Gestionar la prioritat de dibuixat.
  - Retransmetre events d'entrada específics.
  - Aplicar transformacions al tot el conjunt de l'aplicació.
  - Apagar les aplicacions al finalitzar el sistema.
  - Apagar les aplicacions quan es re-inicii el sistema.
- El sistema ha de ser capaç de suportar com a mínim una *aplicació de control*.
- El sistema ha de ser capaç de proporcionar un feedback a tots els events de la taula.
  - Així com activar-lo i desactivar-lo quan es vulgui.

Pel que fa a la part del TServer això és tot, val a dir que s'han complert tots els requisits que varem plantejar en un principi així com requisits nous que han anat sorgint. Com per exemple:

- El sistema ha de ser capaç de suportar aplicacions no enfinestrades i amb diferents àrees tangibles.
- Les aplicacions no tenen perquè estar contingudes en una regió de la pantalla.
- S'ha de poder variar la prioritat de pintat de les aplicacions en calent.

En quan a les aplicacions remotes, els requeriments que ens varem plantejar varen ser:

- Les aplicacions han de poder dibuixar en qualsevol lloc de la pantalla.
- Han de suportar widgets[6].
- S'han de poder crear de manera fàcil mitjançant una api intuïtiva.
  - han d'existir llibreries per a la creació d'aplicacions tangibles.
    * Per al dibuixat.
    * Per al tractament d'àrees tangibles (TArea).
    * Per a la comunicació amb el servidor.
    * Per a la reproducció multimèdia.

Pel que fa als requeriments del sistema, aquests són els que ens varem plantejar inicialment, però com hem dit abans, s'han anat afegint molts més de mica en mica fins a arribar al sistema TDesktop.

[6]Entenem per a widget, un objecte que reacciona quan li incideix una figura.

2.3. Eines emprades

2.3.1. Trac

Trac[7] és un wiki ampliat i un sistema de seguiment d'errors per a projectes de desenvolupament de programari. Trac utilitza un enfocament minimalístic a la direcció de projectes de programari basada en web. La seva missió és ajudar als desenvolupadors a escriure bon programari sense molestar-se.

Proporciona una interfície a Subversion, un Wiki integrat i les corresponents eines d'informe.

Trac permet vincular wiki, descripcions i missatges, creant lligams i referències entre errors, tasques, historials de canvis, arxius i pàgines de wiki. Un histograma mostra tots els esdeveniments de projecte en ordre, fent molt fàcil l'adquisició d'una visió de conjunt del projecte i del progrés que segueix.

2.3.2. Wiki

Un wiki és un lloc que permet llegir, editar i crear pàgines que al final són visualitzades en format HTML. Els wikis s'utilitzen com a suport per a la col·laboració, documentació, recopilació d'informacions... Típicament, les pàgines s'emmagatzemen en una base de dades i s'inclouen hipervincles generats dinàmicament.

En el projecte hem fet servir 2 wikis diferents: el MediaWiki[8] i el del Trac.

2.3.3. SubVersion

SubVersion[9] és un sistema de control de versions dissenyat específicament per reemplaçar el popular CVS, el qual té vàries deficiències. És programari lliure sota una llicència de

[7] http://trac.edgewall.org/
[8] http://www.mediawiki.org/wiki/MediaWiki/es
[9] http://subversion.tigris.org/

tipus Apache/Llicència BSD i se'l coneix també com a svn per què aquest és el seu nom a la línia de comandes.

Una característica important del Subversion és que, a diferència del CVS, els fitxers versionats no tenen cadascun un número de revisió independent. En canvi, tot el dipòsit té un únic número de versió que identifica un estat comú de tots els arxius del dipòsit en un cert punt del temps.

2.3.4. RapidSVN

RapidSVN[10] és un front-end per al sistema de revisió de versions SVN escrit en C++ usant el framework xwWidgets. Facilita la utilització dels dipòsits SVN en un entorn gràfic.

2.3.5. MonoDevelop

MonoDevelop[11] és un sistema integrat de desenvolupament de codi obert, popular, per a la plataforma Linux, l'objectiu del qual és el desenvolupament de programari que usa tant l'entorn de Mono com de Microsoft .NET. Integra trets similars als de l'Eclipse o als de Microsoft Visual Studio.

Actualment , a part del C#, suporta Java, Boo, Nemerle, Visual Basic.NET i MSIL.

2.3.6. MonoDoc

MonoDoc[12] és el sistema de documentació de Mono. Conté les descripcions , exemples , principalment de les llibreries exclusives de mono i gtk, tot i que també documenta algunes classes de l'entorn .NET.

[10] http://rapidsvn.tigris.org/
[11] http://www.monodevelop.com/Main_Page
[12] http://www.mono-project.com/Monodoc

2.3.7. Inkscape

Inkscape[13] és un programa de codi font obert de dibuix de gràfics vectorials amb capacitats similars als programes privatius com Illustrator, Freehand o CorelDraw utilitzant el format de fitxer estandarditzat pel W3C SVG. Las característiques de SVG suportades inclouen les formes bàsiques, els camins, el text alfa, les transformacions, els gradients, l'edició de nodes, l'exportació de SVG a PNG, l'agrupació d'elements i molt més.

2.3.8. Gimp

GIMP[14] és el programa de GNU per al tractament d'imatges (GNU Image Manipulation Program), creat per voluntaris i distribuït sota la llicència GPL. Està construït amb les llibreries GTK les quals es van crear pensant en aquest programa. La primera versió es va desenvolupar per sistemes Unix i va ser pensada especialment per GNU/Linux, no obstant actualment existeixen versions totalment funcionals per Windows i Mac OS X.

2.3.9. LyX, LaTeX

LyX[15] és un editor WYSIWYM (Allò que veus és allò que vols dir). Això vol dir que l'usuari només s'ha de preocupar de l'estructura i el contingut del text, mentre que el format és gestionat pel LaTeX[16], un sistema de formatat i edició avançat.

Així amb LyX s'obtenen resultats professionals sense ser un expert en LaTeX, amb una interfície gràfica que en facilita l'edició.

[13] http://www.inkscape.org/
[14] http://www.gimp.org.es/
[15] http://www.lyx.org/
[16] http://www.latex-project.org/

2.3.10. KBibTEX, BibTEX

BibTEX[17] és una utilitat per formatar llistes de referències. Normalment és utilitzat en combinació amb LATEX.

KBibTEX és una interfície gràfica per a gestionar la biblioteca de referències BibTEX basada en KDE.

2.3.11. UniGnuPlot, Gnuplot

Gnuplot[18] és un programa informàtic flexible sota intèrpret de comandaments per a generar gràfiques en dues i tres dimensions de funcions i dades.

UniGnuPlot és un front-end de gnuplot que permet configurar fàcilment les gràfiques de gnuplot

[17] http://www.bibtex.org/
[18] http://www.gnuplot.info/

3. Desenvolupament

Per Daniel Gallardo.

El projecte TDesktop, consta d'una aplicació principal, anomenada TServer, que respon a mode de kernel de la interfície tangible: gestiona el pintat i els events d'entrada de totes les aplicacions tangibles que s'executen.

L'entrada del sistema, ve donada pels objectes o dits que es posen sobre la taula, i són les aplicacions qui s'encarreguen de donar una resposta a l'entrada dels usuaris.

3.1. Llibreries i tecnologia emprades

TDesktop no pretén ser cap projecte final per a ser utilitzat per a la gran majoria d'usuaris, sinó, pretén ser una plataforma d'experimentació per a dissenyar aplicacions tangibles i experimentar amb tots els aspectes d'aquestes interfícies. Donades aquestes premisses, hem dissenyat un sistema d'aplicacions i llibreries que permeten executar un gran nombre d'aplicacions amb certes peculiaritats: totes dibuixen sobre TServer, i reaccionen a events d'entrada proporcionats per dits o cursors i figures o fiducials(3.1.5.4 a la pàgina 46).

Degut a la gran varietat de seccions i subseccions de les que composen el TDesktop, hem utilitzat una gran quantitat d'eines i llibreries diferents per a dur a terme la seva implementació, tot seguit en detallem el seu paper i funcionament:

3.1.1. Llenguatge utilitzat

Per a la implementació de la major part del sistema, hem utilitzat Mono[1], un llenguatge orientat a objectes i multi-plataforma.

Mono[3] està compost per diferents eines:

[1] Mono: projecte de codi obert impulsat per Novell que proporciona eines lliures basades en GNU/linux, compatibles amb .NET (http://www.mono-project.com/Main_Page)

1. Una màquina virtual CLI (llenguatge comú d'infraestructura) que conté un carregador de classes, un compilador en temps d'execució (JIT[2]) i rutines de recol·lecció de memòria.

2. Biblioteca de funcions compatibles amb CLR[3].

3. CTS (Sistema de tipus comú) junt amb la compatibilitat CLR, permet que l'aplicació i biblioteques que el composen, siguin escrites en altres llenguatges i acabin compilant en bytecode compatible amb Mono. Ex: pots tenir una classe escrita amb C# de la que hereti una altre escrita amb C++.

4. Compilador de C#, monoBasic, java i phyton.

El fet que ens va dur a triar aquest llenguatge, va estar la seva facilitat d'ús (té una corba d'aprenentatge molt bona) , la capacitat de portabilitat i compatibilitat amb altres llenguatges[4].

- Facilitat d'ús :
  - Al tenir sistema de recol·lecció de memòria, no cal reservar-ne ni definir punters.
  - Les classes estan organitzades dins de Namespaces i les llibreries estan contingudes dins d'Assemblies (fitxers .dll)
  - Conté un gran ventall de llibreries sobre aspectes molt variats.
- Capacitat de portabilitat:
  - Com que és un llenguatge que compila a bytecode, l' "executable" generat, pot ser portat a qualsevol plataforma que suporti mono[5].
- Compatibilitat amb altres llenguatges:
  - Dins d'un mateix programa poden haver classes o parts de classes escrites en altres llenguatges fins i tot hi pot haver herència entre classes escrites en diferents llenguatges.

[2] *Just in time o Traducció dinàmica : Tècnica per a millorar el rendiment dels sistemes que compilen a bytecode, consisteix a traduir el bytecode a codi màquina en temps d'execució.*
[3] Common Line Runtime
[4] En part també ens va convèncer el creador de Mono, Miguel de Icaza en una conferència sobre aquesta plataforma-llenguatge.
[5] Actualment: linux, FreeBSD, UNIX, Mac OS X, Sun i Windows

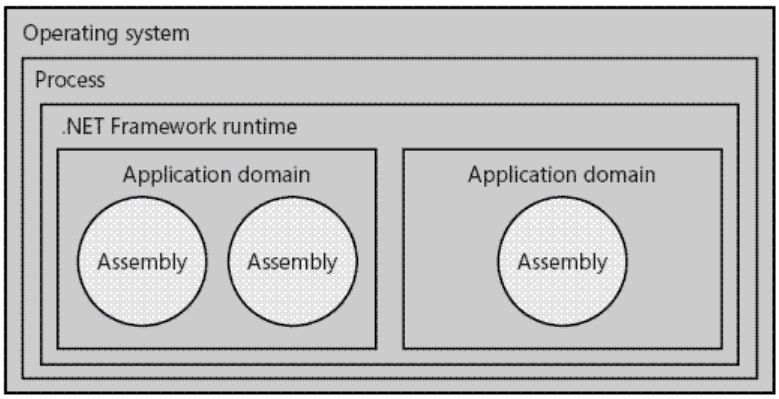


Figura 3.1.: Estructura de capes del runtime Mono.

En aquesta figura es poden observar les capes que hi han des de l'aplicació fins al sistema operatiu, es poden destacar dues peculiaritats:

- Les aplicacions dins del mateix "Application domain", poden comunicar-se entre elles directament.
- Les aplicacions situades dins del mateix nivell de capa, poden comunicar-se utilitzant un proxy per a intercanviar missatges gràcies a processos de "marshalling" i "serialization" (System.Remoting).

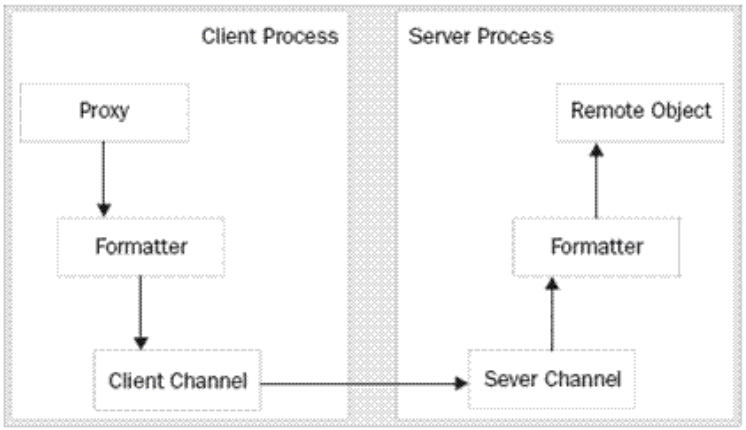


Figura 3.2.: Exemple objecte remot (Remoting).

Hi ha un objecte apoderat a la part del client que té domini sobre l'objecte de la part del servidor. Per a mantenir la persistència de dades, l'objecte és serialitzat a XML i transmès per un canal, per l'altra banda es fa el mateix, però, a la inversa.

3.1.1.1. Mono.Remoting[2]

Part del projecte mono que ofereix un canal de comunicació entre diferents aplicacions ja sigui dins de la mateixa màquina o cap a l'exterior.

Remoting utilitza estàndards com SOAP[6] per als missatges i HTTP i TCP com a protocols de comunicació. Concretament nosaltres utilitzem una classe sota el patró singleton que pot ser instanciada per a qualsevol aplicació. Remoting mitjançant la serialització XML s'encarrega de mantenir la persistència de les dades.

Fer això implica marcar com a serialitzables tots els objectes que pogués contenir i o utilitzar aquesta classe, per tant s'ha compilat una llibreria a part[7].

3.1.1.2. System.XML

Com hem explicat abans, els objectes es poden serialitzar a XML. Explotant aquesta funcionalitat, es pot serialitzar qualsevol classe a XML, fet que hem aprofitat per a crear classes que continguin dades de configuració per a posteriorment poder-les emmagatzemar en format text per a mantenir els paràmetres de configuració en futures execucions.

El procés és el següent:

[6] Simple Object Acces Protocol: estàndard acollit sota la W3C que defineix com diferents objectes es poden comunicar mitjançant intercanvi de dades XML.

[7] Es pot veure més endavant a l'estructura del codi.

1. Un cop iniciat el programa es crida als mètodes de lectura del rootSerializer i et retorna una instància del objecte que conté els atributs[8] que hi havien emmagatzemats en format text XML.

2. Les dades poden variar durant l'execució del programa.

3. Abans de tancar s'emmagatzemen les dades fent la crida als mètodes de rootSerializer per escriure les dades del objecte[9] al disc.

3.1.2. TAO[7]

Tao[10] és un framework que conté varies implementacions i bindings de les llibreries més importants de manipulació d'imatge, so i vídeo.

Al començar amb C# li vàrem trobar mancances a l'hora d'utilitzar certes llibreries que creiem essencials per a dur a terme el nostre projecte, varem optar per utilitzar Tao que utilitza la capacitat de mono(3 a la pàgina 38) i .Net per a fer bindings de les següents llibreries: OpenGl 2.1.0.3, FreeGlut 2.4.0.1, OpenAl 1.1.0.0, Cg 1.4.1.1, DevIl 1.6.8.2, Lua 5.1.1.0, SDL 1.2.11.1, PhysFs 1.0.1.0, ODE 0.6.0.3 and Glfw 2.5.0.0.

Tot seguit detallem les llibreries del Tao utilitzades al TDesktop.

3.1.2.1. OpenGl (Open Graphics Library)[5]

Especificació estàndard que defineix una API multi-llenguatge i multi-plataforma per a escriure aplicacions que produeixen gràfics 3D. Desenvolupada originalment per Silicon Graphics Incorporated (SGI).

C# i Mono incorporen ja de per si mètodes bastant bons per al dibuixat vectorial 2D, tot i que per al nostre sistema serien ideals, varen sorgir una sèrie de problemes inesperats que ens van dur a utilitzar OpenGl[11].

Aquests problemes varen ser:

- Mantenir un refresc de pantalla mínim per a no veure anomalies quan les aplicacions es dibuixen de manera distribuïda.

- No ens solucionaven els problemes de calibració(3.2.1.3 a la pàgina 55).

[8] Al document XML no tenen per que constar tots els valors dels atributs de l'objecte, per aquesta raó s'han de definir valors per defecte

[9] Si un objecte conté altres objectes aquests també quedaran emmagatzemats per a una posterior lectura.

[10] http://www.taoframework.com/

[11] http://www.opengl.org/

- El procés d'assemblar tots els fragments de les imatges de les diferents aplicacions per a mostrar-les per pantalla consumia molts recursos.

Per aquestes raons varem decidir migrar a OpenGl, ja que és bastant robust, podíem concatenar diferents "buffers"(**??** a la pàgina ??) per a ser dibuixats posteriorment i permet transformar la imatge resultant al nostre gust.

3.1.2.2. Glut

Extensió d'OpenGL per a facilitar la creació de diferents components d'OpenGL.

3.1.2.3. SDL (Simple DirectMedia Layer)[6]

Conjunt de llibreries desenvolupades amb el llenguatge C que proporcionen funcions bàsiques per a realitzar operacions de dibuixat 2D, gestió d'efectes de so i música i càrrega d'imatges.

Nosaltres les hem utilitzat per a Crear la finestra de renderitzat, carregar imatges i per a la llibreria de so. Ens vàrem decantar per SDL[12] perquè a part d'integrar totes les funcionalitats que necessitaven, en un futur ens podria permetre afegir reproductors de vídeo al TDesktop.

3.1.3. FreetypeFont

Tipus de lletra vectorial definida per fitxers que contenen diferents tipografies.

Per a carregar lletres dins del TServer, utilitzem la llibreria "libfreetype.so" que conté una sèrie de mètodes per a la lectura i càrrega de caràcters procedents d'arxius .ttf que contenen les fonts. Utilitzem les imatges resultants de l'extracció dels caràcters per a mapejar-les sobre textures OpenGl i així poder-les disposar al nostre gust sobre la pantalla.

Com que libfreetype.so no esta portat a mono, hem hagut de portar-la nosaltres mateixos important-la com a dll i aprofitant les característiques de Mono(3 a la pàgina 38).

3.1.4. LibMooTag[13]

Llibreria escrita en c que permet llegir els tags dels fitxers Mp3 i Ogg.

L'utilitzem a la part del TAudioLib per a poder extreure les meta-dades dels fitxers d'àudio.

Com que esta escrita i compilada amb C, també l'hem hagut de portar important-la com a dll.

[12] http://www.libsdl.org/
[13] http://mootag.sourceforge.net/

3.1.5. Tecnologies específiques del ReacTable

3.1.5.1. ReacTIVIsion

Part essencial del TDesktop, és un programa que s'encarrega de fer un tracking de tot el que esta passant sobre la taula, codificar-ho d'una manera concreta i enviar-ho a qui li pugui interessar via socket UDP.

Ha estat desenvolupat per Kaltenbrunner, M. & Bencina, R.

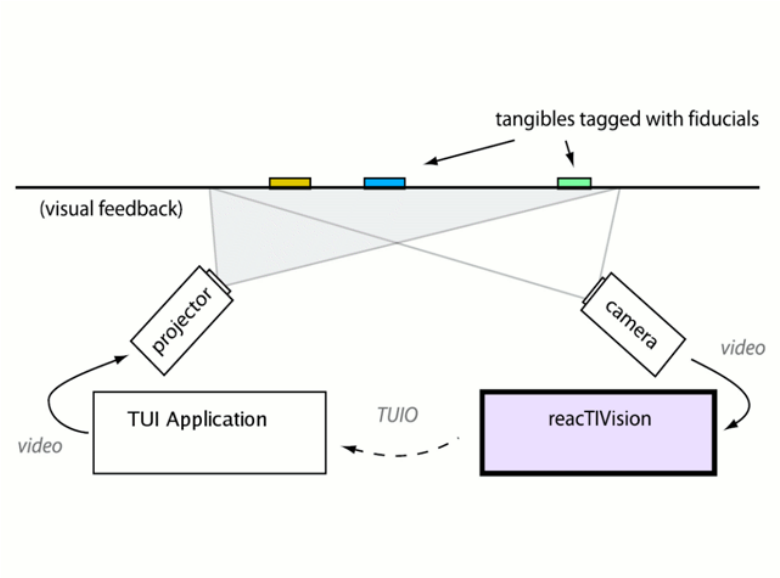


Figura 3.3.: Esquema ReacTIVIsion.
En aquesta il·lustració, es pot veure clarament com el ReacTIVIsion[23] és l'encarregat de processar les dades procedents de la càmera.

Un factor interessant és que a l'utilitzar sockets per a comunicar-se amb altres programes, el ReacTIVIsion pot estar corrent en una altra màquina per així poder utilitzar més d'una màquina per a fer córrer el sistema tangible complert.

El programa diferencia nomes dos tipus de figures sobre la taula, els cursors o dits i les figures o fiducials. Per a cada un d'ells fa una detecció i composició de dades diferents:

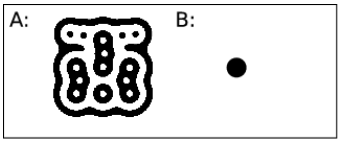


Figura 3.4.: Il·lustració d'un fiducial i d'un cursor.

- Cursor: El programa els capta com a punts sobre la taula, així que no hi ha manera de distingir-los un dels altres però si conèixer l'ordre amb que han aparegut a escena (identificador de sessió). Per tant les dades que se'n poden extreure són:
  - Posició X [x].
  - Posició Y [y].
  - Identificador de la sessió [Sid].
  - Vector de moviment del cursor [X,Y].[14]
  - Acceleració del cursor [m].
- Fiducial: Un fiducial és mes complex que un cursor, al ser un objecte, aquest pot ser diferent dels altres i al tenir una forma diferent, es pot trobar l'angle respecte la posició inicial del objecte. Per tant les dades que obtenim són:
  - Posició X [x].
  - Posició Y [y].
  - Identificador de la sessió [Sid].
  - Identificador de la figura [Fid].
  - Angle de rotació [a].
  - Vector de direcció [X,Y].
  - Vector rotació [A].
  - Acceleració de la figura [m].
  - Velocitat de rotació [r].

3.1.5.2. Estructura de la taula

La taula on fem funcionar la nostra aplicació, consta bàsicament de 4 parts:

1. Un projector per a mostrar el feedback visual.
2. Una càmera infraroja[15] per a captar els objectes / dits.
3. Un mirall[16] per a allargar el recorregut del projector.
4. Una superfície translúcida on es projecten les imatges i es disposen les figures i dits.

[14] Tenint en compte la posició en t i t+1.
[15] Òbviament, també, uns fars emissors de llum infraroja.
[16] Necessari, però afegeix un problema a l'hora de dibuixar, les imatges es veuen afectades per la distorsió que s'afegeix amb la inclinació del mirall i les seves característiques físiques.

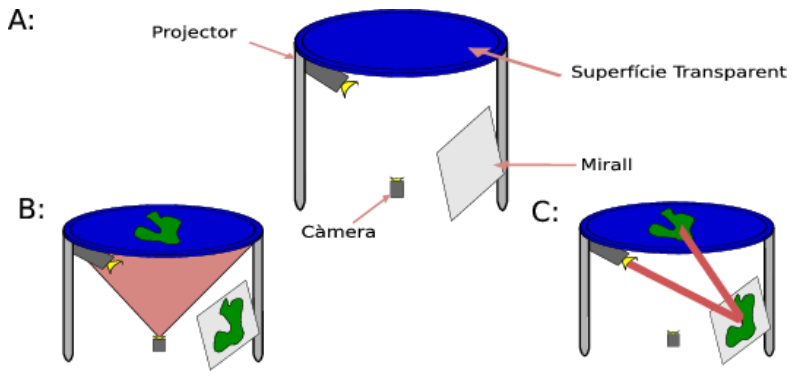


Figura 3.5.: Esquema general de la taula.

- A.- Parts que composen la taula.
- B.- Captura dels objectes de la taula.
- C.- projecció de les imatges.

3.1.5.3. TuioSimulator

Degut a les dimensions i cost[17] de la taula és necessari disposar del TuioSimulator. Vindria a ser com el ReacTIVIsion però en comtes de processar les imatges procedents d'una càmera, processa la posició del ratolí i on deixem certs objectes virtuals sobre la pantalla.

També envia les mateixes dades pel socket que el ReacTIVIsion, per tant és l'eina ideal per quan no es pot tenir accés a una taula de les característiques esmentades.

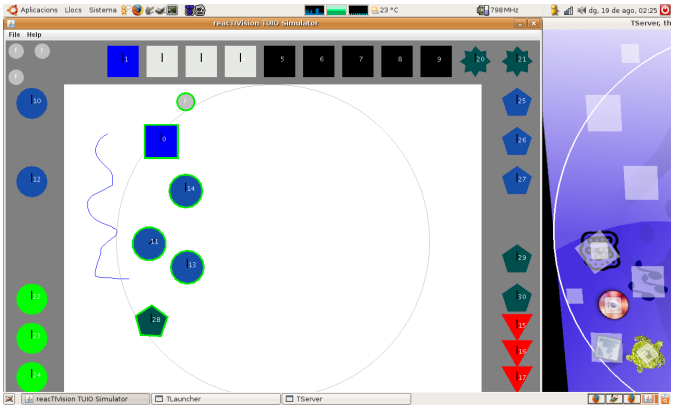


Figura 3.6.: TuioSimulator
Aquesta figura mostra el TuioSimulator funcionant i enviant dades al TDesktop.

[17] Bàsicament del projector (veure annex sobre taula casolana a la pàgina 135).

3.1.5.4. Tracking fiducials

Com hem dit abans, els fiducials són figures que poden ser diferents unes de les altres, per tant han de tenir trets característics distintius. Els fiducials, serien una espècie de codis de barres, però amb propietats 2D, o sigui, que la posició i rotació si que tenen repercussions finals. Les característiques principals dels fiducials són:

- Cada fiducial és diferent dels altres, i conté un numero de figura que l'identifica com a tal.
- El seu dibuix ha de ser el mes asimètric possible per a poder detectar la rotació.
- Tots els fiducials estan composts de punts negres i anelles.
- Han de ser visibles en l'espectre infraroig.
- Per a no produir possibles problemes en la seva detecció han de tenir la mateixa mida.

Com llegir un fiducial? Per a llegir un fiducial, partirem de la característica de que estan formats per anelles i punts, el fiducial en si és una anella gran que conté una combinació d'anelles i punts al seu interior.

Dividint la lectura en dues parts, detectar la posició relativa i l'identificador:

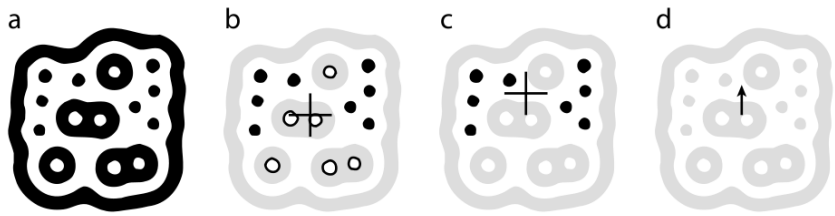


Figura 3.7.: Detecció posició i angle fiducial.

Per a detectar la posició del fiducial, és tant fàcil com trobar el punt mig de tots els forats i punts negres interiors (veure gràfic b). Per a trobar l'angle, necessitem el vector posició del fiducial, i per això necessitem obtenir dos punts característics dins del fiducial. Un és el trobat a la figura b, i l'altre és el punt mig de tots els punts negres(figura c)[24].

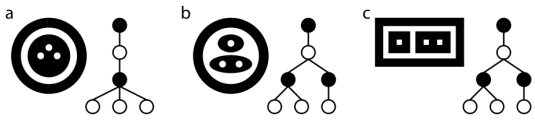


Figura 3.8.: Detecció Fid.

Per a detectar quin Fid correspon a la figura, es genera un arbre a partir de la informació del fiducial, per exemple el cas b:

1. Anella principal (negre)
2. Forat de l'interior (blanc)
3. El forat conté dos conjunts a l'interior (negre) (negre)
4. Una anella conté dos forats (blanc, blanc) i l'altre un (blanc)

Un cop generat l'arbre, es pot extreure un nombre binari assignat 1 als punts negres i 0 als forats, per tant un exemple de codificació de b seria : 0001101.

3.1.5.5. TuioMessage / OSC

Un TuioMessage, és el tipus de missatge que van del TuioSimulator o ReacTIVIsion fins a l'aplicació tangible. Aquests missatges s'envien via socket UDP per un port a escollir, com que la connexió es fa mitjançant UDP, els TuioMessages, han d'incorporar ells mateixos un mecanisme de control propi per a donar una possible solució a la pèrdua de paquets.

OSC Per a explicar l'estructura d'un TuioMessage, cal veure com és un missatge OSC[18]. Els missatges osc, estan dissenyats per a ser transmesos entre ordinadors, sintetitzadors de so i altres aparells multimèdia. Pretenen ser una evolució del protocol Midi.

Estructura d'un missatge osc:

1. Bundle o capçalera.
2. Declaració del tipus de dades que conté.
3. Dades

Com a exemple[4], codifiquem les següents dades en un missatge OSC.

```
1. The int32 1000
```

[18]Open Sound Control

```
2. The int32 -1
3. The string "hello"
4. The float32 1.234
5. The float32 5.678
```

El missatge osc en qüestió serà:

```
2f (/)  66 (f)  6f (o)  6f (o)   // /foo -> capçalera o Bundle
0 ()    0 ()    0 ()    0 ()     //  -> Es marca un canvi de dades amb un
                                 //     espai fins al proper octet.
2c (,)  69 (i)  69 (i)  73 (s)   // ,iisff -> dades del missatge
                                 //           (int int string float float)
66 (f)  66 (f)  0 ()    0 ()
0 ()    0 ()    3 ()    e8 (è)   // dades...
ff (ÿ)  ff (ÿ)  ff (ÿ)  ff (ÿ)
68 (h)  65 (e)  6c (l)  6c (l)
6f (o)  0 ()    0 ()    0 ()
3f (?)  9d ()   f3 (ó)  b6 (¶)
40 (@)  b5 (u)  b2 (?)  2d (-)   // fi dades
```

Com es pot observar a l'exemple anterior, els tipus de dades que poden anar en un missatge estan determinats per les característiques del protocol, a continuació detallem els tipus de dades permesos:

tag osc	tipus de dada corresponent
i	int32
f	float32
s	osc-string
b	osc-blob

Taula 3.1.: Tipus de dades permeses en osc.

TuioMessage Els TuioMessages[24], es separen en dos tipus de missatges, els orientats a cursors i els orientats a fiducials, degut a la diferència en la quantitat de dades.

Com que és necessari establir un mecanisme de control[19] cada missatge va enumerat amb un número de seqüència i són enviats amb redundància. El que li interessa al client que escolta aquests missatges és l'estat de l'últim update, les dades que han passat abans o s'han perdut, manquen d'importància ja que només interessa veure quin és l'ultima posició del objecte a la taula.

[19]Els missatges són transmesos per UDP, no orientat a connexió.

Cas d'un fiducial	Cas d'un cursor
/tuio/2Dobj	/tuio/cur
set	set
alive	alive
fseq	fseq

Taula 3.2.: Estructura dels dos tipus de tuiomissatges.

- SET: tag que indica la creació o actualització de les dades d'un objecte.
- ALIVE: tag que indica una llista amb tots els números de seqüència dels objectes que encara figuren sobre la taula.
- FSEQ: número de seqüència del missatge.

Per tant les dades d'un possible TuioMessage serien:

```
/tuio/cur                   // capçalera del missatge, en aquest cas un cursor
,siffffsiisi                // tipus de dades
set                         // indica actualització de dades
2                           // esta actualitzant les dades del cursor amb sid = 2
0.34                        // posició x
0.56                        // posició y
0.3                         // vector moviment component a
0.4                         // vector moviment component b
0.01                        // acceleració
alive                       // indica totes les figures que continuen sobre la taula
2                           // en aquest cas la 2 i la 4
4
fseq                        // nombre de seqüència del missatge 34
34
```

3.2. Implementació

TDesktop segueix el model client servidor, hi ha un servidor (TServer) i varis clients (TAplics), en el següent apartat detallem com esta construït el TServer i les llibreries bàsiques de TAplic.

Estructura del codi: Els assemblies i executables generats per a dur a terme la implementació són els següents:

TServer.exe és el client TUIO i a l'hora el servidor del TDesktop.

TServerObjects.dll Objectes compartits entre el servidor i client, necessari per a utilitzar remoting(3.1.1.1).

TLib.dll Llibreria que conté els mètodes necessaris per crear una aplicació compatible amb TDesktop(TAplic).

TAudio.dll Llibreria de so per a TAplic.

Comunicació entre processos: Per a comunicar TServer i les diferents TAplics, utilitzem Remoting per a comunicació directa entre servidor i clients, on es gestionen els inputs del sistema, operacions bàsiques d'aplicacions i dibuixat d'aquestes.

Com el projecte va orientat a experimentar amb aplicacions tangibles, hem creat un altre canal de comunicació el qual és d'ús exclusiu per a les aplicacions, ja que varem considerar que no calia re-compilar o entendre TSever per a poder crear aplicacions tangibles.

Aquest canal, utilitza remoting, però de forma transparent, ja que qui programa les aplicacions ho rep com a events del sistema. Aquests events porten una codificació especifica(veure 3.2.3 a la pàgina 60) per conèixer a quina aplicació va dirigida i quina és la funció per la qual ha estat generat.

3.2.1. TServer

TServer, està compost de dos threads, un fil és per al client TUIO(3.1.5.5 a la pàgina 48) i l'altre per al servidor de les TAplics.

Com que els dos fils estan dins del mateix *Application Domain,* la comunicació entre aquests és directa(3.1 a la pàgina 39), per tant comparteixen objectes i dades estàtiques.

Per a veure les parts que formen el TServer, hem decidit dividir-lo en 4 parts (tres de les quals bàsiques i una quarta de configuració).

3.2.1.1. Input (Part vermella de la Figura 3.10)

És la part on s'escolten, processen i calculen les col·lisions dels objectes de la taula amb les diferents àrees de les aplicacions.

Per fer-ho hem creat la classe *TuioClient* que s'executa en un thread diferent i es comunica directament amb *RemotingCollision. RemotingCollision,* s'encarrega de demanar a *AppBuffer* una llista ordenada[20] de les aplicacions[21] que s'executen en aquell moment.

[20]Ordenada segons prioritat de pintat (veure Comunicació a TLib), ordre de refresc.

[21]Cada aplicació té un seguit d'àrees tangibles definides per qui ha programat l'aplicació o el Widgets que utilitza.

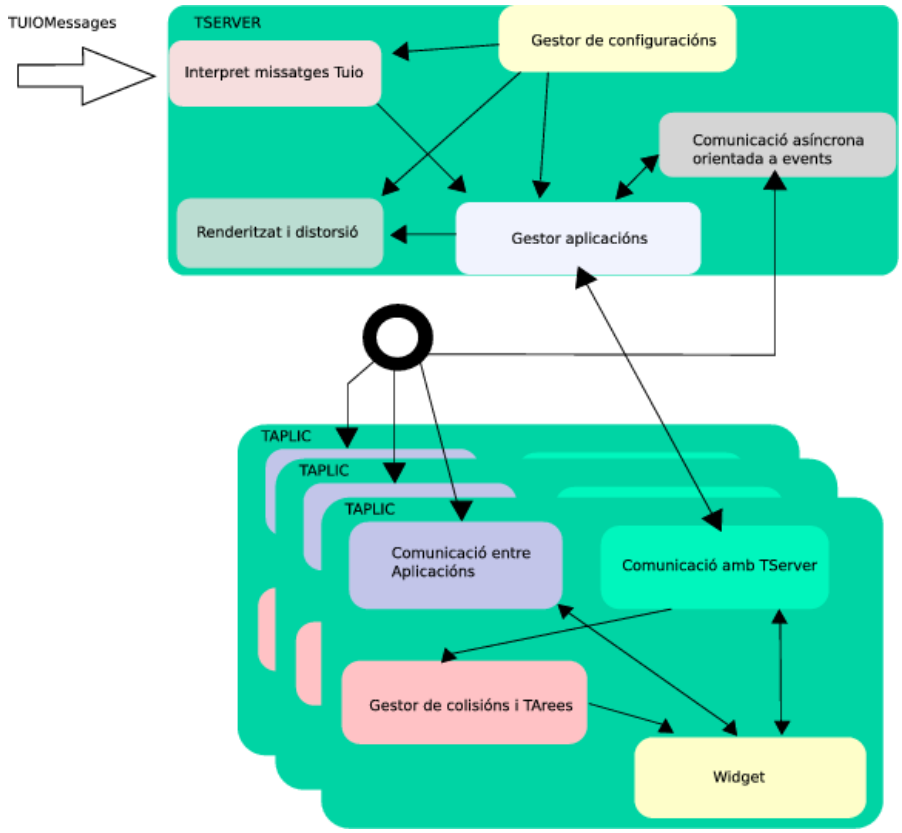


Figura 3.9.: Esquema general del TServer.

Cal destacar els dos canals de comunicació:

- Un bidireccional entre el *Gestor d'aplicacions* i la part de *comunicació amb TServer*.
- Un altre del tipus multicast entre les aplicacions , elles mateixes i TServer.

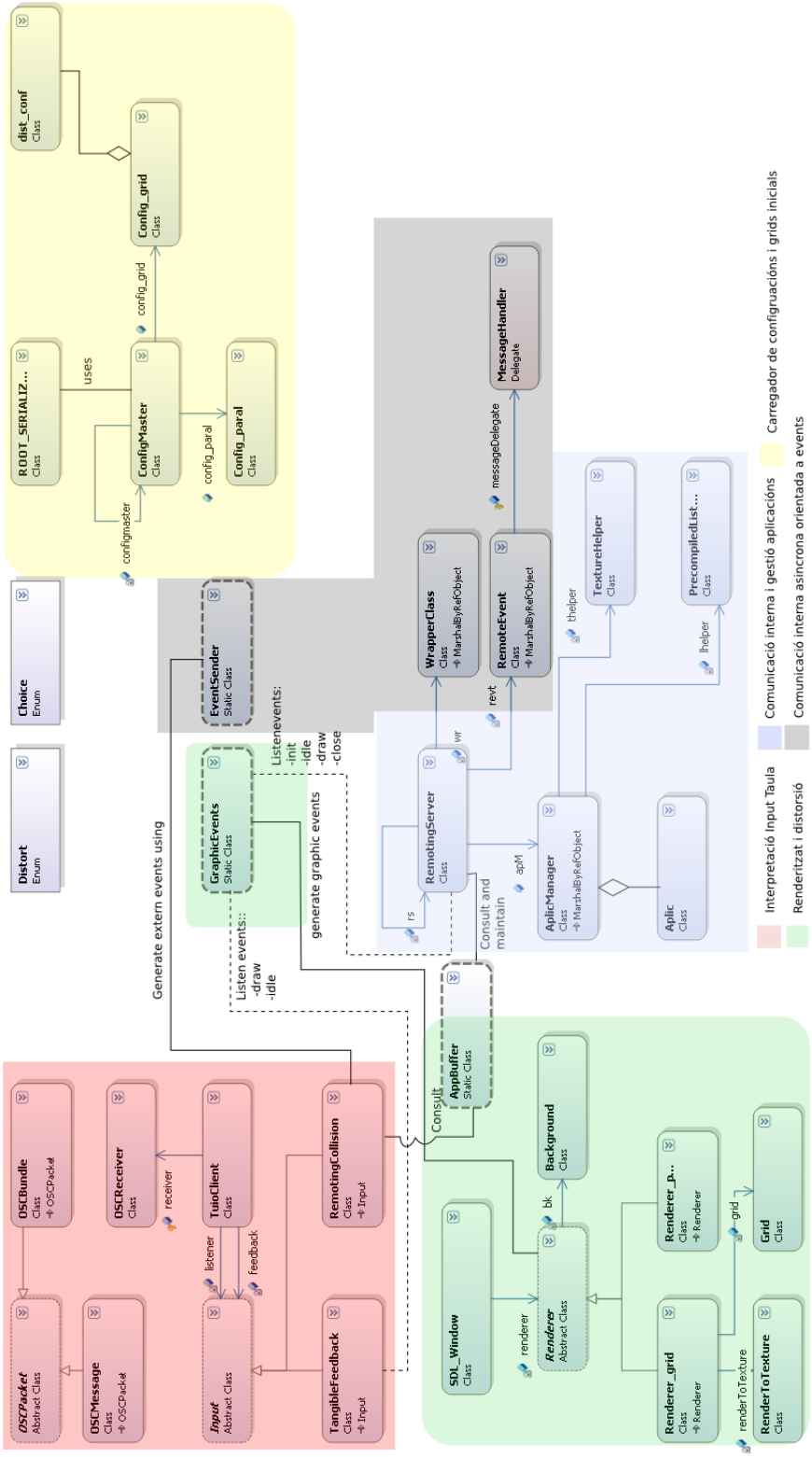


Figura 3.10.: Esquema TServer reduït.

Un Cop *RemotingCollision* ha detectat el tipus de col·lisió (veure 3.2.4.2 a la pàgina 63) ,es notifica a totes les aplicacions registrades: el tipus d'event que ha rebut, l'identificador de l'aplicació destí i el TuioMessage corresponent; per mitjà d'*eventsender*[22].

TangibleFeedback *TangibleFeedback* s'encarrega de dibuixar les ombres per sota de cada figura de la taula per a tal que l'usuari pugui veure si la seva figura ha produït algun tipus d'efecte al sistema. Pot ser desactivada ja sigui mentre funciona el TDesktop o canviant el fitxer de configuració3.2.1.4.

3.2.1.2. Control(Part blava de la Figura 3.10)

Aquesta és la part on es manipulen les aplicacions, es crea l'enllaç de comunicació i es tradueixen els buffers de pintat procedents de les diferents TAplics.

RemotingServer, s'encarrega d'instanciar els dos objectes que podran ser accedits remotament des de les altres aplicacions(TAplics). Aquests objectes són:

- AplicManager.
- RemoteEvent.

AplicManager, és l'encarregat de gestionar totes les aplicacions que es connecten al TServer, és l'objecte compartit per tothom, però està en possessió de TServer, per tant des de RemotingServer es pot accedir per a pintar les aplicacions cada vegada que ho demani la part gràfica.

Un altra tasca important de AplicManager, és la de gestionar les prioritats de les aplicacions, això es duu a terme mitjançant una pila (que es troba a AppBuffer) amb certes peculiaritats. Les aplicacions són separades per tipus i dins de cada tipus una aplicació passa a la part superior de la seva secció de pila cada vegada que aquesta es refresca[23]

opció de prioritat	Descripció
Fullscreen	L'aplicació es dibuixa sola, totes les altres aplicacions queden minimitzades.
System	Només per a les aplicacions del tipus sistema, es dibuixa sempre per damunt de totes les altres.
AlwaysOnTop	L'aplicació es dibuixa per sobre de la resta però per sota de les de sistema.
NormalMode	L'aplicació es dibuixa per sota de les AlwaysOnTop, Totes les aplicacions que no tinguin funcionalitats especials.

Taula 3.3.: Taula de seccions de prioritat de pintat dins de la pila.

[22] Veure apartat 3.2.3
[23] Rep un update procedent de la taula i es re-dibuixa.

Finalment, a la part de control, podem trobar dos objectes (TextureHelper i PrecompiledList) on la seva funció és carregar textures i GlList respectivament procedents de les aplicacions, ja que les diferents aplicacions no contenen cap apartat amb OpenGl sinó un wrapper (veure 3.2.2 a la pàgina 58).

Gestió de Processos No sempre passa, però a vegades una aplicació es penja o no respon. En un entorn de consola no seria problema, però en l'entorn del TDesktop, l'aplicació penjada té registrades unes àrees tangibles, un buffer pintant-se i tot un seguit de coses que anomenem aplicació fantasma. Per tant a vista d'usuari, hi hauria en pantalla una sèrie d'objectes que no produirien cap mena de feedback i s'estarien mostrant per pantalla les dades d'aquesta aplicació.

Per a evitar aquest problema ens hem aprofitat de RemoteEvent[24] i del sistema de "delegates" de mono. Quan a RemoteEvent li arriba un event, ho envia a tothom que hagi escrit un mètode delegat per a ell[25] per tant si no pot obtenir un delegat, es pot conèixer que hi ha una aplicació fantasma.

Per a descobrir quina i eliminar-la es segueix el següent procediment:

1. Es detecta que hi ha una aplicació Fantasma.
2. S'envia a Totes les aplicacions un event per a que emetin un senyal de vida.
3. S'espera un cert temps.
4. un cop tenim la llista de totes les aplicacions vives, podem conèixer quina és l'aplicació fantasma.
5. Eliminem les seves dades del TServer.
6. Matem el procés[26] si encara esta engegat[27].

3.2.1.3. Motor Gràfic (Part verda de la Figura 3.10)

El motor gràfic, és l'encarregat de: generar la finestra de dibuixat *SDL_ Window,* que conté el loop principal de l'aplicació; distorsionar la imatge resultant[28] i generar els events de dibuixat i control per mitjà de *GraphicsEvents*.

[24] S'utilitza per a la comunicació asíncrona 3.2.3 a la pàgina 60.
[25] Sistema d'events multicast.
[26] Com a identificador de l'aplicació dins del TServer, utilitzem el procés de la TAplic.
[27] Si encara esta engegat, vol dir que l'aplicació estava penjada.
[28] Per a evitar la distorsió del mirall (veure 3.1.5.2).

Event	Funció
Start	Event que indica a tots els objectes que hi estan subscrits que s'inicialitzin.
Idle	Event que indica refresc de figures. Es fa servir per a carregar textures i GlList, ja que és cridat abans de Draw.
Draw	Event per a que es dibuixin tots els buffers de pintat, hi estan subscrits els objectes TangibleFeedback i RemotingServer.
Close	Event de sortida del sistema, es fa servir per a indicar a les aplicacions subscrites que s'apaguin.

Taula 3.4.: Taula d'events gràfics i d'engegat i apagat.

Distorsió El problema de la distorsió, va ser un dels aspectes que més remodelacions del sistema va produir (veure: 3.1.2.1 a la pàgina 41 i D a la pàgina 137) .

Aquesta distorsió ve produïda per la reflexió de la imatge que surt del mirall (3.1.5.2 a la pàgina 44) i va cap a la taula. Així, s'ha d'aplicar una contra-distorsió per a que els efectes del mirall no es notin[29].

Depenent del mirall, hem implementat dos tipus de distorsió: La de malla[30] i la plana o paral·lela.

- La *distorsió de malla* consisteix:

  1. Abans d'iniciar:

     a) Calibrar cada punt de la malla. La malla consta de 7*7 punts que defineixen un pla Bezier, de tal manera que la imatge es mapejarà no sobre els 7*7 punts sinó sobre una interpolació d'aquests.

  2. Un cop calibrada la taula, per a cada refresc de pantalla:

     a) Agafar el buffer OpenGl i transformar-lo en una imatge.
     b) Mapejar aquesta imatge sobre una malla poligonal que s'ha deformat prèviament al nostre gust.
     c) Mostrar la malla amb la imatge mapejada

  Un dels principals avantatges d'aquest tipus de distorsió és que es poden posar gairebé tots tipus de miralls[31] (còncaus, convexos, plans, ...). Però aquest sistema, té un inconvenient, i és que per a fer tot el procés de distorsió, s'utilitzen bastants recursos i no totes les targetes gràfiques el suporten.

[29] Si no s'apliques aquesta distorsió, la resposta visual no correspondria amb les imatges.
[30] Ideada per Marcos Alonso per a la ReacTable, nosaltres l'hem adaptat a les nostres necessitats i portat a Mono.
[31] Sempre i quan ajudin a mostrar sobre la la taula tota la dimensió de la pantalla

- La *distorsió plana o paral·lela* és la distorsió que nomes es produeix en la utilització de miralls plans, de tal manera que la imatge resultant només queda blincada degut a l'angle de refracció del mirall.

  Com que aquesta distorsió és més simple, és tant fàcil com afegir rotacions i escalats abans de dibuixar el buffer:

  1. Abans d'iniciar l'aplicació:

     a) S'ajusta el centre de la imatge al centre de la taula.

     b) Es rota en l'eix de les X per a inclinar la imatge fins que totes les línies de control siguin paral·leles.

     c) S'ajusta la mida de la sortida.

  2. Un cop calibrada la taula, nomes cal pintar a cada update gràfic.

  Com es pot observar, aquest sistema gairebé no consumeix recursos, ja que només aplica 3 transformacions (traslladar, rotar i escalar) abans de dibuixar. Però només és vàlid per a miralls plans.

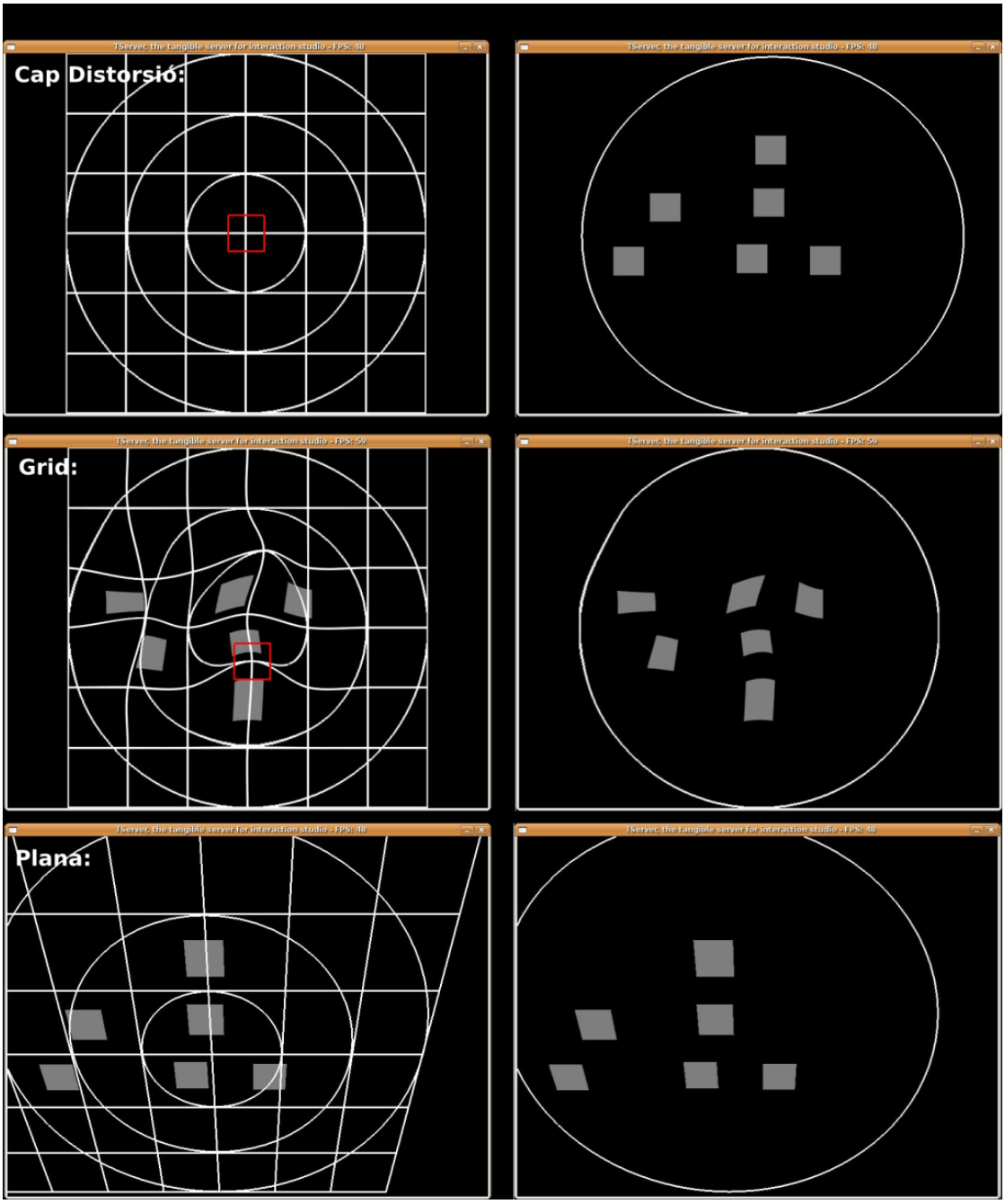


Figura 3.11.: Esquema del tipus de distorsions.

3.2.1.4. Configuració (Part groga de la Figura 3.10 a la pàgina 52)

Per a no haver de tornar a calibrar la taula cada cop que s'inicia el programa, hem aprofitat la característica de serialitzar els objectes a xml i emmagatzemem les dades necessàries per a recuperar els paràmetres de calibració de l'ultima sessió, així com algunes variables que no tenen per a què estar predefinides.

Exemple de fitxer de configuració TServer

```
<TServerConfigurationFile>
  <Distortion_kind>PARAL</Distortion_kind>
  <Window_width>1024</Window_width>
  <Window_heigth>768</Window_heigth>
  <MAX_FPS>60</MAX_FPS>
  <TangibleFeedback>TRUE</TangibleFeedback>
  <TuioPort>3333</TuioPort>
  <RemotingPort>1984</RemotingPort>
  <Manager>../bin/TManager.exe</Manager>
  <config_paral grid_width="7" grid_height="7" Dist_heigth="7.860026"
  Dist_width="7.800036"Angle="-359.5128"Center_X="0.3299981"Center_Y="0.
  00999924"/>
  <config_grid BufferTexture="512" grid_width="7" grid_height="7">
    <Distort_points>
      <P X="0" Y="0" />
      <P X="0" Y="0" />
      ...
    </Distort_points>
  </config_grid>
</TServerConfigurationFile>
```

Com es pot observar a l'exemple anterior, hi han codificats a Xml 3 objectes diferents, el ConfigManager (TServerConfigurationFile), el config_paral i el config_grid (que alhora conté un array de punts de distorsió).

3.2.2. TGL

Com hem dit abans, les diferents aplicacions client, no disposen d'accés a la llibreria OpenGl, si no que dibuixen mitjançant un wrapper que hem desenvolupat específicament per a aquesta tasca.

Aquest wrapper es compon d'una classe estàtica anomenada TGL que conté els mètodes necessaris per a poder dibuixar lliurement des de les aplicacions, el que fa Tgl és anar posant les instruccions dins d'un buffer cada vegada que es crida a un mètode.

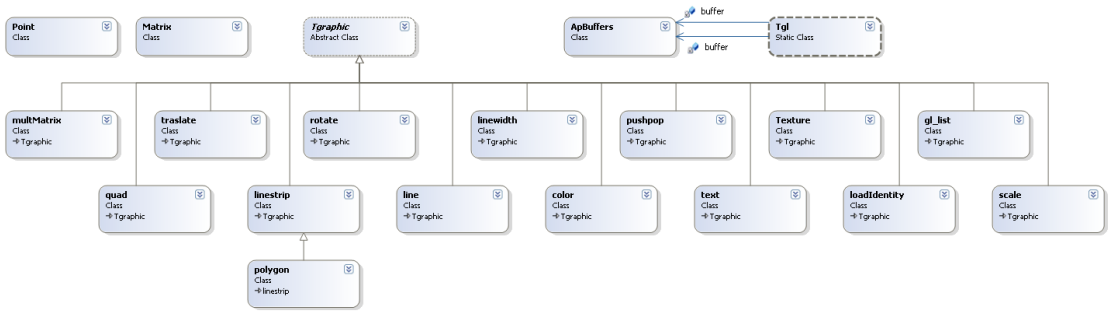


Figura 3.12.: Detall dels components de la part gràfica de TServerObjects.

Per a evitar problemes de refresc (passen quan TServer demana el buffer i resulta que l'aplicació s'està actualitzant en aquell precís moment[32]), s'ha creat dins de TGL, un sistema de doble buffer, de tal manera que quan s'està pintant un, l'altre es va omplint.

Un altre aspecte a destacar, és el de les Textures i les GlList, com que aquestes no es poden generar a l'instant, es fa de la següent manera:

Textures Es genera un index intern dins de Tgl i s'afegeix a un buffer específic per a carregar figures, juntament amb l'index generat per TGL. Un cop el TServer genera la textura i extreu l'index OpenGl, el que es fa, és cada vegada que a TServer li arribi una instrucció per a pintar la textura, fa la transformació per l'index GL de veritat.

GlList S'agafa el buffer que s'està editant en aquell moment, i s'encapsula amb un index generat per TGL. Dins de TServer, es genera la figura i s'associa a l'index de TGL tal i com es fa amb les textures.

Exemples de fragments de codi TGL

```
//Dibuixar un quadrat rotat
Tgl.color(r,g,b,A);
Tgl.rotate(Math.PI);
Tgl.pushMatrix()
Tgl.quad(0,0,0.5,0.5);
Tgl.popMatrix();
//carregar una textura i dibuixar-la
int index = Tgl.loadTexture("imatge.png");
Tgl.drawTexture (index, punts_posicio_PoligonMapeig, puntsMapeig);
```

[32] Passa molt sovint.

3.2.3. Events i comunicació entre aplicacions

En aquesta secció es pretén explicar el canal de comunicació que hi ha entre aplicació i aplicació i entre TServer i aplicacions. Cal remarcar que aquest tipus de comunicació es asíncrona, o sigui que no es genera cada cert temps i no necessita de resposta per part del receptor.

Aquesta comunicació s'estableix per mitjà de RemoteEvent, objecte el qual és instanciat dins de TServer i s'encarrega de generar events multicast per a totes les aplicacions registrades. Aquests events serveixen tant com per enviar events de sistema del TServer a les aplicacions com per que les aplicacions es comuniquin entre elles.

Els events remots, segueixen una estructura determinada, en primer lloc han d'heretar de System.EventArg, per tant hem dissenyat tota una estructura d'events per a poder posar dades de destí, canal[33], i receptor.

Dins de l'estructura dels events (veure 3.13), cal destacar:

eventMsg És l'event base de la comunicació ja que pot contenir altres events, els seus arguments són l'id de l'aplicació al que va destinat, l'identificador de missatge (canal), i els arguments del missatge (un altre event imbricat).

CurEventArgs Correspon a un missatge de cursor (TUIO) passat a event, per tant contindrà totes les dades procedents del TuioMessage.

FidEventArgs És l'homòleg al CurEventArgs però per al cas dels fiducials.

3.2.3.1. Events del sistema

Són els events que envia TServer per a controlar i notificar col·lisions[34] a les TAplics. Aquests events estan registrats als canals del 1 al 4 de la següent manera:

```
Apagar / resetejar aplicació:
 EventMsg[ Aplicació_destí , 0 , eventInt[-1]] --> Apagar
 EventMsg[ Aplicació_destí , 0 , eventInt[ 3]] --> Reset
Modificar estat de l'aplicació:
 EventMsg[ Aplicació_destí , 0 , eventInt[0]] --> FullScreen
 EventMsg[ Aplicació_destí , 0 , eventInt[1]] --> NormalScreen
 EventMsg[ Aplicació_destí , 0 , eventInt[2]] --> AlwaysOnTop
Permisos de pintat ( minimització ):
 EventMsg[ Aplicació_destí , 0 , eventInt[4]] --> Dibuixar
 EventMsg[ Aplicació_destí , 0 , eventInt[5]] --> No Dibuixar
Evnviar senyal de vida:
```

[33] El canal, és simplement un numero per identificar quin tipus de missatge és.
[34] S'entén per col·lisió que una figura de la taula ha incidit amb una àrea tangible d'una aplicació.

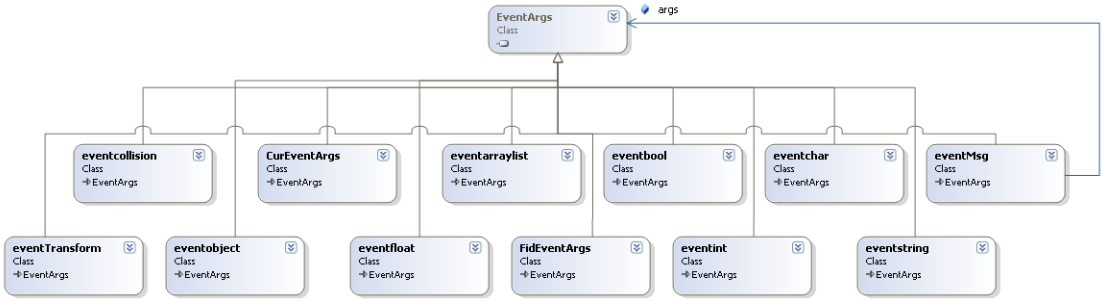


Figura 3.13.: Detall dels events del sistema.

```
 EventMsg[ Aplicació_destí , 0 , eventInt[6]] --> Alive
Colisions:
 EventMsg[ Aplicació_destí , 1 , EventMsg[tipus_de_col·lisió , null, event_TUIO]]
Transformacions:
 EventMsg[ Aplicació_destí , 2 , eventTransform[matriu de transformació]]
```

3.2.3.2. Events entre aplicacions

Aquests events, poden ser definits sense haver de canviar codi del TDesktop, però l'única condició, és que no utilitzi un canal reservat [0-3]3.2.3.1. Com a exemples, al TDesktop, els utilitzem per establir un canal de comunicació entre l'aplicació teclat i les diferents aplicacions que continguin caixes de text.

3.2.4. TAplic

TAplic és el nom genèric de tota aplicació que es connecta com a client al TServer. La principal característica d'aquestes aplicacions és que estan orientades a Widgets.

Entenem com a widget una entitat que es representa gràficament i respon als events d'entrada de la taula. Aquests widgets, poden ser compostos per altres widgets per a crear-ne un conjunt més gran, l'aplicació.

3.2.4.1. Comunicació

La comunicació de TAplic, també té dues vies, la comunicació directa entre TAplic i TServer per mitjà de l'objecte compartit AplicManager i la comunicació entre aplicacions orientada a events (3.2.3.2).

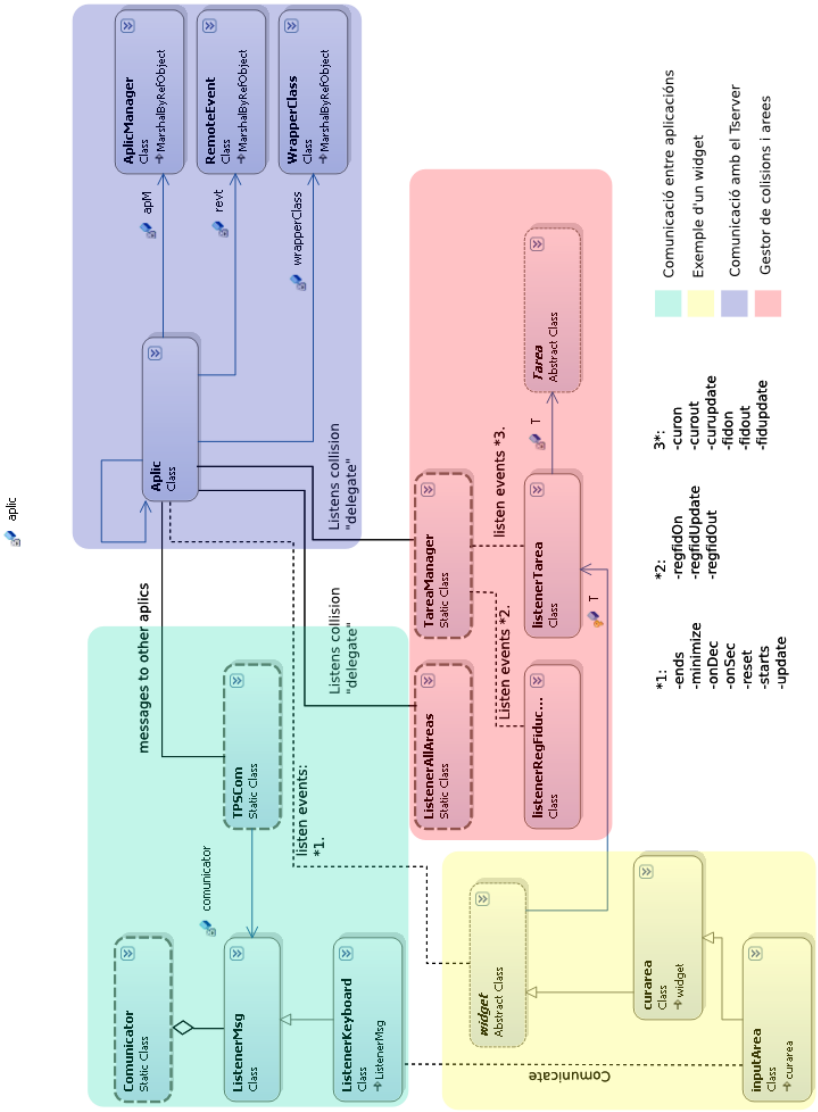


Figura 3.14.: Detall de la llibreria bàsica de les TAplics (TLib).

Si s'observa la Figura 3.14, a la part verda, es troba la classe estàtica *Comunicator,* aquesta classe és una façana per a que qui hagi de programar les aplicacions tangibles no hagi d'entendre tota l'estructura d'events ni del remoting, tanmateix per a facilitar la programació de les TAplics.

Es pot observar, que aquesta classe conté *ListenerMsg*, la qual és una classe auxiliar que determina un canal prefixat (veure 3.2.3 a la pàgina 60) per a poder crear noves formes de comunicació entre aplicacions fent servir herència o simplement una instància de *ListenerMsg* sense haver de passar ni tant sols per *Comunicator.*

Com a canals de comunicació preestablerts per il·lustrar diferents formes de comunicar-se, tenim TPSCom i ListenerKeyboard.

TPSCom Conté una instància de ListenerMsg, fa servir el canal 5 i serveix per a que les aplicacions passin dades a TManager[35] (icona, opcions, icones de les opcions, pid,...)

ListenerKeyboard Hereta directament de ListenerKeyboard, utilitza el canal 4 i serveix per a que hi pugui haver comunicació entre l'aplicació teclat i les diferents àrees de text.

3.2.4.2. Gestor Col·lisions

Com es pot veure a la part vermella de 3.14, es poden observar dues classes estàtiques que escolten directament de Aplic tots els events relacionats amb les col·lisions.

TareaManager Escolta totes les col·lisions d'àrees relacionades amb la TAplic on es troba, de tal manera que obviarà les dades de les figures que incideixen sobre la taula però no sobre les àrees de l'aplicació. Aquesta classe, genera uns events que són escoltats per diferents instàncies dels objectes següents:

- **ListenerRegFiducial** Aquesta classe escolta casos especials de fiducials. Algunes aplicacions poden tenir registrats un numero d'identificació de fiducial per a poder rebre totes les dades relacionades amb aquell fiducial en concret. Es fa servir sobretot amb aplicacions de control, per exemple un fiducial que quan el posis sobre la taula apagui totes les aplicacions.
- **ListenerTarea** Escolta tots els events de col·lisió relacionats amb una àrea específica. Aquesta àrea és qualsevol tipus d'objecte que hereti de de TArea[36]. Es fa servir sobretot dins dels widgets.

ListenerAllAreas Com el nom indica, escolta els events relacionats amb totes les àrees i tots els fiducials que es disposen sobre la taula. La seva funcionalitat fins ara és proporcionar un feedback de pulsació amb els dits i d'orientació amb els fiducials des de la TAplic TangibleFeedback.

[35] veure 5.1 a la pàgina 93
[36] Solen ser formes predefinides: Arc, cercle, poligonal, buida.

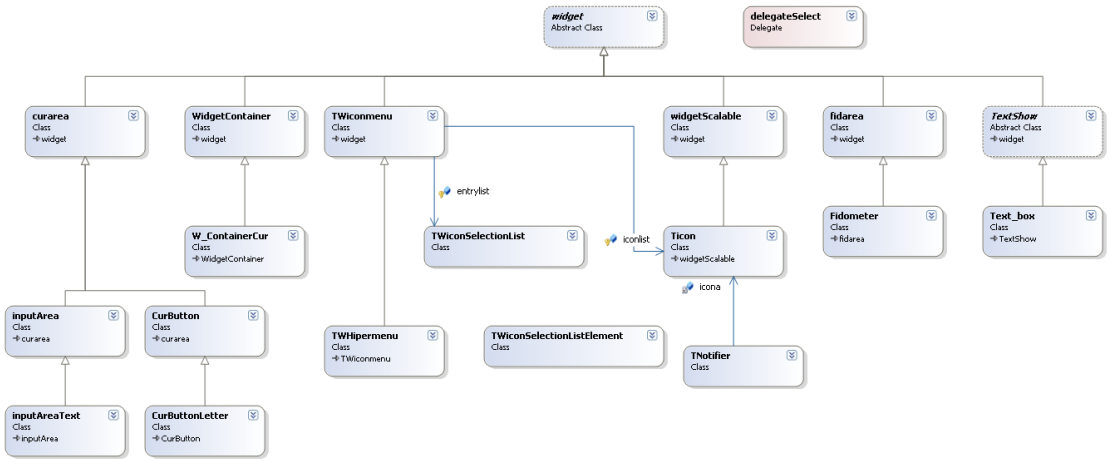


Figura 3.15.: Detall de l'estructura dels widgets.

3.2.4.3. Widgets

Els widgets són la part essencial de les Aplicacions tangibles, interpreten l'input de l'entrada de dades, i generen l'output visual.

La seva estructura (com podeu veure a la Figura3.15) penja tota de la classe widget, widget és una classe que es registra automàticament a Aplic i escolta els events de control. A la figura, es pot observar que widget conté una instància de ListenerTarea, la qual escoltarà tots els events relacionats amb l'àrea del widget.

Cal destacar, dins de l'estructura dels widgets els següents elements:

curarea (hereta de widget) conté una àrea tangible i tots els mètodes de *widget*, però, només escolta els events relacionats amb els dits (cursors).

fidarea Com curarea però només escolta els events relacionats amb fiducials.

WidgetContainer Contenidor de widgets, pot contenir tants widgets com es vulgui per a crear un widget compost.

W_containerCur Hereta de *WidgetContainer,* però té la peculiaritat de poder escalar i transformar amb els dits tots els widgets dels que està compost (exemple teclat).

3.2.5. Sobre el sistema complert

Com es pot observar a l'estructura general del sistema TDesktop, no hi ha res que estigui tancat o definitiu. Això és degut a que des d'un principi volíem que el disseny del sistema

no afectés a les aplicacions tangibles[37]. Per aquesta raó s'ha ideat tot el sistema de pintat independent per a cada aplicació, sistemes de comunicació entre aplicacions completament oberts i una base per a les aplicacions per a que puguin utilitzar els components que vulguin (els widgets són simples façanes que es poden reescriure fàcilment).

Totes aquestes mesures, han estat preses perquè el disseny de la interacció en les interfícies tangibles és del tot obert ja que no existeixen unes receptes especials per a les aplicacions tangibles, i el seu potencial no està tan explorat com en el cas de les WIMP.

Per tant, no és el sistema qui decideix quin és el disseny d'interacció, sinó que són les aplicacions que se'l definiran per elles mateixes. Tot seguit, trobareu un estudi sobre el disseny de la interacció en les aplicacions tangibles al qual hem intentat ser bastant fidels a l'hora de dissenyar les aplicacions demostració.

[37] No estar sotmeses a finestres(Wimp), gràfics predefinits, events fixats.

4. Disseny de la interacció

Per Carles F. Julià

En les següents seccions llistem una sèrie de formes d'interactuar tant per l'usuari com per el sistema intentant classificar-los pel concepte que representa o empeny a cada acció.

Som conscients que aquesta no serà probablement l'única divisió possible ni tan sols la millor, però hem intentat ser prou exhaustius per distingir-ne els tipus bàsics. Ja existeixen altres formes de classificació i de descobriment[35], però aquesta és la que poc a poc ha anat aflorant durant el desenvolupament del projecte.

Les reflexions que ens han portat a tal conclusió intenten ser resumides a cada apartat, però segurament no ho fan amb tota la fidelitat que hauríem desitjat. Igualment les múltiples disputes sobre un i altre model les intentem fer aparèixer sempre que són raonablement necessàries.

Finalment l'adequació proposada d'un o altre model o d'una o altra alternativa d'interacció és fruit d'una combinació de reflexió conceptual i experiència personal de l'equip, a l'espera d'una metodologia precisa i objectiva de proves amb usuaris reals per a una conclusió més definitiva.

4.1. Elements de la interacció a disposició de l'usuari

Com ja s'ha explicat en l'apartat tècnic (3.1.5.1 a la pàgina 43) l'usuari podrà manipular el sistema amb les mans, a través dels dits i dels tangibles. Per tant a l'hora de desenvolupar el llenguatge específic del sistema podrem utilitzar totes les possibilitats que ens ofereix aquesta combinació.

4.1.1. Accions bàsiques detectades pel sistema ReacTIVIsion

La plataforma ReacTIVIsion ens permet detectar certs esdeveniments d'interacció a la taula. Aquests són els següents:

1. En referència als dits:
    a) Posar un dit a la taula.
    b) Arrossegar el dit per la taula.

Acció	Dades proveïdes
1a	coordenades
1b	noves coordenades
1c	(cap)
2a	coordenades, angle i identificador
2b	noves coordenades
2c	nou angle
2d	(cap)

Taula 4.1.: Dades proveïdes per les accions simples detectades per ReacTIVIsion.

c) Treure un dit de la taula.

2. En referència als tangibles:

a) Posar un tangible a la taula.
b) Arrossegar un tangible per la taula.
c) Canviar l'angle del tangible.
d) Treure un tangible de la taula.

Per a més detall la Taula 4.1 ens mostra cada acció amb les dades proporcionades pel sistema ReacTIVIsion.

4.1.2. Accions complexes

Sabent aquestes accions bàsiques detectades en podem derivar-ne moltes de complexes que podrien tenir sentit en el nostre llenguatge comunicatiu amb la taula, formades amb una combinació de les anteriors, o de una metàfora concreta aplicada a una acció bàsica.

Cada acció complexa normalment correspon a una metàfora ja existent en altres camps i situacions de la cultura comuna. Això fa que com a la vida real les accions tenen unes conseqüències reals, les accions rals en el nostre cas han de tenir unes conseqüències digitals semblants o equivalents a les de les metàfores originals.

Podem classificar aquestes accions segons si es fan amb els dits o a través dels tangibles, o si es fan contra un objecte digital(per manipular-lo) o no (Taula 4.2). Veurem així que hi ha diferents estratègies que compleixen les mateixes funcions i per tant són en teoria equivalents.

4.1.2.1. Arrossegar

Combinació de Posar el dit(1a) arrossegar-lo (1b) i treure'l(1c).

	dits	tangibles
manipula objectes digitals	Arrossegar, Redimensionar, Deformar	Moure, Girar
no manipula objectes digitals	Clicar, Dibuixar	Cargolar, Col·locar, Colpejar, Pintar

Taula 4.2.: Classificació de les accions complexes.
Classificació de les accions complexes. segons els dos eixos dits-tangibles i manipulació d'objectes.

Aquesta acció ve inspirada per una acció de la vida real a l'hora de moure un full en una taula, per exemple.

La resposta que l'usuari pretendrà percebre serà que l'objecte lògic es desplaça mantenint les proporcions i orientació per a mantenir el dit en la mateixa posició relativa.

4.1.2.2. Redimensionar, orientar

Combinació d'Arrossegar feta amb 2 dits.

Aquesta acció tot i ser bastant intuïtiva en trobem pocs exemples en la vida real. Podríem imaginar una malla elàstica que amb l'ajuda dels dits podem anar-la redimensionant. Possiblement la naturalitat d'aquesta acció rau més en ser combinació directa de accions molt intuïtives que en ser-ho ella mateixa.

La resposta esperada per l'usuari serà de veure com l'objecte s'escala proporcionalment i es trasllada per a fer coincidir les posicions relatives de cada dit amb l'objecte amb les que originalment ocupaven al ser posats els dits.

4.1.2.3. Deformar

Combinació de Arrossegar diversos dits.

Aquesta acció tot i ser menys usada (més per la dificultat de implementació i la falta d'utilitat d'àmbit general que per poc intuïtiva) és equiparable al que faríem amb una massa de pizza o arrugant un paper.

La resposta esperada serà la de deformació de la imatge o *morphing[8]*[1] mantenint els punts relatius sempre en el mateix lloc (Figura 4.1).

4.1.2.4. Clicar, prémer

Combinació de posar un dit(1a) i treure'l(1c).

[1] Actualment es sol conèixer aquest efecte com a *wrapping* i es manté *morphing* quan involucra una fusió amb una imatge resultant.

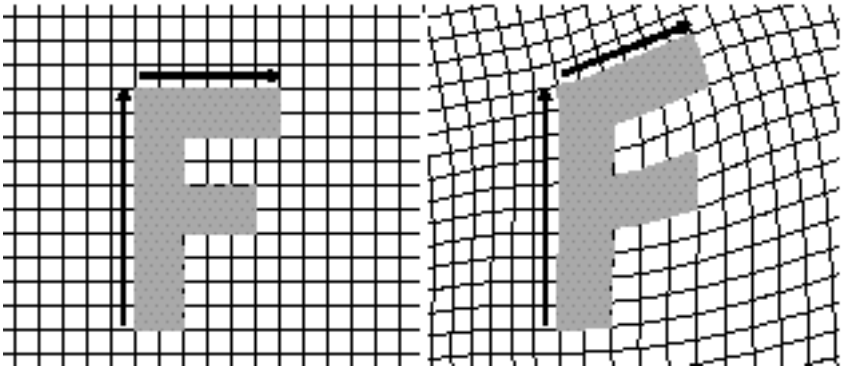


Figura 4.1.: Exemple de deformació de la imatge o *morphing*.
Exemple de deformació de la imatge o *morphing* amb tres punts de control que correspondrien a tres dits a la taula.

Aquesta acció directament hereva dels polsadors constitueix una de les hereves més clares de les metàfores d'interacció de les WIMP.

L'usuari esperarà l'activació de l'objecte o que desenvolupi una acció determinada. Normalment aquest tipus de activació ha estat sempre especificada per un indicador ja sigui textual o icònic.

4.1.2.5. Dibuixar

Sota aquesta denominació que inevitablement indueix a confusió, vull no només posar el que ens imaginaríem per *dibuixar* pròpiament dit al parlar de interfícies tangibles, sinó també a tota una sèrie d'accions que poden involucrar objectes, però que en cap moment s'estan manipulant.

Així doncs aquí englobaríem totes les accions que tenen com a metàfora per exemple una pissarra. En podem veure una mostra:

- Dibuixar.
- Encerclar.
- Ratllar o tatxar.
- Escriure (tant en gestos com en escriptura manual).
- Subratllar.
- Unir.

Els resultats esperats venen des de la introducció de dades (dibuixar, escriure) passant pel canvi d'estat (ratllar) fins a destacar o agrupar (encerclar, subratllar, unir).

Figura 4.2.: Ventosa usada per aixecar els terres de les oficines

4.1.2.6. Pintar

Podem definir breument aquesta acció dient que es tracta de l'acció Dibuixar realitzada amb tangibles. L'avantatge hi és en la opció de aprofitar les propietats dels tangibles que no tenen els dits: poden portar indicacions i forma, tenen angle, i es poden deixar indefinidament a la taula.

Per tant diem-li combinació de Posar el tangible(2a) arrossegar-lo (2b) i treure'l(2d).

4.1.2.7. Moure

Combinació de Posar el tangible(2a) arrossegar-lo (2b) i treure'l(2d).

Podríem pensar en una nansa, o en un agafador. Ens serveix per a moure un objecte, per a agafar-lo. Una imatge per il·lustar aquest concepte seria les ventoses usades per aixecar els terres de les oficines (Figura 4.2).

La resposta que l'usuari pretendrà percebre serà que l'objecte lògic es desplaça mantenint les proporcions i orientació per a mantenir el dit en la mateixa posició relativa.

4.1.2.8. Girar

Canviar l'angle d'un tangible(2c).

Tot i que sembla molt bàsic, cal explicar aquesta acció, ja que no és l'única que s'identifica amb l'acció bàsica. Aquesta correspondria sobretot a la manipulació de la direcció.

S'espera bàsicament que l'angle real i el lògic siguin exactament el mateix. La rotació representa un angle, i al fer-ne una volta sencera el tangible afectat ha d'estar exactament al mateix lloc (fora d'haver-hi límits que es poden resoldre de diferents maneres però que presenten una discontinuïtat en l'angle).

Un exemple paradigmàtic seria per exemple la rotació d'un objecte digital.

4.1.2.9. Cargolar, descargolar

És una variant de Canviar l'angle d'un tangible(2c).

La diferència és bàsicament en la interpretació del valor de l'angle. Posem-nos l'exemple d'una aixeta: en fer-la girar en el sentit horari disminuïm el cabal d'aigua mentre que si ho fem de forma antihorària l'augmentem. Un exemple clar de l'aplicació d'aquesta metàfora n'és la instrucció d'escalat del Turtan (Annexe E a la pàgina 143)

L'usuari espera simplement un augment o disminució proporcional[2] d'un aspecte visual de l'aplicació. En aquest cas la resposta visual és crucial ja que les dades físiques i la interpretació lògica poden ser de diferent natura, així que l'usuari no pot saber a priori la traducció entre l'angle i la magnitud.

4.1.2.10. Col·locar

Simplement és posar un tangible(2a).

Posar un objecte en un lloc determinat per a que la seva influencia canvii alguna cosa. Aquesta metàfora la podríem haver manllevat simplement dels jocs de cartes. Una carta a la taula té un efecte desitjat de forma persistent mentre està a la taula.

Aquí s'espera que l'objecte mantingui la seva funció al llarg del temps, aquesta funció o estat pot ser múltiple (podríem canviar-lo de moltes formes) però sempre es un estat perpetu. Almenys 2 estats són necessaris: Actiu i no actiu.

4.1.2.11. Colpejar

Com l'anterior es tracta de posar un tangible(2a).

Posar un objecte sobre la taula amb l'objectiu de que s'executi una funció en el moment. L'objecte hauria de ser inútil després. També possiblement inspirat en els jocs de cartes, hi ha cartes que modifiquen el joc però un cop usades es poden descartat. La mateixa filosofia està darrera d'aquesta acció.

4.2. Elements de la interacció a disposició del sistema

De la mateixa manera que l'usuari disposa de un alfabet i un lèxic particular per a comunicar-se, el sistema necessita d'uns de propis per la seva part.

[2] Fixi's que evitem dir *directament* proporcional, ja que no sempre cal que sigui així per a ser intuïtiu (veure 4.6.1 a la pàgina 86).

Figura 4.3.: Retorn visual al TurTan.

4.2.1. La projecció

Claríssimament la projecció de les imatges sobre la superfície és el canal principal de comunicació des del sistema cap a l'usuari. Els humans en general basem la nostra comprensió del món en la informació visual.

Les eines següents les podríem classificar segons si aporten alguna cosa per si soles a la interacció, amb l'usuari, o són purament informatives.

- Aporten quelcom a la interacció.
  - Retorn visual.
  - Àrea d'influència.
- Són informatives
  - Icones.
  - Text.

4.2.1.1. El retorn visual

Podem entendre-ho també com l'ombra dels tangibles i els dits. Aquest confirma la presència digital del tangible o el dit en el sistema (Figura 4.3).

El retorn visual ens serveix per moltes coses: en primer lloc la indicació ja citada de la presència digital del tangible. Tant per la detecció del mal funcionament del ReacTIVision[3] o per la lògica particular del sistema (el tangible ja no té cap influència per tant indiquem que ja no és necessari).

En segon lloc les seves propietats poden tenir un significat concret: el seu color, forma, moviment... poden servir d'indicadors d'estat per exemple.

4.2.1.2. Les àrees d'influència

El sistema ha de dibuixar els límits de les àrees d'influència on els tangibles un cop posats allí hi tenen una funció específica. Un mateix tangible pot tenir diferents funcions segons on es dipositi, fins i tot oposades.

Per la pròpia experiència del projecte, cal dir que no cal abusar de les àrees fixes a objectes traslladables o escalables (Veure 4.5.1).

4.2.1.3. Les icones o símbols

Un dels avantatges, i potser el més important, d'usar símbols o icones és que normalment tot i tenir un sentit (tenen un *amunt* i un *avall*) no en són dependents per la seva comprensió. Per tant suposant que la superfície de la taula és accessible des de qualsevol dels seus costats. Això és importantíssim ja que altres tipus d'informadors (com el text) estan limitats en aquest sentit, tot i que veurem que hi ha algunes tècniques per a evitar-ho.

Les icones poden ser usades lliurement a la superfície de projecció per a indicar informació a un usuari. Aquesta informació pot indicar un estat del sistema, un esdeveniment o pot representar en ell mateix un objecte digital.

Per a distingir entre un indicador d'estat i d'esdeveniment cal tenir en compte que:

- Un estat és durador i per tant la seva icona ha de perdurar sempre que hi perduri l'estat que representa.
- Un esdeveniment és puntual i per tant la presencia de l'indicador ha de ser efímera. Cal tenir en compte , però que cal que duri suficient temps per a que l'usuari el pugui veure i entendre. En el nostre cas afegim també efectes visuals per a cridar l'atenció (fer pampallugues, desaparèixer...)

[3] El mal funcionament molts cops és degut a la mala il·luminació, configuració dels llindars de detecció, enfocament de la càmera... en cap moment es critica el ReacTIVIsion com a programari de detecció.

Figura 4.4.: Ambigrames.
A sobre un ambigrama automàtic ("TDesktop"), a sota un de fet manualment ("Revelation").

4.2.1.4. El text

El text és en general desaconsellable en sistemes com el nostre, que no tenen una orientació definida. Per un usuari pot ser per exemple dificultós de llegir un text del revés. Tot i així podem millorar-ne la condició amb diferents tècniques:

- Generar textos que es llegeixin del dret i del revés. A Aquests els anomenem *Ambigrames[27].* El problema aquí el tenim amb que no són fàcils d'aconseguir, i els mètodes automàtics[1] donen un resultat bastant pobre. A més no tots els texts són susceptibles a construir un ambigrama (Figura 4.4).
- Escriure textos de forma circular. Hem determinat que si part del text està del dret pel lector , és molt més fàcil llegir la part invertida (Figura 4.5). Per altra banda sempre es pot fer rotar el text perquè el lector pugui llegir-lo tot del dret, evitant així haver de llegir del revés.

4.2.2. El so

El so és útil per al sistema per a 2 motius.

En primer lloc per aplicacions multimèdia (reproduir música, vídeos...). Ja coneixem l'exemple del Reactable i per tant podem comprendre la importància que pot tenir el so multimèdia en el sistema.

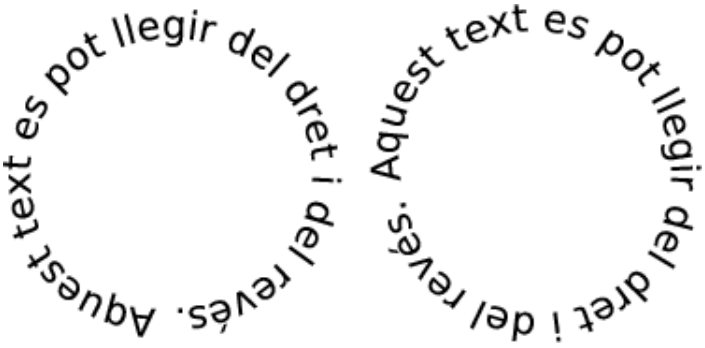


Figura 4.5.: Text circular.
El text circular és fàcil de llegir des de qualsevol cantó.

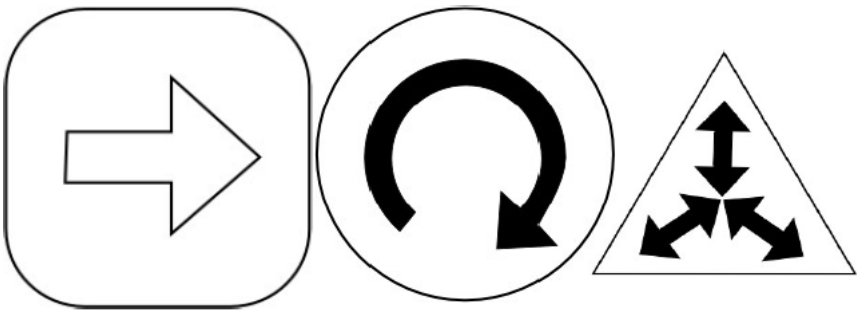


Figura 4.6.: Indicacions i formes dels tangibles.
Indicacions i formes d'alguns tangibles util·litzats en el Turtan (Subsecció 5.7). Cadascun ens ajuda a identificar-ne la funció.

En segon lloc per a cridar l'atenció als usuaris: anunciar-los esdeveniments o requerir-los l'atenció.

4.2.3. Forma i indicacions del tangible

Una forma una mica curiosa de comunicar-se amb l'usuari poden ser tant la forma com el color i indicacions escrites en el tangible. Aquestes anuncien una funció preestablerta o un rol del tangible en qüestió (Figura 4.6).

Aquest tipus de comunicació és estàtica i per tant forma part del disseny del sistema o aplicació; però no per això hem de treure-li importància.

La forma del tangible té també un rol significatiu no només d'informació, sinó per les limitacions que provoca en la relació amb altres tangibles. Per exemple podríem tenir un tangible que permetés l'annexió d'un altre, no només una annexió digital, sinó física. Tractem el tema en més detall a la Secció 4.4.

	Nansa	Modificador d'estat	Colpejador	Manipulador de propietat	Pinzell
Traslladable	x				
Escalable	x				
Deformable	x				
Polsador			x		
Commutador		x			
Manipulable de propietat				x	
Dibuixable					x

Taula 4.3.: Compatibilitat entre rols d'objectes físics i rols d'objectes digitals.

4.3. Elements comuns

Els objectes, tant físics (tangibles) com digitals, tal com hem vist estan controlats tant per l'usuari com per el sistema (de forma estàtica o dinàmica) així que es converteixen en objectes amb significat compartit per ambdós interlocutors.

Podem veure la compatibilitat entre els objectes físics i els digitals en la Taula 4.3.

4.3.1. Rols dels objectes físics (tangibles)

Una indicació en el tangible (Subsecció 4.2.3) i el context específic del sistema fan que l'usuari interpreti el rol del tangible i per tant intenti usar-lo segons les metàfores relacionades a través d'accions complexes (Subsecció 4.1.2) que involucrin tangibles (Moure, Girar, Col·locar, Cargolar, Colpejar). Intentarem que quedin cobertes totes amb els rols llistats aquí.

Els rols presentats aquí són genèrics i intuïtius (ideals). Altres rols més limitats o elaborats seran presentats en la discussió sobre la interacció de cada aplicació concreta en la Secció 5.

4.3.1.1. Nansa

Aquest rol serà el de una nansa (com la de la Figura 4.2), objecte quotidià de tota la vida.

Implementarà principalment Moure i Girar, que bàsicament són les accions complexes de manipulació a través de tangibles. Així doncs tenim un *manipulador d'objectes digitals.*

4.3.1.2. Modificador d'estat

Aquest rol compliria amb la metàfora de les cartes anteriorment mencionada.

Implementarà Col·locar, que bàsicament afecta a les coordenades. D'aquesta manera ens queda l'angle lliure per a , per exemple, implementar Cargolar per a escollir sobre una llista discreta però cíclica d'estats, o la intensitat de l'estat.

En aquest cas el símbol de l'estat és molt útil que estigui indicat en el mateix tangible, ja que té significat i influència tot i que no estigui en contacte amb la superfície (la seva no presència indica que l'estat està desactivat).

4.3.1.3. Colpejador

Aquest rol compliria amb l'altra metàfora de les cartes anteriorment mencionada.

Implementa Colpejar i en principi la seva posició i angle no és important (a no ser que es vegi influït per àrees d'influència).

Com en el cas anterior el símbol de l'acció indicat al tangible és bàsic si no essencial.

4.3.1.4. Manipulador de propietat

Aquest rol correspon al tangible que s'usa per a canviar una propietat del sistema amb el seu angle.

Evidentment implementa Cargolar i la seva posició no és essencial (a no ser que hi hagi àrees d'influència).

La diferència essencial que ens fa distingir aquest rol del modificador d'estat és que treure o posar el tangible a la taula no té cap significat. No hi ha cap estat que depengui de la presencia del manipulador de propietat, només de les seves variacions de l'angle[4].

Aquest tipus d'objecte és molt amigable, com podem veure en les diferents formes que pren en la vida real (Figura 4.7).Alguns exemples van des d'aixetes, passant per controls de so fins a volants.

4.3.1.5. Pinzell

Aquest rol té el nom de la seva descripció, la metàfora no pot ser més clara.

I fent honor al seu nom, implementa Pintar.

És interessant mencionar que la seva forma no és accessòria i es pot jugar molt amb aquest paràmetre. L'angle usualment tant pot indicar la direcció com l'amplada del pinzell, tot i que no ha de fer literalment de pinzell per obligació.

[4] Per usabilitat no considerem una variació d'angle si traiem i tornem a posar el tangible a la taula amb un angle diferent. En l'acció complexa Cargolar només importen les variacions relatives i no l'angle en si.

Figura 4.7.: Manipulador de propietat.
Un amplificador que usa la metàfora de Manipulador de propietat.

4.3.2. Rols avançats dels objectes físics (tangibles)

El rol que té cada objecte físic dins de la aplicació és un de concret molt més específic que els rols anteriors. Aquests poden ser rols estàtics (inherents al tangible) o dinàmics (adquirits a través de la taula).

Per decidir quin model seguim hem de tenir en compte algunes coses:

- Un rol estàtic ocupa un tangible que en principi no ha de tenir cap més rol.
- Els rols estàtics ocupen lloc.
- Els rols dinàmics necessiten d'un sistema d'adjudicació de rols que pot ser complicat o ocupar lloc a la taula).
- Els rols dinàmics són difícils de recordar un cop no són a la taula.

Per tant si fem tangibles dinàmics que perden el significat al treure'ls de la taula amb un bon sistema d'adjudicació de rols, tindrem un sistema que no necessiti de moltes peces per a funcionar. Aquesta serà la nostra prioritat.

4.3.3. Rols d'objectes digitals manipulables

Els objectes digitals han de presentar una sèrie de propietats complint rols específics que ens permetin d'executar totes les accions complexes dirigides a la manipulació d'aquests.

4.3.3.1. Traslladable

Aquest objecte té la propietat de deixar-se moure per la taula. Bàsicament intentarà emular un objecte qualsevol en aquest aspecte.

Implementa Arrossegar i Moure i gestiona el trasllat del mateix. Aquesta metàfora no té gaire secret i ja l'hem discutida a l'apartat de arrossegar.

4.3.3.2. Escalable

Aquest objecte pot canviar la mida i l'orientació. Es pot fer amb tangibles o amb els dits. Estén Traslladable i per tant n'hereta les seves implementacions.

Implementa Redimensionar i Girar, a més de Arrossegar i Moure heretades de Traslladable.

4.3.3.3. Deformable

Aquest objecte ja canvia les proporcions relatives sota demanda amb el morphing (recordar Figura 4.1). Hereta de Escalable, ja que la Deformació amb 2 dits o posicionadors és equivalent a l'escalat.

Implementa Deformar, així com inevitablement per la seva herència de Escalable, també Redimensionar, Girar, Arrossegar i Moure.

Cal dir que degut a la freqüent utilització d'àrees d'influència lligades a objectes manipulables i degut a la dificultosa computació d'aquest efecte, no és massa utilitzat i en el nostre cas no l'hem usat.

4.3.4. Rols d'objectes receptors de dades

Podem entendre per objectes receptors de dades, aquells que capten les accions de l'usuari i en capturen les dades per processar-les. Estaríem parlant doncs de les ja citades àrees d'influència.

Aquestes àrees defineixen on un tangible tindrà un cert significat i on no. Estableixen doncs un context dins d'aquesta.

L'aplicació més immediata és la de receptor de les accions de manipulació d'objectes per aplicar aquestes manipulacions sobre l'objecte lògic pròpiament dit. Aquest concepte, però, ja ha estat suficientment tractat a l'apartat anterior (4.3.3).

Així ens centrarem en els objectes receptors de dades tal i com les entenem en un context on els objectes digitals ja implementen les metàfores anteriors: les dades de magnituds, de accions, de estats, i altres.

4.3.4.1. Polsador

El polsador àmpliament conegut per tots tant per les polsadors físics com pels digitals dels sistemes WIMP, funcionen esperant que algú els polsi.

Implementarà, per a l'acció de polsar, Clicar i Colpejar. Sempre volent dir que aquestes accions es faran dins de l'àrea d'influència.

4.3.4.2. Commutador

El commutador, també àmpliament conegut pel seu ús real i digital, ens permet passar a través de diferents estats.

Implementa Clicar i Col·locar; si és Col·locar, sempre tenint en compte que quan el tangible no està a l'àrea d'influència, aquest ja correspon a un estat. En canvi si s'opera amb els dits només es pot anar canviat d'estats de forma cíclica.

4.3.4.3. Manipulable de propietat

Aquest és un objecte comodí molt interessant: per modificar una propietat qualsevol d'un programa, ho fem a través d'aquest. Definir-lo massa seria contraproduent ja que és bastant abstracte i es pot arribar a solapar amb molts d'altres però ho podem intentar.

Implementa sobretot Cargolar, i explicarem els diferents tipus de relació entre les dades que proporciona l'usuari i les propietats modificades a l'apartat 4.6.1.

Podem saber que aquest tipus, que es relaciona bàsicament amb l'objecte Manipulador de propietat de l'apartat anterior, és de les formes més còmodes de manipulació. hem de pensar en l'exemple de la clau i el pany. La clau és un Manipulador i el pany un Manipulable (Figura 4.8).

4.3.4.4. Dibuixable

Aquest rol d'objecte implementa Dibuixar i Pintar. Pot rebre el rol Pinzell.

No només s'ha d'entendre que serveix per dibuixar en el sentit estricte de la paraula. En la majoria dels casos simplement serà una acció que deixa un traç a través de la taula i que pot significar alguna cosa concreta.

Una llista la tenim a la definició de Dibuixar (Secció 4.1.2.5).

Potser ho podríem comparar una pissarra o un objecte més modern, com un tabletpc, tot i que el concepte és molt ampli.

Figura 4.8.: Model Clau-Pany.
El model Clau-Pany és un exemple de la relació Manipulable de propietat i Manipulador de propietat. El pany només es pot modificar amb la clau.

4.3.5. Equivalències i preferències en ambdós rols

Hi ha alguns rols receptors (bàsicament els referents als objectes digitals) que poden rebre accions tant de tangibles com de dits per utilitzar un mateix concepte. Això posa de rellevància l'equivalència que ja havíem observat entre diferents accions complexes.

4.3.5.1. Traslladable

El primer cas el trobem en el Traslladable. Aquest pot ser traslladat amb una Nansa o amb Arrossegar. Així en general no podem discernir quina és la millor estratègia d'interacció. Tot i així sempre que sigui possible i compatible, és una bona idea permetre-les les dues. Hem de tenir en compte que:

- Els objectes ocupen un lloc.
- Els objectes tapen la projecció.
- Els dits costen de ser ben detectats per ReacTIVIsion.
- Els objectes poden implementar un altre rol amb el seu angle.

Més enllà d'aquestes qüestions en principi no tenim més criteris per decidir i dependrà de cada cas.

4.3.5.2. Escalable

En el cas de Escalable, pot ser parcialment implementat per un tangible (fent de Modificador de la mida o fent de nansa rotant) però l'experiència ens diu que els dits són

molt intuïtius per a aquesta labor. Fent servir dos Nanses també faran indubtablement la labor dels dits però cal tenir en compte que no cal abusar dels tangibles donat l'espai físic que ocupen.

4.3.5.3. Deformable

En el cas de Deformable, ho podem fer amb dits o amb nanses. Més enllà de del problema típic dels tangibles hi ha el problema de la distància dels dits, que pot ser determinant. No tenim massa experiència en aquest rol, així que no podem indicar a priori cap direcció.

4.3.5.4. Polsador i Commutador

El Polsador ja és un cas més complex: tècnicament tenim la possibilitat d'usar dits i Colpejadors. Veiem-ne algunes observacions:

- Els dits necessiten d'indicadors visuals que ocupen "lloc digital".
- Els colpejadors virtualment no ocupen lloc ja que just després es poden retirar.
- Els dits es detecten malament amb ReacTIVIsion.
- Els colpejadors necessiten estar indicats amb un dibuix o forma prèviament.
- Els colpejadors poden ser usats fora de context i per tant erròniament.
- Una àrea receptora d'un dit pot ser bastant més petita que la d'un tangible.

El mateix podem dir pel Commutador; l'única diferència és que aquí el tangible sí que ocupa espai quan està posat.

Podem suposar que la solució a aquests dos problemes pot dependre de si un estat o una acció és usual o no en el sistema. L'ús de tangibles preestablerts amb indicacions té l'avantatge de alliberar la taula de receptors de dits que poden ser potencialment mal detectats i que fan ombra a altres coses i l'inconvenient de haver de disposar d'un gran nombre d'objectes físics que es poden perdre, desordenar i pesats de portar.

4.3.5.5. Manipulable de propietat

El Manipulable de propietat no té gaire color. Per pròpia experiència la comoditat és molt més gran util·litzant un manipulador. Per a utilitzar dits podem dissenyar alternatives a partir d'objectes digitals Traslladables o Escalables, que podrien ser més petits i discrets. Un exemple n'és el control de volum del TPlayer, que a l'estar a prop del selector de cançons queda és recollit i s'evita tenir 2 tangibles diferents els quals només un n'és Nansa sobre un objecte Traslladable.

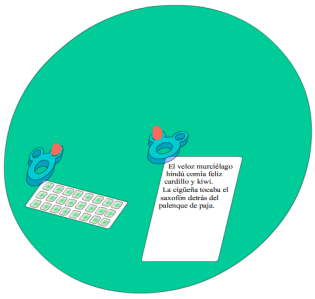


Figura 4.9.: Connexió física entre dos tangibles.

La connexió física entre dos tangibles es fa amb dos tangibles petits emparellats que encaixen amb els tangibles de tipus "nansa d'aplicacions".

4.3.5.6. Dibuixable

Finalment el Dibuixable, que es pot fer servir amb els dits o un Pinzell, en tot lo relacionat a gestos, encerclar, escriure, etc... apreciarem molt l'avantatge dels dits en tant que mida d'aquests. Els dits poden passar entre una gran quantitat de tangibles per rodejar-los per exemple. Pel que fa el pinzell, la funció més semblant a la metàfora original serà adequada, sobretot pel que fa a les seves formes físiques.

4.4. Relacions entre elements

Les relacions o connexions entre diferents objectes, lògics o físics es poden abordar des de moltes bandes. Hi ha moltes formes de enllaçar, emparellar, agrupar,... elements entre sí. L'objectiu d'aquesta secció no és doncs llistar-los tots sinó comentar un parell de reflexions que han sorgit al llarg de dissenyar els programes del TDesktop.

4.4.1. Relacions físiques

Com ja hem apuntat abans, la forma dels tangibles pot ser determinant per a construir part de la interacció que utilitzarà l'usuari. Per exemple podem fer que dos tangibles encaixin entre si, o fins i tot que tinguin un forat on es puguin posar connectors. Un exemple en pot ser aquesta idea que vam tenir en projecte per a emergir al món real el concepte de pipe del sistema operatiu (Figura 4.9).

Tot i que pot semblar una mica complicat, les relacions físiques entre tangibles poden ser molt útils i sobretot, donen un sentit físic a una restricció del sistema, de manera que no deixen confondre a l'usuari.

4.4.2. Relacions lògiques

Podem fer que les relacions que abans les fèiem de forma física, tinguin una presència lògica. Aquestes relacions poden aparèixer en forma de connectors per exemple entre els dos objectes.

Les formes de connectar dos objectes, a diferència de les relacions físiques, al no ser immediat es pot fer de diferents formes: utilitzant els dits o fent servir les distàncies per determinar quins objectes s'enllacen.

Pel que fa a les formes d'enllaç depenent de les distàncies , aquests enllaços es poden calcular només quan s'incorporen objectes a la taula o sempre que es mouen. Un exemple del segon cas seria el ReacTable[21] on els enllaços canvien cada cop que canvia la configuració particular dels objectes de la taula.

Aquest tipus de relacions permeten el canvi molt constant, en canvi penalitzen aplicacions on la configuració dels enllaços és important i es vol donar llibertat per posar els objectes en qualsevol lloc.

Per aquest motiu al TurTan hem fet servir enllaços que queden determinats només quan es posen els objectes a la taula, així aconseguim mantenir la formació encara que es vulgui apartar els objectes per a veure el màxim de projecció possible.

4.5. Restriccions

4.5.1. Límits dels tangibles

Cal considerar a l'hora de dissenyar la interacció amb la taula, que els tangibles són objectes físics en principi inanimats. Això que podria semblar una obvietat és extremadament important si un no es vol complicar la vida i complicar la de l'usuari.

Acostumats[5] als sistemes WIMP, tendim a pensar en reemplaçar nombroses funcions amb tangibles ja que aquests són molt còmodes alhora de controlar certes magnituds. Cal tenir en compte que si mai volem moure alguns objectes digitals a traves de la taula, mai podrem moure els tangibles associats.

Això és especialment greu si els tangibles són dependents d'una àrea, ja que si es mou aquesta àrea d'influència l'usuari no sabrà del cert si la funció que desenvolupa es la mateixa o no. En qualsevol cas, minimitzar d'incidència dels tangibles dependents d'àrees que tenen la capacitat de moure's es mostra com una bona pràctica.

D'altra banda si que hi ha tècniques per a simular l'extracció d'un tangible de la taula, per exemple fent ús d'un retorn visual que ens indica la si presència física del tangible transcendeix al món digital.

[5] O *mal* acostumats.

4.5.2. Nombre de tangibles i dits a la taula

Cal dir que a nivell de ReacTIVIsion no hi ha cap límit pel que fa el nombre de tangibles i dits detectats. Per altra banda , a les proves que hem fet amb la taula hi hem trobat certes dificultats, per exemple en la detecció dels dits, rarament poden ser detectats els cinc dits d'una mà alhora, a causa de la pròpia ombra de la mà.

Altre cop sembla molt obvi, però la quantitat de dits d'un usuari és limitada, i cal tenir-ho en compte. Una altra restricció és que els dits d'una mateixa mà no es poden allunyar gaire i per tant no se li pot exigir a l'usuari que posi els dits massa allunyats.

Una altra restricció òbvia però necessària és que els tangibles ocupen espai. Com que han d'estar en contacte amb la superfície de la taula no es poden apilar ni solapar. Per tant hem de preveure que:

1. Dos tangibles poden estar tant propers com la seva geometria particular els hi permeti.
2. El nombre de tangibles a la taula és tants com hi càpiguen.
3. Els tangibles poden impedir la lliure interacció de l'usuari (mitjançant un altre tangible o un dit)

4.6. Interpretació de les dades

4.6.1. Control de les magnituds(filtres del llenguatge)

Les dades al sistema ens arriben en les seves unitats físiques o equivalents. La majoria de cops usarem directament aquestes dades , ja que és el que realment ens es útil casi sempre: la posició real del tangible ens servei per a traslladar objectes i el seu angle per pintar bé el retorn visual, per exemple.

Però altres vegades, com per exemple el cas de Cargolar, les dades tal com ens arriben no tenen massa sentit per elles mateixes ja que no representen el que l'usuari vol comunicar.

Centrem-nos en el cas de Cargolar. En aquest, hem dit que la variació de la propietat que l'usuari vol controlar és *proporcional* a les diferències de l'angle del tangible.

4.6.1.1. Discretització

El primer cas en que voldríem filtrar l'angle, seria per exemple per fer-ne una discretització. El cas típic seria un selector de diferents opcions. El filtre aplicat (d) seria el descrit a la fórmula

$$d\left(x\right) = \lfloor \frac{\alpha m}{2\pi} \rfloor$$

on m és el nombre d'elements entre els que triar, α és l'angle del tangible en radiants i $\lfloor x \rfloor$ una funció que ens torna la part entera de x.

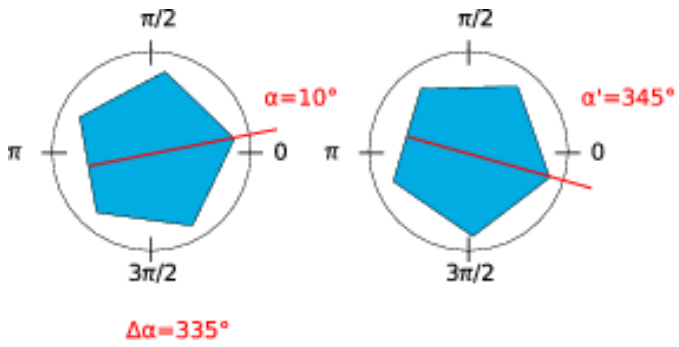


Figura 4.10.: Mala interpretació del gir d'un tangible.

4.6.1.2. Trobar el gir real

Un segon cas típic és el de intentar trobar el gir real fet per l'usuari: donat que ReacTIVIsion dóna l'angle absolut del tangible, a vegades és difícil saber a priori el gir més probable (Figura 4.10). Per a il·lustar aquest concepte posem-ne un exemple: Un usuari manipula l'angle d'un tangible, primer està a 10° i disminueix 25°. Per a ReacTIVIsion la diferència d'angle seria:

$$\Delta\alpha = \alpha\prime - \alpha$$

I per això ens donaria una diferencia de 345° quan segurament en realitat només és de 25. Per a determinar la diferència d'angle real hem de fer:

```
S := signe(diferència)
si absolut(diferència) > pi llavors
    diferència := (S*pi-absolut(diferència))
fi si
```

4.6.1.3. Escalat lineal

Finalment podem escalar els valors que ens donen. Ho podem fer linealment o de forma no lineal. Linealment no té massa secret:

$$\Delta\varepsilon = A\Delta\alpha$$

Així la variació de la nostra propietat ($\Delta\varepsilon$) es veu escalada directament proporcionalment amb un factor A.

4.6.1.4. Escalat no lineal

El tipus de filtre anterior és molt bàsic i per això no ens hi entretindrem. Els realment interessants són els filtres no directament proporcionals. En concret els que *exageren*

les velocitats de modificació: si l'usuari gira molt lentament la propietat es modificarà molt poc. Si l'usuari gira molt de pressa la propietat es modificarà encara més de pressa(Figura 4.11).

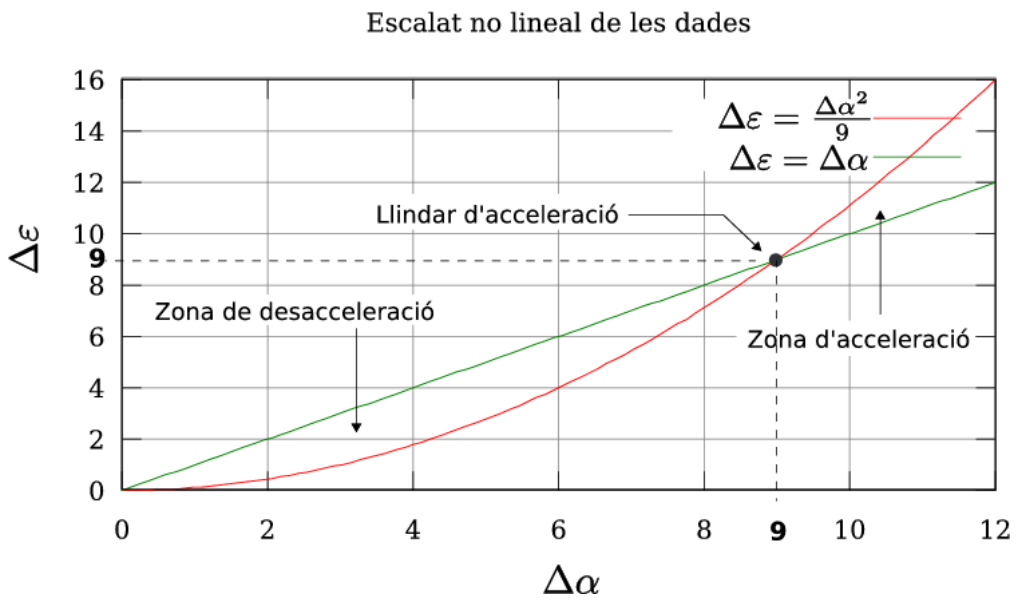


Figura 4.11.: Escalat no lineal.
Exemple de l'escalat no lineal de les dades. A la part de l'esquerra podem veure una zona de desacceleració de les diferències: la sortida es modificarà molt més apoc a poc que l'entrada; a la part de la dreta, una zona d'acceleració: la sortida es modificarà molt més ràpid que l'entrada; al mig veiem el punt d'intersecció: és el llindar d'acceleració.

Ara mirarem diferents funcions per a filtrar les dades d'una forma similar a l'anterior, per a comparar-les i veure què tal es comporten.

La primera que se'ns pot acudir és una cònica: *la paràbola*.

$$\Delta\varepsilon = (\Delta\alpha)^n$$

on $n = 1, 2\ldots$ és el grau de la paràbola. Una comparació de les diferents la podem veure a la Figura 4.12. Com podem veure, l'acceleració és realment molt gran, cosa que fa que de seguida es dispari. Com que la intersecció amb $\Delta\varepsilon = \Delta\alpha$ es produeix al punt $\Delta\alpha = 1$qualsevol variació de més de 1 grau s'accelera. Un grau de variació és una variació molt petita i resulta incòmoda. Per a canviar aquesta intersecció i desplaçar-la a un llindar més gran, hem de canviar la inclinació. Això ho podem fer per exemple així:

$$\Delta\varepsilon = \frac{\Delta\alpha}{l^{n-1}}^{n}$$

on l és el llindar de acceleració. Quan la variació $\Delta\alpha > l$ llavors s'accelera , si en canvi $\Delta\alpha < l$ llavors es desaccelera proporcionant més precisió. Llavors només cal calcular amb tests als usuaris, el valor òptim de l.

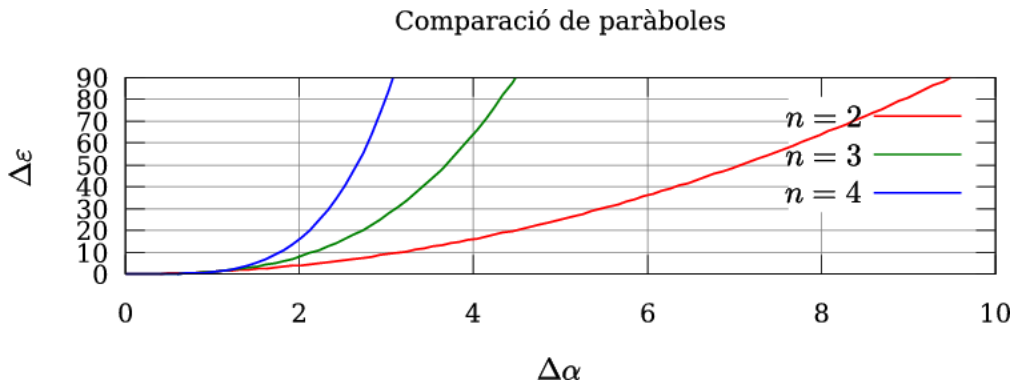


Figura 4.12.: Comparació dels diferents graus de les paràboles.
Les gràfiques corresponen a $\Delta\varepsilon = (\Delta\alpha)^n$.

Presentem ara un altre filtre d'acceleració-desacceleració que hem tingut en compte: *l'exponencial*.

$$\Delta\varepsilon = b^{\Delta\alpha}$$

on $b > 1$ és la base de l'exponencial. Com podem veure a la Figura 4.13 aquest filtre és més difícil de fer servir, ja que el tall de $\Delta\varepsilon = \Delta\alpha$amb $\Delta\varepsilon = b^{\Delta\alpha}$ no és gens immediat. Per aquest motiu el llindar d'acceleració no es pot ajustar fàcilment.

Aquest filtres els podem aplicar també en magnituds com la diferència de posició dels dits i dels tangibles. En aquest cas ens podria servir per el mateix argument de precisió-comoditat amb la forma de acceleració-desacceleració:

$$\Delta X = f(\Delta x)$$
$$\Delta Y = f(\Delta y)$$

4.6.2. Control d'errors

El ReacTIVIsion fa un filtratge dels esdeveniments d'entrada per a detectar errors de processament de la imatge. Per exemple desaparicions i aparicions sobtades de tangibles o dits a la taula, vibració de les posicions...

A nivell d'aplicació cal tenir en compte la detecció de tangibles fantasmes[6] i pèrdues de xarxa, ja que la connexió amb ReacTIVIsion no és segura i la pèrdua de paquets no és rara.

[6] Per tangible fantasma podríem entendre, per exemple, un tangible que es treu de la taula sense haver-se posat.

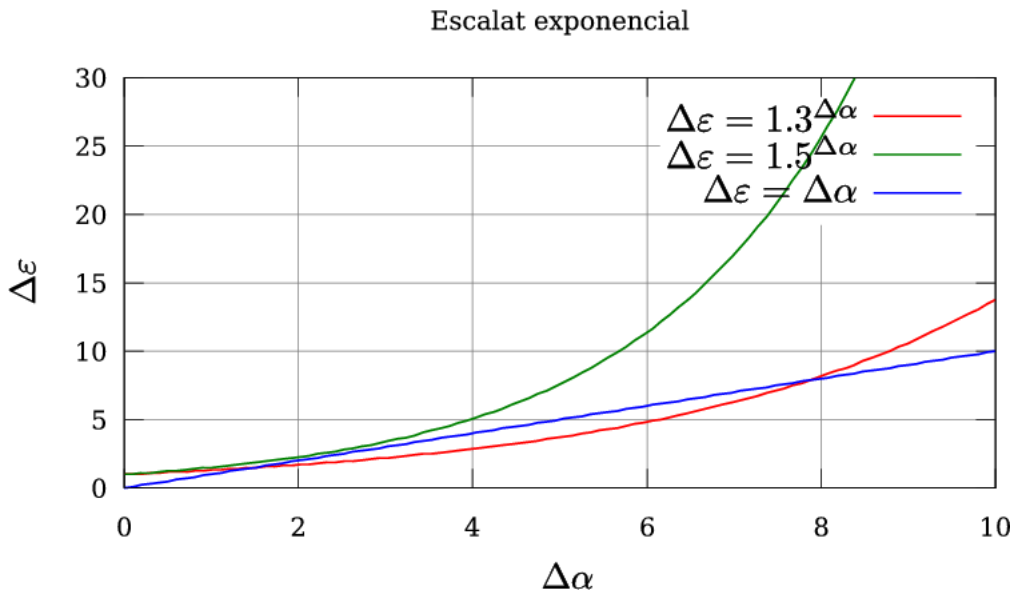


Figura 4.13.: Comparació de diverses exponencials.
Vegi's que no totes les exponencials tenen zona de desacceleració ($b = 1.5$).

4.7. Els Widgets

Aquí veurem uns quants widgets fets servir per les aplicacions, i en comentarem el seu model d'interacció.

4.7.1. CurButton

CurButton és un widget que implementa Polsador (Subsecció 4.3.4.1). Només és sensible als dits.

4.7.2. TWiconMenu

És un selector d'accions representades per icones. S'activa amb un tangible implementant així Modificador d'estat (Subsecció 4.3.1.2). Llavors es desplega al seu voltant una llista d'icones que representen diverses accions.

Girant el tangible s'il.lumina cada cop una icona diferent. En aquest moment es comporta usant el rol de Manipulador de propietat (Subsecció 4.3.1.4) discretitzant els valors. L'usuari sel·lecciona una opció retirant el tangible de la taula. Llavors desapareixen les icones.

Les accions es duen a terme en el moment que es van il·luminant les diferents opcions (accions de previsualització) i en el moment de triar la definitiva.

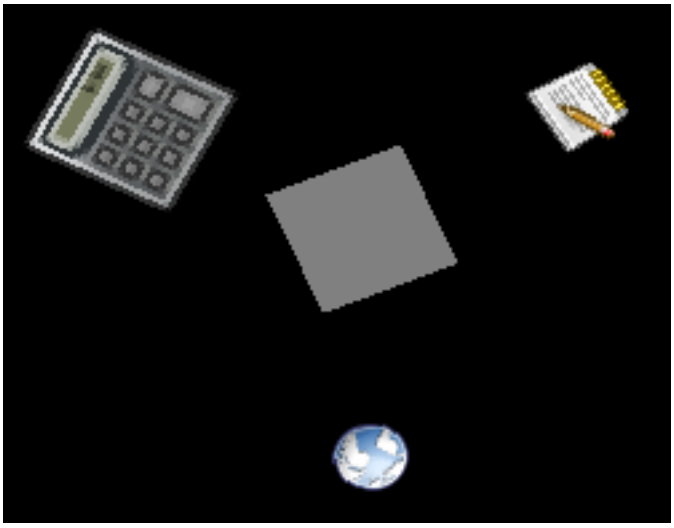


Figura 4.14.: TWiconMenu.
El veiem desplegat ensenyant una sèrie d'opcions (representades per una calculadora, un bloc de notes i un món). El quadrat del mig és el retorn visual del tangible.

Figura 4.15.: TWHiperMenu.

4.7.3. TWHiperMenu

Aquest widget es va pensar per a poder buscar en llargues llistes d'opcions en un TWiconMenu. Per a fer-ho es va idear crear un espai hiperbòlic al voltant del tangible amb les opcions. La opció triada en tot moment seria molt més gran que les altres i molt més opaca. Pel que fa a les altres coses s'hauria de comportar igual que el TWiconMenu.

4.7.4. widgetScalable

De fet es tracta d'una classe abstracta, però que implementa els rols Traslladable i Escalable (Subseccions 4.3.3.1 i 4.3.3.2 respectivament). Un exemple n'és el widget Ticon que pot implementar a petició aquests rols. Aquest es fa servir a l'aplicació TPhoto

representant les fotos.

4.7.5. Fidometer

Aquest és de fet una implementació de Manipulador de propietat (Subsecció 4.3.1.4) Discretitzant-ne els valors. El fem servir per a triar dins de llistes que no tenen una identificació icònica. Llavors la selecció és immediata per a poder saber què estem triant.

Podríem considerar-lo com un substitut del TWHiperMenu per a llistes d'elements no iconitzables amb funció de previsualització. O més senzill: és un menú cec. Podríem pensar en el dial d'una ràdio per exemple.

Un exemple entenedor és el del TPlayer: Amb aquest widget es seleccionen les cançons i per saber quina hem seleccionat, un text ens apareix en el reproductor i la música pels altaveus. No hi ha necessitat de icones.

5. Les aplicacions

Per Carles F. Julià i Daniel Gallardo

Tot seguit, farem un breu repàs a les aplicacions que hem elaborat per a poder experimentar amb TDesktop, detallarem el punt de vista de la interacció i molt per sobre el de la implementació.

En línies generals hem intentat que sigui Escalable tot el que sigui possible sempre que la interacció ho permeti. També hem intentat util·litzar tangibles genèrics en front dels específics (Veure 4.3.2 a la pàgina 79) sempre dintre dels límits raonables.

5.1. TManager

La adaptació de la complexa realitat dels sistemes operatius en sistemes més visuals ja ha estat tractada altres cops[2] i és un tema difícil de portar. Habitualment es tendeix a amagar la complexitat dels sistemes gràfics per a no atabalar l'usuari final, el contrari dels usuaris avançats, que es veuen obligats a recórrer a sistemes més flexibles com la línia de comandes.

TManager és part de TDesktop en el sentit més genuí, ja que desenvolupa funcions que atribuiríem en un sistema operatiu: la gestió de processos:

- Creació de nous processos
- Iconització
- Menú de les aplicacions
- Finalització de processos
- Informe de fallades

De fet, la gestió de processos més bàsica es fa a l'interior del TServer. La tasca del TManager és definir una capa entre l'usuari i la gestió dels processos, per a permetre a l'usuari de gestionar-los. En molts sentits és similar al concepte de la barra de tasques i llançadors del MacOS, o els accessos directes i la barra de tasques del Gnome o del Windows.

Figura 5.1.: TManager: vista inicial

De fet proveeix dos tipus d'objectes, el Llançador d'aplicacions, que representa una aplicació i la icona de procés, que representa una instància d'aquesta aplicació (o que simplement representa el procés). Aquesta distinció, que ja fem en altres sistemes operatius és un retrat de tal com funciona la computació avui dia. Tot i així cal reforçar-ne la distinció, ja que a vegades indueix a confusió.

A primera vista el TManager ens presenta una sèrie de llançadors d'aplicacions en forma de icones amb un símbol de llançador al mig. Aquests llançadors representen totes les aplicacions que pot executar l'usuari. Aquests són Traslladables i Deformables.

Per a executar una aplicació l'usuari diposita un tangible sobre del llançador desitjat. A partir d'ara ens referirem a aquest tangible com a nansa d'aplicacions. Automàticament s'executa el procés i apareix la icona de l'aplicació sota la nansa. Aquesta icona és Traslladable i Deformable i es controla també amb la nansa. La lògica del procés és la que descriu la Figura 5.2.

5.1.1. El menú

Una de les capacitats que donen més joc al TDesktop són els menús d'aplicació del TManager. Aquests , a part de proveir les opcions bàsiques de totes les aplicacions (Iconització, bloqueig i tancar) pot proveir-ne de l'aplicació mateixa.

El TManager escolta un canal de comunicació reservat *TPSCom* a l'espera de l'anunci de cada instància que vulgui comunicar-li opcions del menú, la seva icona o altra informació (com per exemple globus de text: Mirar Treball Futur 6.3.1.2 a la pàgina 114). Com que les aplicacions no tenen per què saber quina aplicació és el TManager, fan un missatge broadcast enlloc de enviar-li directament. Això és important ja que així no estem lligats a un sol gestor d'aquesta informació. Podríem dividir la responsabilitats en diferents programes per exemple.

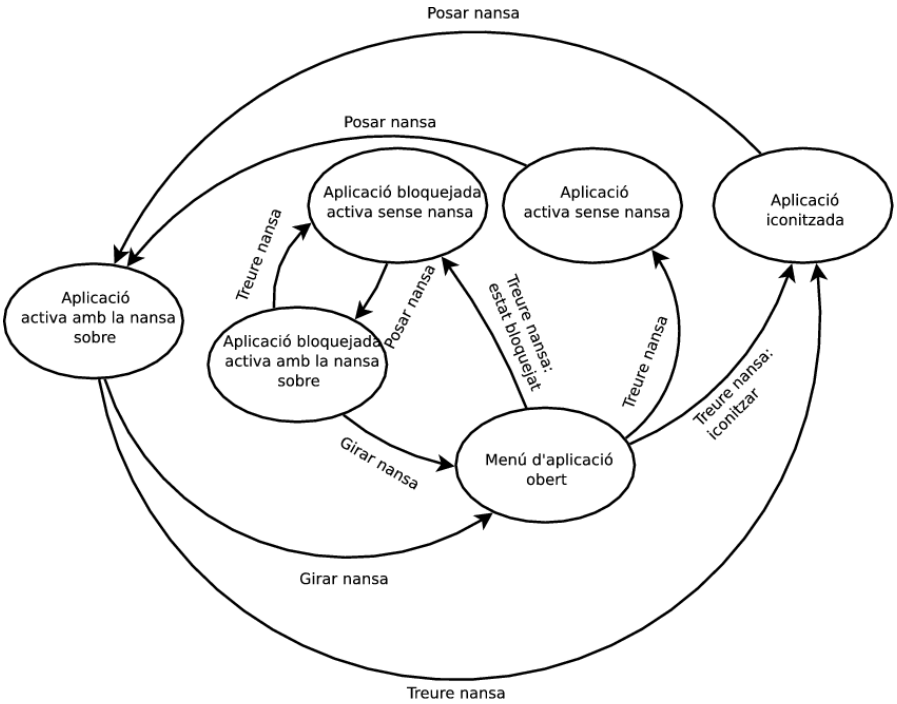


Figura 5.2.: Diagrama d'estats dels processos en relació a la nansa d'aplicacions.

Les dades enviades són exactament una llista de les icones de les opcions del menú amb un codi numèric que les identifica cadascuna. Quan l'usuari tria una opció, el TManager envia directament a l'aplicació, a través del mateix canal, el codi de la opció per a que el gestioni ella mateixa.

El menú no és definitiu, i la aplicació pot re-enviar-lo sempre que li sigui convenient. Com a exemple, el TPlayer fa ús d'aquesta característica per passar del mode amb interfície al mode iconitzat de reproducció. El mateix passa amb la icona del procés,en qualsevol moment es pot canviar.

Per a fer sortir el menú (es tracta de un TWiconMenu) cal girar significativament la nansa (aproximadament $\frac{\pi}{3}$). Per a seleccionar una opció només cal retirar la nansa. A no ser que s'hagi triat iconitzar, l'aplicació restarà activa a l'espera que s'hi torni a posar la nansa.

5.1.2. Les opcions per defecte

El menú d'aplicació sempre té com a mínim 4 opcions:

 **Iconitzar:** es tracta de la icona de l'aplicació amb ella mateixa feta més petita a

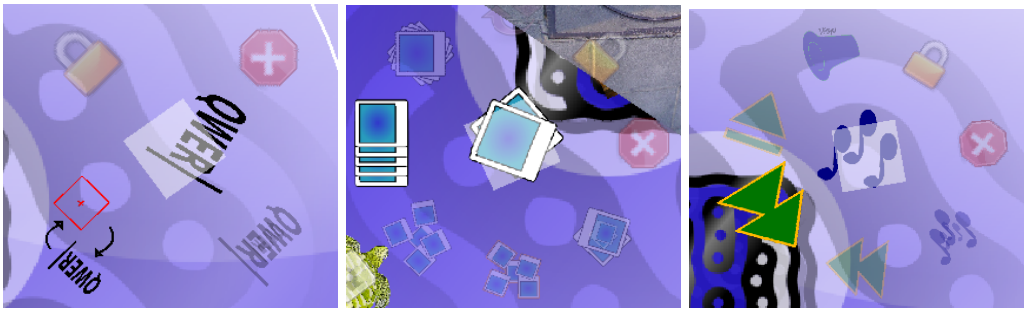


Figura 5.3.: Diferents menús a petició de diferents aplicacions

sobre. El que fa és iconitzar l'aplicació; tal com si haguéssim tret el tangible de sobre de la icona de procés.

 **Bloquejar:** inhabilita la capacitat de la nansa per a iconitzar l'aplicació al treure-la de sobre la icona de procés. Només podrem iconitzar a través del menú.

 **Desbloquejar:** desfà l'efecte de Bloquejar. A partir de llavors, la icona de procés es comporta normalment.

 **Tancar:** es tanca l'aplicació i es destrueix la icona de procés.

L'aplicació pot estar activa (es visualitza a la taula) o iconitzada (tan sols es veu la icona). Per iconitzar-la només cal retirar la nansa d'aplicacions o triar iconitzar del menú. Quan està iconitzada , a part de no veure's, ni és sensible a les àrees que ella mateixa hagi creat. Per tant simplement és com si hagués desaparegut.

La icona de procés pot estar *bloquejada*: això vol dir que no es pot iconitzar, encara que es retiri la nansa d'aplicacions. Per a bloquejar-lo cal triar al menú la opció bloqueja. Anàlogament es pot desbloquejar la icona a través del menú.

Per a tancar una aplicació només cal seleccionar la icona de tancar. Si mai l'aplicació es penja o surt inesperadament, un avís temporal pampalluguejant apareixerà sota de la nansa i desapareixerà la icona del procés de la taula.

5.1.3. Configuració dels llançadors

En el fitxer de configuració *TManager.config.xml* tenim les opcions dels llançadors: la seva posició, deformació , executable i icona.

<TManager_configuration_file>

```
  <Fiducial_id_for_killer>1</Fiducial_id_for_killer>
  <Fiducial_id_for_priority_screen_switcher>0<
  /Fiducial_id_for_priority_screen_switcher>
  <Fiducial_id_for_reseter>2</Fiducial_id_for_reseter>
  <Box_size_visual_feedback_elements>0.045</Box_size_visual_feedback_elements>
  <THlaunchers>
    <launcher X="0.761136531829834" Y="0.163386702537537" png_icon="..
    /Images/TClock/clock.png" program="../bin/TClock.exe">
      <matrix>
        <value>1</value>
        <value>0</value>
        <value>0</value>
        <value>0</value>
        <value>0</value>
        <value>1</value>
        <value>0</value>
        <value>0</value>
        <value>0</value>
        <value>0</value>
        <value>1</value>
        <value>0</value>
        <value>0.761136561632156</value>
        <value>0.163386702537537</value>
        <value>0</value>
        <value>1</value>
      </matrix>
    </launcher>
    <launcher X="0.13549892604351" Y="0.355581283569336" png_icon="..
    /Images/TPlayer.png" program="../bin/Turtan.exe"/>
  </THlaunchers>
</TManager_configuration_file>
```


Com podem veure la matriu de transformació *matrix* és opcional. Si no hi és present s'usaran només els paràmetres X i Y per a determinar-ne la posició. Si sí que hi és, els paràmetres X i Y són ignorats.

Quan el TManager es tanqui reescriurà la seva configuració al fitxer: així si l'usuari ha canviat les posicions i deformacions dels llançadors aquestes seran recordades posteriorment.

Figura 5.4.: Captura del TPhoto.

5.2. TPhotos

TPhotos és un visualitzador d'imatges. Aquesta és una aplicació típica sense interfície que posa objectes per l'escriptori. L'aplicació carrega a l'inici les imatges d'un directori i les situa al centre de la taula.

Aquestes imatges són Ticons i implementen Traslladable i Escalable. Així l'usuari les pot distribuir per la taula al seu gust.

A través del menú d'aplicació que es fa aparèixer amb la nansa d'aplicacions es poden redistribuir les fotos de diferents formes: centrades, en línia, o aleatòriament.

També tenia previst una funcionalitat d'agrupar les fotos i traslladar-les i redimensionar-les totes juntes[1]. D'aquesta manera es pot conservar la configuració desitjada i guanyar espai o rotar-les totes.

5.2.1. Origen les fotos

Les fotos que carrega el TPhotos estan situades en un directori definit pel programa. A l'iniciar-se simplement mira a tots els subdirectoris per trobar imatges PNG. Només carrega imatges PNG degut a la limitació en formats d'imatges carregables del TServer.

[1]En el moment de la redacció d'aquest document aquesta funcionalitat encara no està operativa.


Recarrega amb aquesta opció del menú d'aplicacions recarreguem el directori.

5.2.2. Redistribució de les fotos

TPhotos permet reordenar les fotos de moltes maneres:


Apila posa totes les imatges al centre de la taula amb un angle progressiu. A l'iniciar-se TPhotos, aquesta és la ordenació per defecte.


Arrenglera posa les imatges en una renglera (circular,donat que la taula és circular).


Aleatoritza distribueix les imatges de forma aleatòria per la taula.

5.2.3. Agrupació de les fotos

La idea de la agrupació de fotos és tractar totes les fotos com un sol objecte Escalable. Així es pot escalar una composició determinada de fotos a voluntat. Malauradament encara no funciona.


Agrupa Passa a mode agrupat per a manipular les fotos com un sol objecte Escalable.

5.3. TPlayer

TPlayer és un reproductor de música. Aquest és un gran exemple de programa amb interfície i que usa tangibles per a la seva interacció bàsica.

Per a dur a terme la seva interfície, s'han creat dos widgets nous: scrollAngle i ScrollBar; els quals hereten de curwidget, per tant només accepten dits. Aquests widgets són del tipus Slide en forma de semicercle i de barra respectivament i les seves funcions a les

aplicacions són controlar el volum de la sortida de so i navegar per la línia de temps de l'aplicació.

Com a resposta de la interacció, a part del so, podem trobar dos indicadors del tipus text, un que mostra les dades de la cançó que està reproduint i l'altre el temps restant per a que finalitzi.

El TPlayer té 2 modes de funcionament:

A.-Interfície Mode que requereix d'un tangible per a la seva interacció. Aquest tangible resol les tasques de navegació per la llista d'àudio quan el fem rotar, manipulació de la posició de l'aplicació quan el movem i reproduir o pausar quan el posem o el traiem.

Inconvenients Com que els tangibles no es poden moure sols, els dits no poden influir en el procés de traslladar l'aplicació. Si ho fessin, el tangible perdria l'enllaç visual de l'aplicació.

Rotació Donat que TPlayer utilitza text per a mostrar el què està reproduint, l'aplicació hauria de ser capaç de modificar la seva orientació, per dur a terme aquesta tasca, ens valdrem del tangible que té associat (serà el centre de rotació) i d'un dit que donarà les dades de l'angle de rotació.

B.-Iconitzat Mode on l'aplicació no dibuixa cap retorn visual ni requereix de cap tangible, sinó que es val del menú d'aplicació per a comunicar-se. Les opcions d'aquest mode són: reproduir / pausa, següent cançó, anterior cançó, tornar al mode Interfície.

Inconvenients A l'hora de navegar per la llista, es fa difícil ja que no hi ha manera de visualitzar quina peça s'està reproduint, a part del feedback acústic. Passa el Mateix quan s'acaba una cançó i comença la següent.

5.3.1. Llibreria d'àudio (TAudio)

Per a reproduir so, TPlayer utilitza TLib, una llibreria creada per a la manipulació del so dins del TDesktop. Aquesta llibreria, permet reproduir fitxers d'àudio utilitzant funcions del SDLMixer i gestionar i mantenir llistes de reproducció. De TAudio, cal destacar que té un binding de la llibreria LibMootag, per tant dels fitxers d'àudio, es pot extreure informació variada com el títol, artista, temps de reproducció,...

L'estructura de TAudio és molt bàsica (veure Figura5.6). Consta de dues classes encarregades de la reproducció i de tres encarregades d'emmagatzemar les meta-dades dels fitxers d'àudio.

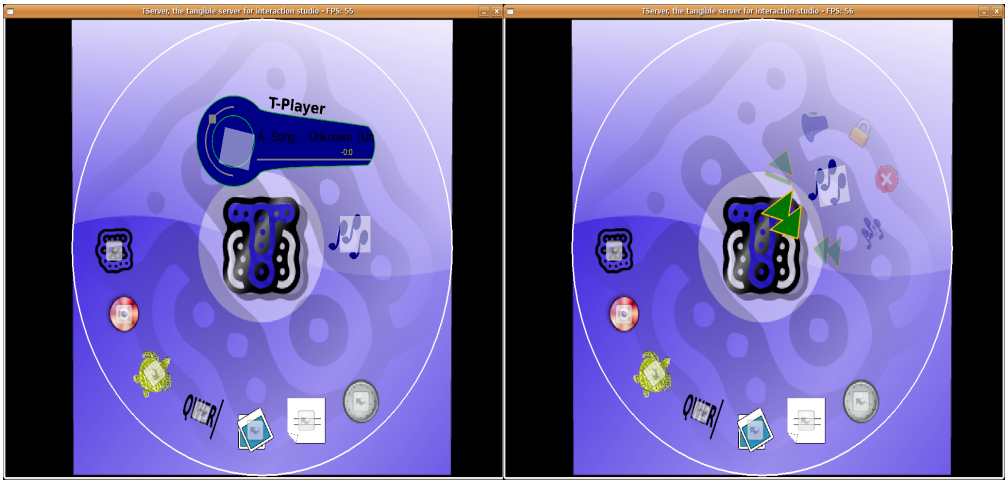


Figura 5.5.: Captura del TPlayer.

AudioPlayer És l'encarregada de reproduir i controlar els paràmetres de reproducció dels fitxers d'àudio. Aquesta paràmetres són: play, pause, volum, avançar / retrocedir dins del fitxer d'àudio i repetir.

PlayListPlayer S'encarrega per mitjà d' AudioPlayer de reproduir llistes de reproducció, gestiona la cua de cançons quan una finalitza, posa a reproduir la següent. Conté la opció de reproduir sense fi.

Song Classe que conté totes les dades d'una cançó, des de la url fins al temps de duració.

TagData Genera tot el contingut de Song a partir d'una url.

PlayList Llista de cançons amb funcions d'exploració i gestió.

5.3.2. Gestió de Llistes

TPlayer, incorpora mecanismes per a la gestió de llistes de reproducció, el problema és que al no haver definit cap sistema de navegació per fitxers, aquestes funcions en part no s'utilitzen.

El programa actualment, llegeix tots els fitxers d'àudio que conté el fitxer *../Music/* i en crea una llista de reproducció, fins aquí no és gaire, però la principal característica és que l'emmagatzema en fitxers xml per a no haver de tornar a llegir els arxius en posteriors execucions. Això permet tenir llistes de reproducció vinculades a fitxers d'àudio que es troben ubicats en directoris no predefinits.

Exemple de llista de reproducció en xml:

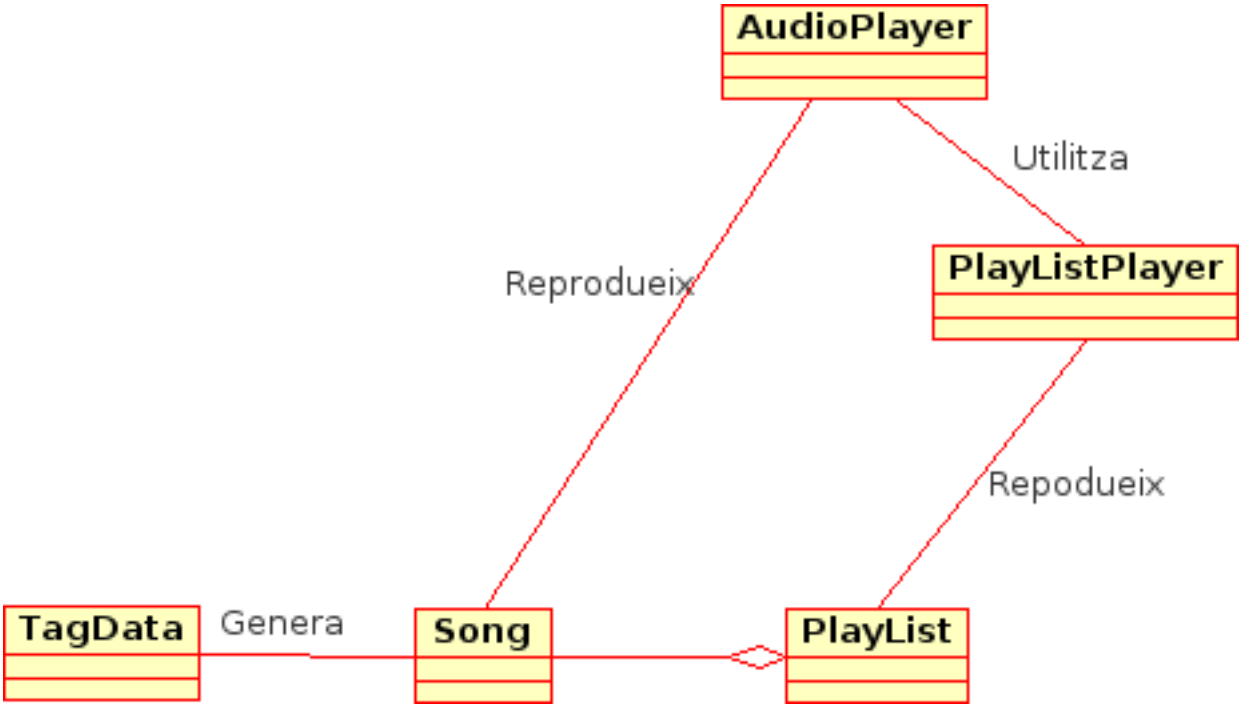


Figura 5.6.: Esquema de TAudio.

```
<PlayListFiles>
<Songs>
<Song Title="Hongroise les doigts"
      Path="../Music/01 - Hongroise les doigts.mp3"
      Artist="La Goutte au Nez"Album="Ouverture facile"
      Year="2005"
      Track="1"
      Genere="Blues"
      Image="../Images/music.png"
      Length="139"/>
<Song Title="Ouverture facile \u00c3\u00a0 la con"
      Path="../Music/02 - Ouverture facile \u00e0 la con.mp3"
      Artist="La Goutte au Nez"
      Album="Ouverture facile"
      Year="2005"
      Track="2"
      Genere="Blues"
      Image="../Images/music.png"
      Length="143"/>
<Song Title="Salsa Malikum"
```

```
      Path="../Music/12 - Salsa Malikum.mp3"
      Artist="La Goutte au Nez"
      Album="Ouverture facile"
      Year="2005"
      Track="12"
      Genere="Blues"
      Image="../Images/music.png"
      Length="363"/>
</Songs>
</PlayListFiles>
```

5.4. TClock

TClock, és la primera TAplic que es va fer, a estil xclock[2] del servidor X de linux. Això no vol dir que conceptualment sigui la més senzilla, sinó que és bastant completa pel que fa a la interacció.

L'aplicació en si, és un rellotge que apareix al centre de la pantalla sense cap mena d'indicador de control. Es tracta bàsicament d'una aplicació per a que l'usuari entengui el funcionament de la transformació dels widgets amb els dits i experimenti amb ella.

TClock, conté altres modes més avançats d'interacció, que són el canvi horari i l'alarma. Per a dur a terme l'acció del canvi d'hora, l'aplicació disposa d'un widget especial que és capaç de fer interpretacions diferents de les dades depenent del mode en que es troba:

- Normal

  En aquest mode el TClock només mostra l'hora i s'hi poden aplicar transformacions amb els dits. Si hem configurat una alarma, apareixeran agulles que marquen quan ha d'activar-se i quan sigui l'hora, sonarà fins que es faci alguna acció amb el rellotge o s'apagui per mitjà del menú d'aplicació.

- Set Up

  En aquest estat, amb el dit fem girar les agulles del rellotge, per a posar-lo en hora. Aquesta nova hora és permanent per a aquesta instància de TClock, no modifica l'hora global del sistema.

[2] http://www.xfree86.org/4.2.0/xclock.1.html

Figura 5.7.: Captura del TClock.

Al utilitzar el dit per a modificar l'hora, òbviament, no es podran aplicar transformacions al rellotge, per tant aquest mode inhabilita les funcions del mode normal.

- Alarma

  Afegeix una alarma al rellotge. La forma d'introduir dades és igual que en el mode Set Up, inhabilitant també, les funcions del mode estàndard. Un cop s'ha sortit d'aquest mode, queden les agulles de l'alarma de forma permanent.

Tots aquests modes són accessibles per mitjà del menú d'aplicació que es desplega de la nansa associada.

Tenir diversos TClock oberts pot servir per a tenir diferents alarmes (funcionen encara que estiguin iconitzats) o per tenir diferents fusos horaris per exemple.

5.4.0.1. Modificació horària

Per a variar l'hora del rellotge, s'utilitza l'angle entre el centre del rellotge, i el dit que duu a terme l'acció (Figura 5.8). Un cop es té aquest angle, es pot extreure l'increment de temps modificat per mitjà d'una regla de tres.

Vàrem optar pels dits per canviar l'hora, perquè volíem representar la metàfora de moure les agulles, com passa a la vida real en segons quins rellotges (més aviat antics). Una altra opció, seria fent-ho amb un fiducial, computacionalment més fàcil, però requeriria d'una figura innecessària que implicaria una formació prèvia[3].

[3] Se li hauria de dir a l'usuari quin fiducial utilitzar i a on.

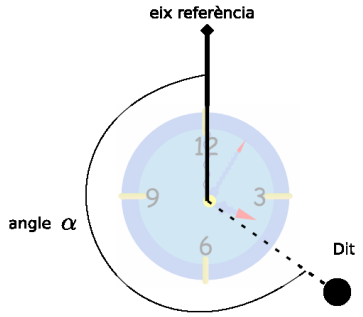


Figura 5.8.: Angle canvi horari.

5.5. TKeyboardManager

Aquesta aplicació, proporciona dos mètodes d'entrada de text per a les aplicacions que ho necessitin, normalment widgets de text.

Un dels principals dubtes a l'hora d'utilitzar eines d'entrada de text, va ser com portar-lo a multi-usuari. Donat que ReacTIVIsion no pot distingir els dits de diferents usuaris[4], la solució és utilitzar més d'una aplicació d'entrada de text, una per a cada usuari.

El fet d'utilitzar més d'una entrada de text, ens plantejava un problema molt més gran que encara no hem solucionat, però tenim varies propostes(veure 5.5). Aquest problema sorgeix a l'hora de situar en cada aplicació un cursor per a cada instància de teclat que hi hagi executant-se, de tal manera que cada usuari escrigui allà on vulgui i no on s'ha seleccionat per ultima vegada.

Actualment, no és recomanable obrir més d'una aplicació teclat a la vegada perquè el problema de la posició del cursor d'escriptura no està resolt. Com hem dit abans el teclat serveix per a introduir text en els widgets que ho requereixin i per a seleccionar on es vol escriure, només cal tocar amb el dit l'àrea d'escriptura i el sistema ja sabrà on ha de re-dirigir el text d'entrada.

Per a dur a terme la coordinació entre totes les àrees de text i el teclat, s'utilitza ListenerKeyboard (veure 3.2.4.1 a la pàgina 63) que proporciona un sistema de comunicació entre les diferents aplicacions i el teclat.

El seu funcionament es basa en anar passant-se un token entre aplicacions de la següent manera:

[4]Tampoc quin dit de la mà és.

1. S'executa l'aplicació teclat, per defecte no té associada cap àrea de text, totes les dades que s'introdueixin no seran processades.
2. Al posar el dit dins de l'àrea de text, aquesta genera un event intern que demana el token d'escriptura.
3. Aquest event intern es tradueix a un event de sistema encapsulat de la següent manera : [idAplicació[idAreatext]]
4. quan el teclat rep l'event, re-dirigeix totes les dades generades a [idAplicació[idAreatext[dades]]].

Incongruències de les TUI Com s'ha analitzat a l'apartat de les GUI (veure 1.2.2.2 a la pàgina 21) el teclat no deixa de ser una eina de control remot de l'aplicació per molt integrat que pugui estar dins de la superfície tangible[5]. Per tant s'ha de tenir molt en compte a l'hora d'utilitzar àrees de text perquè podríem estar tornant sense voler a la metàfora de les GUI.

Per aquesta raó el teclat no disposa de cap tecla de navegació per tal de no caure en l'error d'implementar funcions de navegació dins de cada aplicació.

TKeyboardManager posseeix dos modes d'entrada de caràcters els quals es detallen a continuació:

5.5.1. Teclat

El mode teclat, és el que s'obre per defecte al executar l'aplicació. Aquest mode ens presenta un teclat del tipus QWRETY envoltat d'un marc translúcid que ens permet escalar, rotar i traslladar l'aplicació.

Per a estalviar el pas de missatges entre l'aplicació i TServer s'ha optat per utilitzar una sola àrea tangible en front de 40 o 50 àrees que correspondrien als widgets botó. Per tant s'utilitza una àrea que interpola la posició real del dit respecte 3 punts coneguts del teclat. D'aquesta manera és pot conèixer quina tecla s'està picant encara que s'hagin modificat les coordenades del teclat.

Pel que fa a la interacció, tot i no haver fet proves amb usuaris, hem pogut determinar unes quantes mancances als teclats virtuals d'aquest tipus:

- Aquests teclats, no disposen d'un retorn tàctil. Fet que provoca una sensació estranya a l'hora de picar les tecles.
- Com que no disposen de tecles físiques, és casi impossible teclejar sense mirar el teclat.
- No se sap si l'usuari té un dit reposant sobre una zona de la pantalla, o té la intenció de que s'escrigui moltes vegades el caràcter d'aquella tecla.

[5] L'únic que s'ha fet és apropar més el control remot a l'aplicació.

- Pot ser que un altra aplicació estigui per sobre del teclat inhabilitant una part d'ell.

Les solucions que hem anat resolent són les següents (ordenades per problemes sorgits)

- El retorn tàctil, no el podrem suplir, però si que hi hem afegit un retorn visual. Quant un dit està sobre una tecla aquesta és de color vermell.
- Passa igual que abans, no hi ha més solució que la pràctica i el costum.
- Per aquest motiu, no permetem que una tecla pugui estar enviant la senyal de clic contínuament, només una vegada quan aquesta és alliberada.
  - Aquest fet combinat amb el retorn visual, ens permet passar el dit pel teclat per tal d'anar il·luminant la tecla desitjada sense polsar-ne realment cap. És una ajuda bastant bona per quan el teclat és suficientment petit com per tenir problemes de tocar dues tecles alhora.
- L'única solució per evitar l'encavalcament del teclat amb altres aplicacions, és dotar al teclat de privilegis de pintat sempre per damunt de les altres aplicacions.

5.5.2. Reconeixedor textual

S'hi accedeix per mitjà del menú d'aplicació desplegat per la nansa. Aquest mode presenta una capsa quadrada on l'usuari pot escriure els caràcters un per un amb el dit (és Dibuixable - Subsecció 4.3.4.4).

L'única diferència amb el mode teclat és el mètode d'entrada de dades, continua essent un problema[6] el fet d'haver-hi una deslocalització de l'àrea d'entrada i l'aplicació (metàfora de les GUI).

Aquest mode no es pot escalar, rotar ni moure. Per tant planteja els següents problemes:

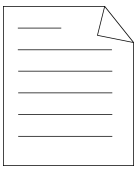
- No és el mateix des d'on es comença a dibuixar el caràcter. Sempre s'haurà de fer des de la posició per defecte.
- Al no poder moure l'àrea, pot ser que eclipsi el lloc on estem entrant dades.
- Continua essent dissenyat per a un sol usuari. Concretament per a un sol dit.

Sobre els mètodes de lectura dels caràcters, se'n pot trobar més informació a l'Annex F a la pàgina 149.

[6] No tant estricte com el del teclat.

Figura 5.9.: Captures del TKeyboardManager.

5.6. TWriter



El TWriter és un editor i visualitzador de text. És Escalable, rotable i permet fer les operacions més bàsiques d'un editor:

- Crear un document nou.
- Carregar un fitxer de text.
- Modificar un document existent.
- Desar un document

Per a fer qualsevol de les opcions es fa a través del menú d'aplicació. Per carregar, no obstant, es crea un Widget Fidometer representat per una icona per seleccionar l'arxiu d'una carpeta determinada.

Els documents que visualitza TWriter, són documents composats per un títol i un cos els quals es visualitzen en dues parts (títol i cos). Per a la formatació del text, cal dir que ara per ara només s'accepta text pla per tant l'únic caràcter especial que accepta és un salt de línia.

Si el cos del document és massa llarg, llavors automàticament apareix una barra de navegació (scroll) associada al widget que mostra el text (en aquest cas TextBoxScrollable).

Tant l'àrea del títol com la del document són widgets que hereten de textArea, per tant poden escoltar els events d'entrada provinents del teclat.

Figura 5.10.: Captura del TWriter.

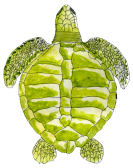
5.6.1. Emmagatzematge dels documents

Per a guardar els documents, hem creat una estructura xml que s'encarrega de desar-los de forma estructurada dins d'un directori que conté només fitxers del TWriter.

Els fitxers que crea són del tipus títol-TWR.xml i segueix l'estructura següent:

```
<TWriterDocument>
<Title>murcielago</Title>
<body>
  El veloz murcielago indú comia feliz cardillo y kiwi.
  La cigueña tocaba el saxofon detras del palenque de paja
</body>
</TWriterDocument>
```

5.7. TurTan



El TurTan és un llenguatge de programació tangible. Les instruccions són representades per tangibles que s'enllacen entre ells per a construir una cadena d'instruccions. El

Figura 5.11.: Captura del TurTan.

resultat, dibuixat per una amable tortuga, apareix instantàniament.

Per a una descripció concreta es pot llegir l'annex dedicat al TurTan (E a la pàgina 143).

6. Avaluació i treball futur

Per Carles F. Julià i Daniel Gallardo

6.1. Taula i programes

Com hem dit abans, aquest projecte no està enfocat a generar un producte final. Això és degut a que ens basem en suposicions sobre noves metàfores que hem anat extraient, per tant ara vindria un procés de millora de tot el que hem fet alimentat per tests amb usuaris no experts.

Aquests tests, ens permetrien de millorar la interacció de les aplicacions per a fer-les més fàcils d'entendre i veure les necessitats dels usuaris per a poder generar noves aplicacions.

6.1.1. Taula

Un punt important per a generar un producte final, és la taula. Actualment la taula està separada del ordinador i de les sortides de so. Aprofitant que és un taula bastant gran, es podria integrar tot a dins de tal manera que l'usuari final hagués d'endollar-la i ja estaria llesta per a fer-la servir (Figura 6.1).

El fet de posar-ho tot dins de la taula, comporta certs problemes:

- S'han de fer sortir controls i connectors bàsics per a l'ordinador (botó d'encesa, ports USB, ports d'entrada d'àudio,...)
- El mètode de calibració de la taula, hauria de ser automàtic. L'usuari no té perquè ser un expert en calibrar taules, com hem dit abans, només l'hauria d'endollar i fer servir.
- Posar-ho tot dins de la taula i que no augmenti de mida, no és obvi. Els objectes nous, no haurien de fer ombra a la projecció ni a les làmpades infraroges.
- La connexió a la xarxa es pot fer mitjançant wireless per a no tenir més d'un cable. Això comporta fer alguna aplicació per a gestionar la xarxa.

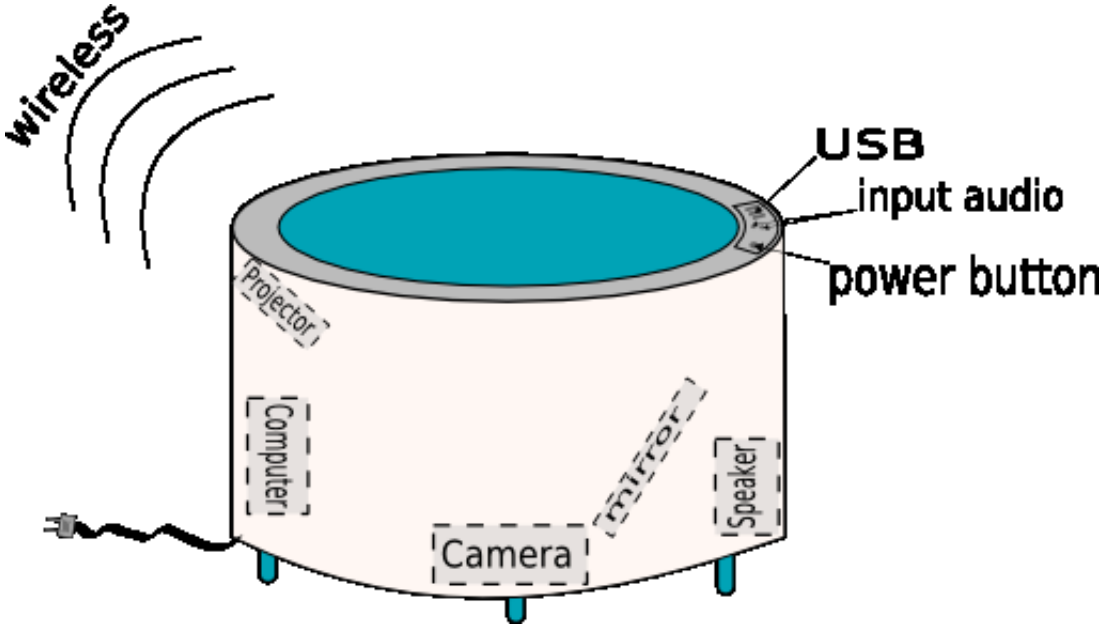


Figura 6.1.: ArtWork taula.

Figura 6.2.: Fiducial memòria USB.

6.1.2. Aplicacions o programes

Un punt important, seria la distribució de nous programes. Donat a que utilitzem el TManager com a utilitat per a llançar les aplicacions, aquest hauria de ser capaç d'acceptar-ne de noves (instal·lar-les) i d'amagar-ne les que no es volen fer servir o inclús eliminar-les (desinstal·lar-les).

Tornant a l'exemple del mediaBlocks (veure 1.3.2 a la pàgina 25), una opció seria distribuir fiducials amb el programa al seu interior (per exemple amb memòries USB) de tal manera que endollant el fiducial al port USB de la taula, el programa s'instal·laria i seria el mateix fiducial que duu incorporada la memòria USB qui llancés l'aplicació.

Una altra opció podria ser mantenir un dipòsit d'aplicacions tal com es fa ara per exemple

amb el sistema operatiu Debian[1] i derivats. La identificació de l'aplicació a instal·lar es podria fer a través d'un programa de cerca o a través d'un identificador a la taula, sigui un fiducial o un codi de barres[2] per exemple.

6.2. TDesktop

6.2.1. Utilització de recursos

Quan es porta una llarga estona utilitzant TDesktop, és fàcil de veure que la màquina virtual de mono triga un temps en alliberar memòria automàticament, per tant s'haurien d'aplicar mecanismes per a destruir objectes quan no s'utilitzin, sobretot buffers i matrius de transformació.

Aquesta remodelació en si no és gaire complicada, però aprofitant la compatibilitat de mono amb C, havíem pensat de traduir els mètodes que pesen més en quant a volum de càlculs a la CPU per a veure si seria una millora significativa o no. Concretament, serien els mètodes de detecció de col·lisions juntament amb la part de processat dels Missatges TUIO i les transformacions de matrius.

També es pot mirar de optimitzar la memòria en concret les transformacions amb matrius passant algunes classes a tipus, que permet al mono representar-ho com a valor i no com a instància; o sigui tractar les matrius com a floats més que com a una classe instanciable.

De totes maneres, TDesktop + les aplicacions funcionant funcionen be en qualsevol ordinador assequible d'avui en dia, provat a un pentium centrino a 1.7GHz 1gb de ram i targeta gràfica dedicada.

6.2.2. Comunicació

Donat que volíem fer un sistema experimental, varem utilitzar mono.remoting per a la comunicació ja que no requeria de grans configuracions i la seva programació és més fàcil. Però si es vol fer un sistema enfocat a l'usuari final expert[3], potser seria millor refer la part de comunicació.

El problema bàsicament rau en que el remoting és exclusiu de mono i limita que totes les aplicacions hi estiguin escrites. Tot i això és difícil trobar una forma multi-llenguatge de computació distribuïda que ens doni les mateixes facilitats (pas d'events a través de xarxa per exemple, o compartició de memòria).

Pel que fa la comunicació entre TAplics, una opció que s'ha de tenir molt en compte , és la utilització de D-Bus, un sistema per a la comunicació integrat dins de freeDesktop[4]. Si

[1] http://www.debian.org
[2] Segons fonts del ReacTIVIsion seria bastant fàcil implementar-hi el reconeixement de codis de barres, ja que existeixen moltes llibreries de codi obert que ja ho fan sobre imatges.
[3] Assumim un tipus d'usuari que vulgui desenvolupar aplicacions per a la plataforma TDesktop.
[4] http://www.freedesktop.org/wiki/

es portés TDesktop a aquesta plataforma, ens asseguraríem una integració millor entre aplicacions natives de TDesktop i no natives.

En aquest moment el TServer necessita d'un gestor de finestres per a executar-se, això consumeix recursos que no util·litzem. Podríem intentar que el TServer es pogués executar sobre del freeDesktop directament com si es tractés de qualsevol entorn d'escriptori més[5]

6.3. TAplics

6.3.1. TManager

El TManager té molt de camí per davant. No està mai gens clara quina és la millor forma de fer les coses, molts cops ens trobem amb usuaris que no entenen res de la seva interacció i en canvi d'altres que ho entenen a la primera.

6.3.1.1. millores gràfiques

Una millora gràfica immediata és intentar distingir millor els llançadors de les icones de procés. Això ho podem fer emfatitzant la condició de llançador o de procés (Proposta a la Figura 6.3).

Figura 6.3.: Noves icones del TManager

Amb aquestes noves icones (propostes) podríem diferenciar a simple vista entre llançadors i icones de procés.

També controlar de forma més eficaç la posició de les capes de les icones de procés seria desitjable. Actualment sempre apareix la icona de procés a sota de el llançador, cosa que desconcerta els usuaris.

6.3.1.2. Millores de la interacció

- La dicotomia Icona de procés ↔ Interfície. El problema es fa patent per exemple amb el TClock:

[5] kde, gnome, xfce...

El TClock té una interfície simple que és Traslladable i Deformable i per usar el menú d'aplicació no es fa des de la seva interfície sinó dels de la seva icona de procés. Per no trencar la coherència de les icones de procés, es podria enviar les coordenades d'aquesta a l'aplicació perquè s'integrés amb la interfície i s'usés la nansa d'aplicacions també per moure-la.

- El Globus de notificació: si una aplicació vol comunicar algun esdeveniment i no té interfície pròpia o està iconitzat, es podria implementar un sistema semblant al dels globus de notificació dels programaris d'escriptori als que estem habituats. D'alguna forma de la icona de procés hauria de sorgir el missatge que l'aplicació vol comunicar.

- El tangible de llançadors. Actualment els llançadors sempre estan a la taula ocupant lloc i no hi ha forma de reordenar-los. Es podria fer que els llançadors només hi siguessin si el tangible de llançadors estigués sobre la taula. També aquest podria proveir funcions de reordenar els llançadors i reposicionar-los.

6.3.2. TPhotos

El TPhotos és potencialment una aplicació molt important. Les possibilitats són infinites.

6.3.2.1. Millores

- Cal arreglar alguns errors causats per la multiplicació inversa de les matrius de TDesktop, aquest és un problema que anem arrossegant des del principi. Quan estigui arreglat, qualsevol redistribució automàtica serà virtualment possible.

- Cal també fer funcionar l'agrupament de fotos. El motiu del seu mal funcionament segurament és el mateix que el de l'anterior.

- Cal millorar la càrrega de imatges afegint-hi nous formats usuals com el JPG.

- Cal detectar la relació d'aspecte de la imatge i presentar-la correctament, actualment les imatges són quadrades.

- Altre cop la correcta distribució per capes. A vegades intentant modificar una foto (escalant-la) es feu afectada la foto per sota d'aquesta que no està visible.

- Noves formes de distribució : a través d'una línia traçada o sota de la icona de procés.

6.3.2.2. Noves possibilitats

- Classificació de fotos.

Es podrien classificar les fotos amb tags de diferents colors: l'usuari faria servir un tangible-tampó per marcar les fotos i encabir-les en unes certes categories. Després podria decidir mostrar només les fotos d'una certa categoria o distribuir-les en llocs diferents de la taula segons aquesta.

- Vídeo.

  Integració amb el vídeo al més pur estil Microsoft Surface.

- Amagar/Mostrar fotos

  Habilitar una forma de treure de la taula certes fotos mantenint-ne algunes altres: una forma seria amb un tangible eliminador de fotos.

6.3.3. TPlayer

Actualment, TPlayer és un reproductor d'arxius de música, suporta OGG i és capaç de llegir els tags de les meta-dades dels arxius d'àudio. Però és incapaç de gestionar llistes de reproducció degut a que no hem establert un sistema de navegació estable i que pugui ser utilitzat per qualsevol TAplic.

De moment hi han pensats alguns nous modes més per a TPlayer, a part del que hi ha fets(reproductor i el reproductor amagat), s'han estudiat els següents modes:

- Mode d'edició de llistes de reproducció: Aquest mode consistiria en utilitzar els fiducials com a portadors de les llistes de reproducció de tal manera que per a crear-ne de noves, només s'haguessin d'enllaçar dos tangibles a la taula com si fossin instruccions del TurTan.
  - Un dels principals avantatges d'això, seria que l'usuari podria disposar de la seva col·lecció de discs etiquetats amb fiducials de tal manera que per a fer-lo reproduir hauria de posar-los sobre la taula.
  - L'inconvenient principal és que només es podran enllaçar llistes d'àudio senceres, ja que no hi ha de moment una cerca interna de fitxers ben estudiada.
- Mode DJ (en pantalla sencera): Aquest mode consistiria en posar les funcions mes comuns que utilitzen els punxa-discos al TPlayer. Inicialment es tenen pensats posar dos fils de musica i mitjançant mètodes de fade anar passant d'un a l'altre.

Vídeo Tenim fetes unes noves funcionalitats per afegir streams de vídeo al TDesktop, però no estan en una versió estable. La pregunta que ens fem és si haurien d'estar associades al TPlayer o Dependre d'alguna altra aplicació.

La idea de tenir un reproductor multimèdia únic, no creiem que sigui la millor, TPlayer està pensat només per al so. Com ja s'ha dit, creiem que el vídeo, es podria associar més al visualitzador d'imatges TPhotos (un vídeo no deixen de ser imatges en moviment) llavors el comportament de l'aplicació no variaria, només que en segons quines imatges apareixerien uns controls de reproduir, pausa, endavant i endarrere.

6.3.4. TKeyboardManager

Aquesta és l'aplicació amb la que menys hem experimentat donat que no es cenyia a l'estudi que volíem fer. De totes maneres a l'annex sobre reconeixement de text F és pot trobar més informació al respecte.

6.3.4.1. Teclats multi-usuari

Com s'ha comentat abans (veure 5.5 a la pàgina 105) encara no s'han resolt els problemes d'utilitzar varis teclats per a diferents widgets de text al mateix temps, en aquest apartat donarem algunes opcions possibles:

- Teclat incrustat al widget

  Aquesta opció consisteix en afegir a cada widget de text, funcionalitats noves. De tal manera que quan es seleccioni per a l'entrada de text, automàticament es desplegui un teclat virtual a sota per a procedir a l'entrada de text. Aquest sistema, planteja problemes depenent de l'àrea de text. El desplegament d'un teclat suficientment gran (com a mínim que les tecles tinguin la mida d'un dit índex), implicaria l'anulació visual de tot el que hi hagi a sota d'ell ja que moure el teclat no tindria cap sentit perquè és perdria el vincle visual entre l'àrea de text i el teclat.

- Teclat associat a fiducials

  Aquesta solució, es basa a tenir un nombre limitat de fiducials dedicats només a l'ús del teclat. Respecte al cas anterior, afegeix la millora de poder desplegar el menú del teclat per tal de canviar de mode de teclat, cosa que a l'opció anterior no era possible.

  Tot i aquestes millores, implica fer àrees de text suficientment grans com per a que l'usuari no expert, pogués crear el vincle entre l'àrea de text i el fiducial teclat. Fet que limitaria la mida d'aquest widget i per tant de tot el conjunt de l'aplicació que el conté.

  Finalment, posats a trobar entrebancs, el fet d'utilitzar fiducials dedicats per a l'ús del teclat, creiem que no és del tot adient, ja que aquests fiducials no serien útils per a qualsevol altre aplicació, a part que generaria un excés[6] de tangibles innecessaris per a utilitzar TDesktop.

- Teclat com a aplicació i token virtual.

  Consisteix a situar una figura virtual a la pantalla associada al teclat. El seu funcionament seria el següent: al executar un aplicació teclat, apareixeria un teclat de color X i una figura del color X; per a introduir text, només caldria moure la figura virtual amb el dit a sobre d'una àrea de text de tal manera que es crearia

[6] En el cas de que les altres aplicacions també requerissin tangibles propis.

una associació visual fàcil d'entendre. Aquest sistema té els avantatges del sistema anterior i millora el problema de tenir fiducials dedicats.

Tot i així cal tenir en compte l'espai virtual que ocupa, que pot tapar la mateixa àrea de text. Per altra banda cal determinar com fer aparèixer i desaparèixer aquests objectes apuntadors. Certament no sembla haver-hi una solució immediata a aquest problema.

6.3.5. TurTan

6.3.5.1. Noves instruccions

Tot i que és l'aplicació més complerta que tenim, no es poden parar de trobar noves instruccions que serien molt interessants d'afegir. Primer de tot la situació ideal seria intentar abastar tot el llenguatge del Logo per a que tingui un reconeixement més estès.Però abastar tot el llenguatge Logo pot ser conceptualment molt difícil de portar a terme tot i que hi ha coses que sí que es poden implementar que ens hi acostarien.

Subfils Es tractaria de implementar el concepte de bloc. En un subfil es poden fer moltes operacions i no afecta al conjunt de les altres aplicacions.

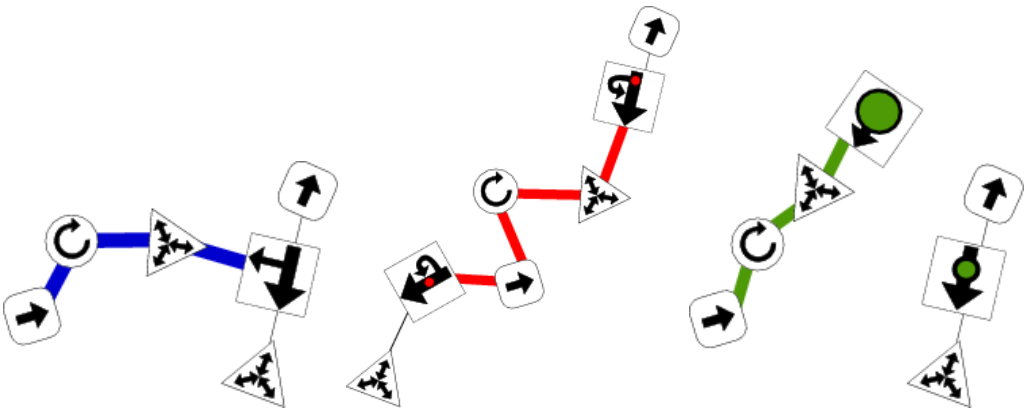


Figura 6.4.: Diferents estratègies per implementar el subfil o subrutina.

Oscil·ladors Modificadors dinàmics de les propietats de les instruccions. És un concepte molt semblant al dels modificadors del ReacTable. Això ens permetria crear dibuixos dinàmics que s'anessin movent cíclicament.

Modificadors de color Ara per ara ja tenim una instrucció de canvi de color, però e que fa és només assignar un color concret. El que es podria fer és un canvi sobre un anell continu de tonalitats de color, de manera que amb les iteracions poc a poc anessin canviant el color.

6.3.5.2. millores gràfiques

Per fer més atractiva la interfície i per ajudar a eliminar la confusió proposem unes millores:

- Indicador de primera instrucció.
- Enllaços proveïts de fletxes direccionals.
- Icona del Turtan vectoritzada.
- Eliminar l'aliasing.

6.3.5.3. Noves funcionalitats

Salvar/recuperar Es podria fer amb un tangible per guardar i un altre per recuperar. Al recuperar podrien aparèixer les instruccions operatives a l'espera que algú hi posi un tangible a sobre per heretar-ne les propietats.

Videojockey Podríem intentar la connexió del TurTan amb altres programes de so i música com el TPlayer o el ReacTable. Aquests generen una sèrie de senyals contínues que es podrien sincronitzar amb els oscil·ladors citats anteriorment. Així podríem transformar el TurTan en un programa de visualització musical molt personalitzable.

6.4. Conclusions

Com hem dit diverses vegades al llarg de la memòria, aquest és un projecte de caire experimental amb molt camí per recórrer per tal de fer un producte útil per a la major part dels usuaris, però tot i això es poden destacar diverses coses.

Tot i que la interacció amb interfícies del tipus TUI pretén ser entenedora, o com a mínim amb una línia d'aprenentatge bastant bona, encara estem en les beceroles: faltaria elaborar infinitats de tests amb usuaris i tot i així no seria un model perfecte. Això ho podem comparar amb l'evolució de lesWimp , quan varen sortir per primera vegada, era un model bast i amb el pas del temps s'han anat perfilant gràcies a l'aportació dels usuaris al llarg del temps[7].

Tot i així hem pogut veure que certes funcions que fins ara havien estat atribuïdes a l'ordinador es poden complir igual o millor amb les interfícies tangibles: la seva condició de taula obre un gran ventall a la possibilitats a l'hora d'obrir vies de comunicació entre usuaris, ja que la taula és un punt de reunió important, la gent s'hi aplega a menjar, l'utilitzen com a superfície auxiliar al costat del sofà.. Dit això, les superfícies

[7] Per exemple el sistema d'escriptori gnome.

tangibles creiem que tenen molt futur en l'oci, enriquiment de la comunicació i control d'electrodomèstics de la casa(domòtica).

Pel que hem pogut veure al llarg d'aquest projecte, una evolució cap a aquest tipus d'interfícies (sense pretensions de substituir l'ordinador en totes les seves facetes) No és una idea tan llunyana com sembla: tenim la tecnologia necessària per a dur a terme aquest tipus de sistemes i el seu cost no és excessivament elevat. Així que només cal seguir desenvolupant un llenguatge propi que ens permeti treure-li tot el suc.

Som optimistes: la aparició i popularització d'aquest maquinari pot afavorir l'aparició en la societat de noves alternatives a l'establishment del programari d'escriptori. Si ho fem bé, l'evolució del TDesktop pot ocupar un lloc important en el futur.

7. Agraïments

Sergi jordà: Tutor del projecte i assessor d'interacció.

Martin Kaltenbrunner: Per al seu suport entusiasta amb el TurTan i tot lo relacionat amb el ReacTIVIsion.

Maiol Pi: Per la seva important col·laboració amb el TurTan.

Marcos Alonso: Per la seva ajuda en quant a la calibració per malla.

Clara Orti: Pels seus consells i suport per al muntatge del vídeo TDesktop.

Al restaurant tasca i vins per alimentar-nos.

Bibliografia

[1] Ambigramatic: Programari de creació automàtica de ambigrames.

[2] Mono documentation library.

[3] Mono project home page.

[4] Open sound control.

[5] Opengl.

[6] Simple direct media layer.

[7] Tao framework.

[8] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Siggraph '92*, 1992.

[9] Kjell Borg. Ishell: a visual unix shell. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 201–207, New York, NY, USA, 1990. ACM Press.

[10] Scott Brave, Hiroshi Ishii, and Andrew Dahley. Tangible interfaces for remote collaboration and communication. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 169–178, New York, NY, USA, 1998. ACM Press.

[11] Philip L. Davidson and Jefferson Y. Han. Synthesis and control on large scale multi-touch sensing displays. In *NIME '06: Proceedings of the 2006 conference on New interfaces for musical expression*, pages 216–219, Paris, France, France, 2006. IRCAM — Centre Pompidou.

[12] Paul Dietz and Darren Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM Press.

[13] Scott Fertig, Eric Freeman, and David Gelernter. Lifestreams: an alternative to the desktop metaphor. In *CHI '96: Conference companion on Human factors in computing systems*, pages 410–411, New York, NY, USA, 1996. ACM Press.

[14] Leah Findlater and Joanna McGrenere. A comparison of static, adaptive, and adaptable menus. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 89–96, New York, NY, USA, 2004. ACM Press.

[15] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM Press.

[16] Tangible Media Group Hiroshi ISHII. Tangible user interfaces. In *CHI 2006 Workshop Proceedings*, page 163, Medford, MA, 02155 USA, 2006. Department of Computer Science Tufts University.

[17] Eva Hornecker. Physicality in tangible interaction: Bodies and the world. *Physucality 2006*, 2006.

[18] Poika Isokoski. Performance of menu-augmented soft keyboards. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 423–430, New York, NY, USA, 2004. ACM Press.

[19] Johann Habakuk Israel. Noncommand-based interaction in tangible virtual environments. In *Phisicality 2006*, page 47, Uk, Lancaster, 2006. British HCI group, Equator.

[20] Shahram Izadi, Harry Brignull, Tom Rodden, Yvonne Rogers, and Mia Underwood. Dynamo: a public interactive surface supporting the cooperative sharing and exchange of media. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 159–168, New York, NY, USA, 2003. ACM Press.

[21] Sergi Jordà;, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The reactable: exploring the synergy between live music performance and tabletop tangible interfaces. In *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 139–146, New York, NY, USA, 2007. ACM Press.

[22] Carles F. Julià, Daniel Gallardo Grassot, and Maiol Pi Blanqué. New metaphors in tangible desktops. In *Taller de Sistemes Interactius*, April 2006.

[23] Martin Kaltenbrunner and Ross Bencina. reactivision: a computer-vision framework for table-based tangible interaction. In *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 69–74, New York, NY, USA, 2007. ACM Press.

[24] Martin Kaltenbrunner, Ross Bencina, Till Bovermann, and Enrico Costanza. Tuio: A protocol for table-top tangible user interfaces. *GW2005*, 2005.

[25] Masatomo Kobayashi and Takeo Igarashi. Considering the direction of cursor movement for efficient traversal of cascading menus. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 91–94, New York, NY, USA, 2003. ACM Press.

[26] Gordon Paul Kurtenbach. *The design and evaluation of marking menus.* PhD thesis, Toronto, Ont., Canada, Canada, 1993.

[27] Alexandra Mazalek. *Media Tables: An extensible method for developing multi-user media interaction platforms for shared spaces.* PhD thesis, Massachusetts Institute of Technology, September 2005.

[28] Microsoft. Microsoft surface, 2007.

[29] Francesmary Modugno, Albert T. Corbett, and Brad A. Myers. Evaluating program representation in a demonstrational visual shell. In *CHI '95: Conference companion on Human factors in computing systems*, pages 234–235, New York, NY, USA, 1995. ACM Press.

[30] Jakob Nielsen. Noncommand user interfaces. *Commun. ACM*, 36(4):83–99, 1993.

[31] James Patten, Hiroshi Ishii, Jim Hines, and Gian Pangaro. Sensetable: a wireless object tracking platform for tangible user interfaces. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 253–260, New York, NY, USA, 2001. ACM Press.

[32] James Patten, Ben Recht, and Hiroshi Ishii. Audiopad: a tag-based interface for musical performance. In *NIME '02: Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–6, Singapore, Singapore, 2002. National University of Singapore.

[33] Stuart Reeves. Physicality, spatial configuration and computational objects. *Physucality 2006*, 2006.

[34] Sony. Eyetoy, 2003.

[35] Dag Svanaes and William Verplank. In search of metaphors for tangible user intefaces. In *DARE '00: Proceedings of DARE 2000 on Designing augmented reality environments*, pages 121–129, New York, NY, USA, 2000. ACM Press.

[36] Brygg Ullmer, Hiroshi Ishii, and Dylan Glas. mediablocks: physical containers, transports, and controls for online media. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 379–386, New York, NY, USA, 1998. ACM Press.

A. Uml Ampliat

En aquest annex, es detallaran els diagrames Uml que han aparegut al llarg de la secció d'implementació.

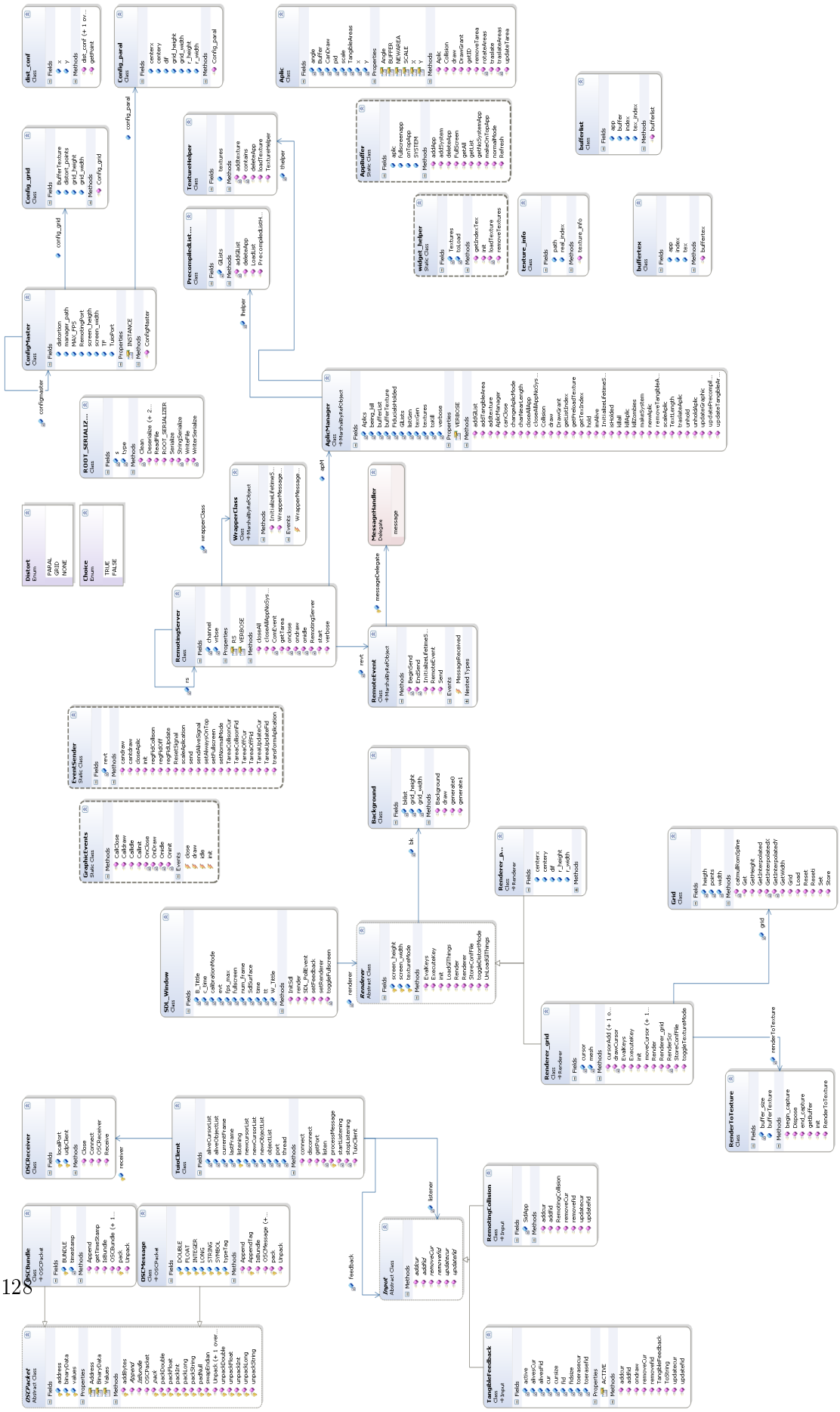


Figura A.1.: TServer versió ampliada.

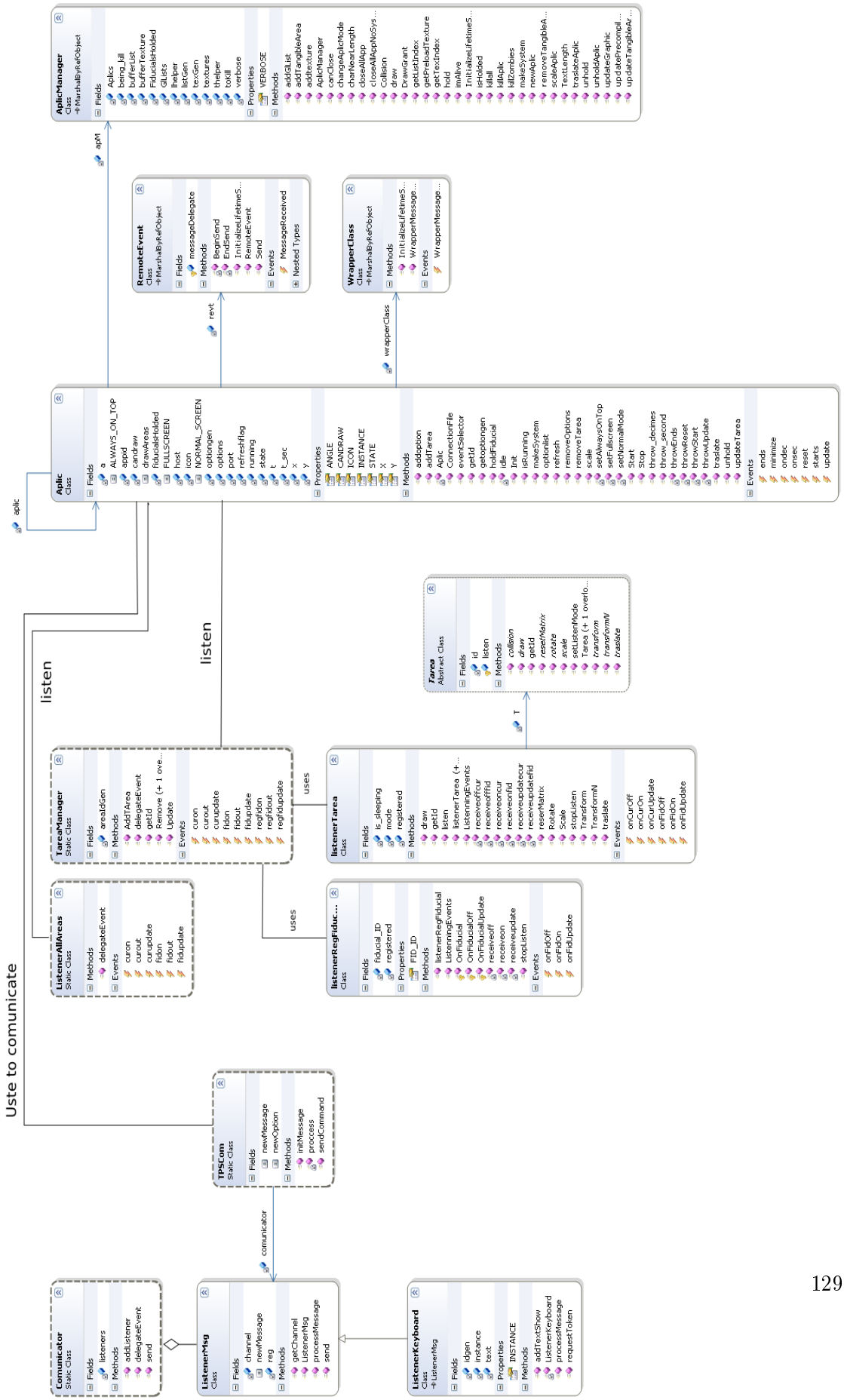


Figura A.2.: TLib versió ampliada.

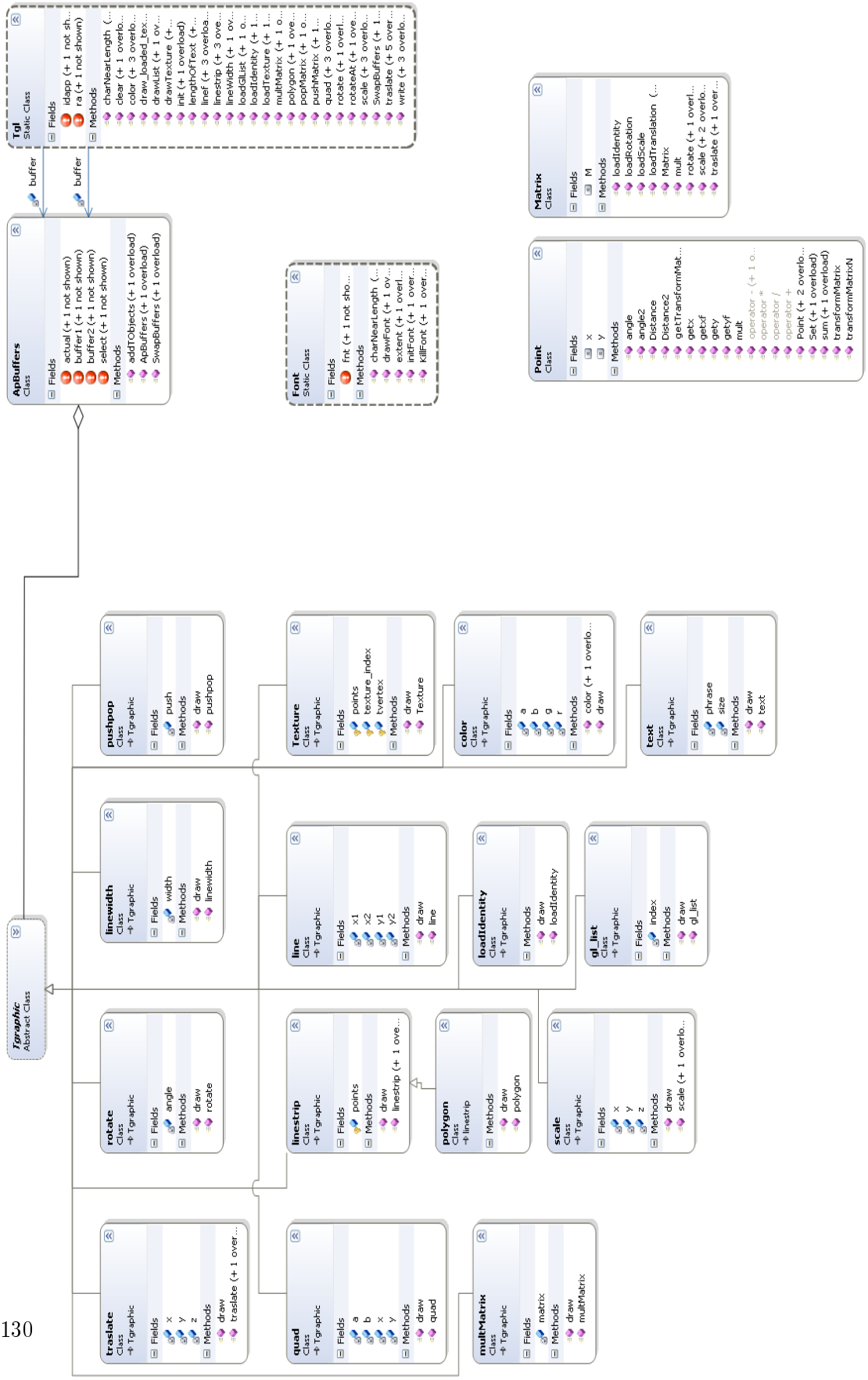


Figura A.3.: TServerObjects, part gràfica.

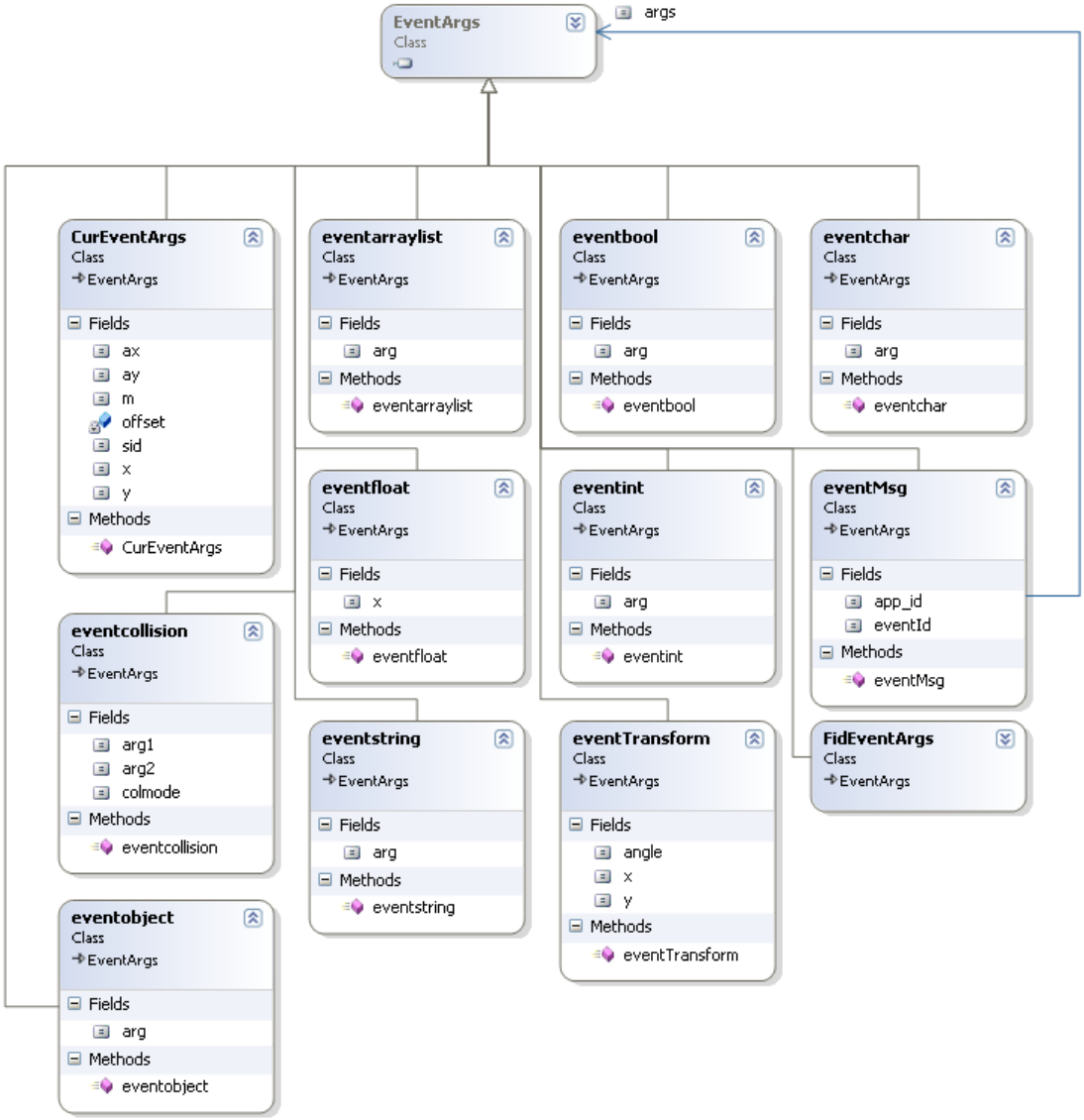


Figura A.4.: EventArgs, versió ampliada.

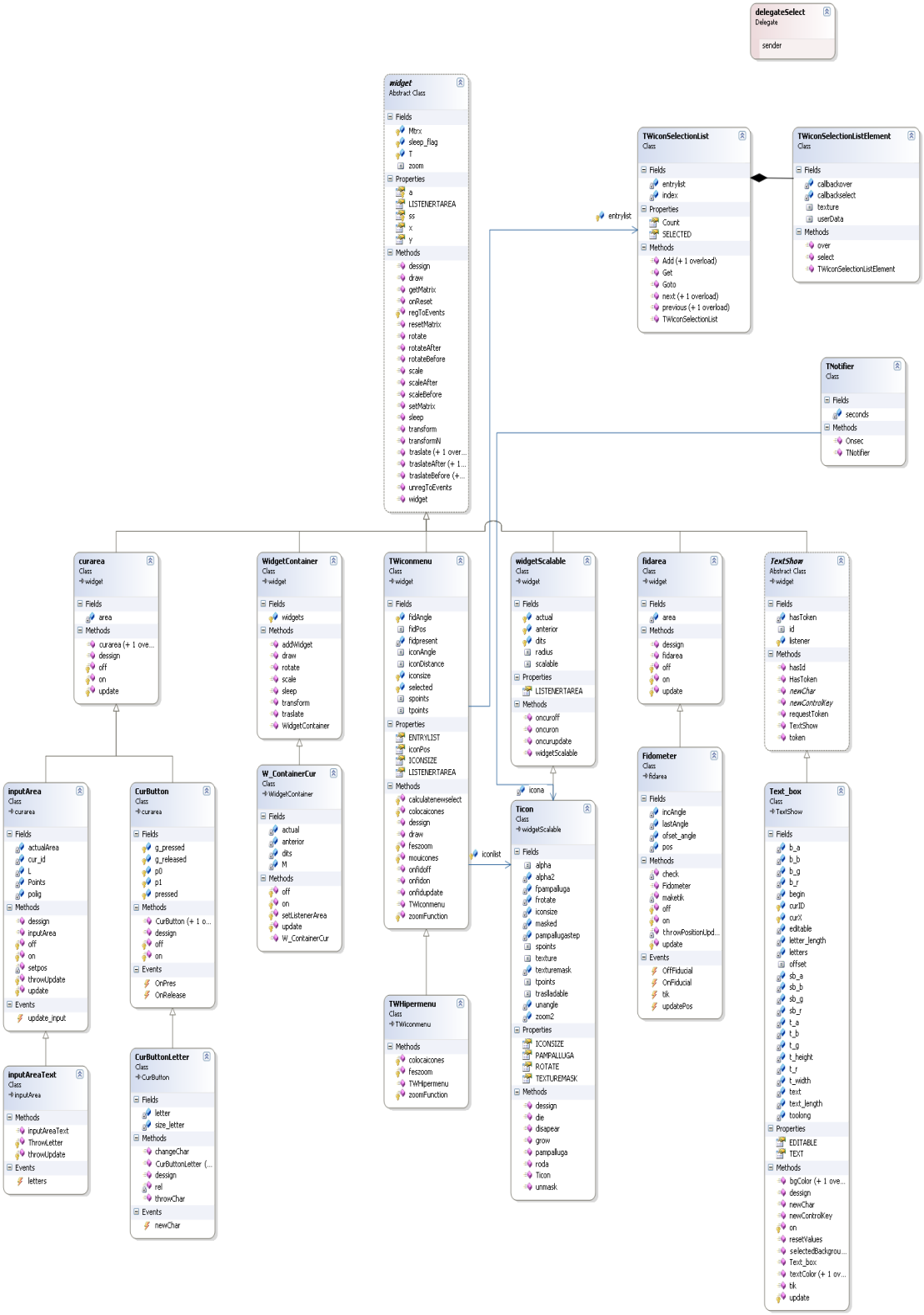


Figura A.5.: Widgets, versió ampliada.

B. TAplic Hello World

```
/*
 * Aquest és un exemple d'una aplicació que corre sobre TDesktop
 * Cal destacar la facilitat en crear el vincle amb el servidor TServer
 * i l'ús dels widgets
 */
using System;
using TLib.Device;          //llibreria bàsica per les TAplics
using TLib.Widgets;         //llibreria de widgets
using TObjects.Graphics;    //llibreria gràfica

namespace Foo {
    public class Program     {
        public static void Main()
        {
            Aplic.INSTANCE.Init();   //Registrem l'aplicació al TServer
            //Afegim el widget botó a la posició 0.5,0.5 de mida 0.8
            CurButton boto= new CurButton(0.5f,0.5f,0.08f);
            //Escoltem l'event del botó
            boto.OnRelease += new System.EventHandler(apretat);
            //iniciem el loop principal de l'aplicació
            Aplic.INSTANCE.Start();
            Aplic.INSTANCE.Stop();
        }

        //Quan arrivi l'event del botó s'executa aquest codi
        public static void apretat( System.Object o, System.EventArgs e )
        {
            //Dibuixem "hola món!" a la posició 0.6,0.5 de mida 0.03
            Tgl.pushMatrix();
              Tgl.translate(0.6f,0.5f);
              Tgl.write("hola món!", 0.03f);
            Tgl.popMatrix();
            //Actualitzem dades
            Aplic.INSTANCE.refresh();
        }
```

```
    }
}
```

C. Taula Casolana

Tot seguit varies proves fins a arribar a tenir una taula mig decent.

El punt més important, és la captura dels fiducials i cursors. Com es pot veure a la figura C.1, hem experimentat amb varies càmeres que teníem per casa i finalment la que millor resultat va donar, va ser una càmera Logitech modificada per a que capturés només l'espectre infraroig gràcies a uns retalls de pel·lícula fotogràfica col·locada davant del sensor CMOS de la webcam.

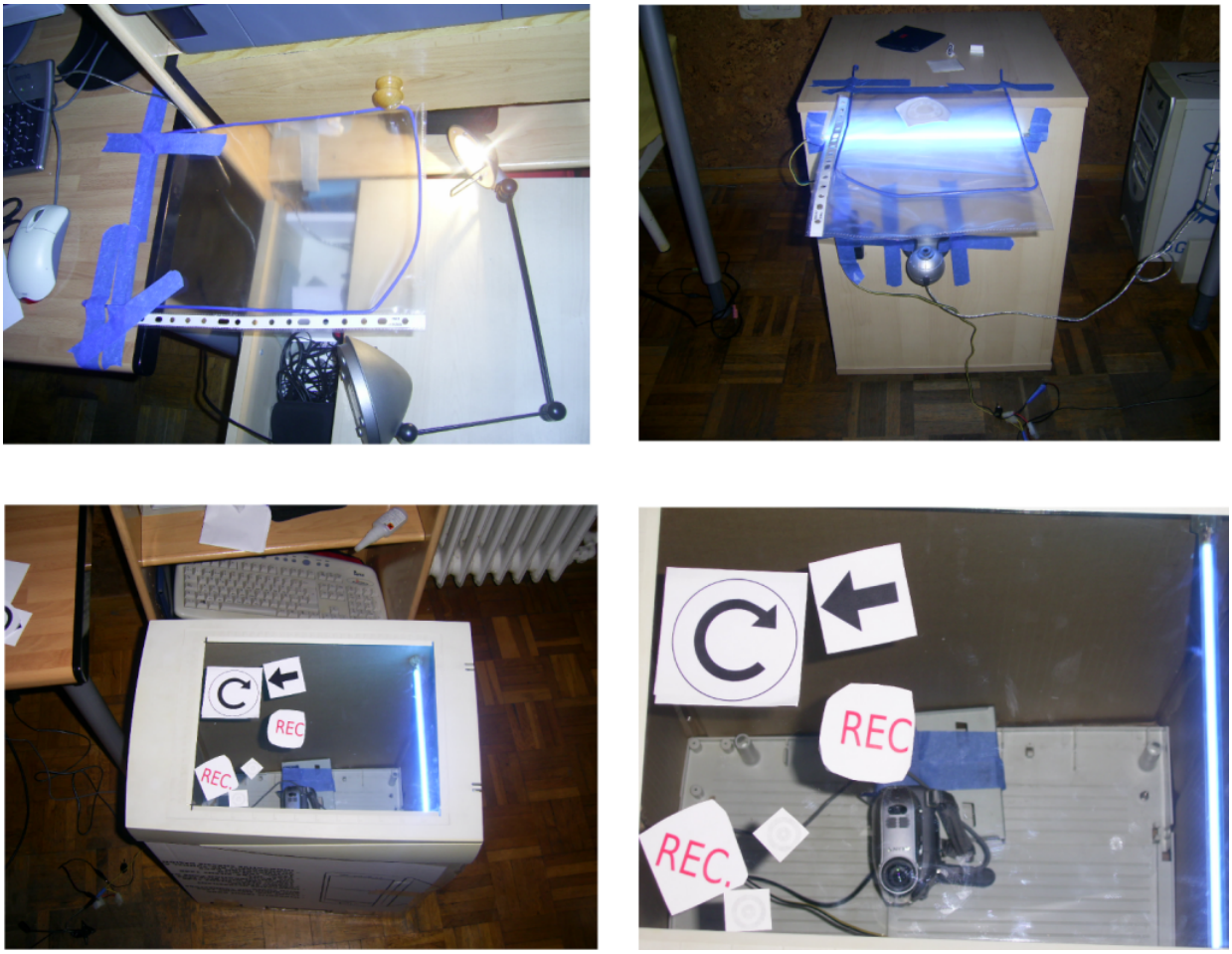


Figura C.1.: Intent A: penjador, portafolis i escànner antic.

El segon punt crucial de la taula, és la superfície on és disposen els tangibles i retorna el feedback visual. Aquesta superfície ha de ser lo suficientment translúcida com per a que no es reconeguin els fiducials com a mínim un cm per sobre d'ella i suficientment nítida per a poder rebre el feedback visual sense problemes.

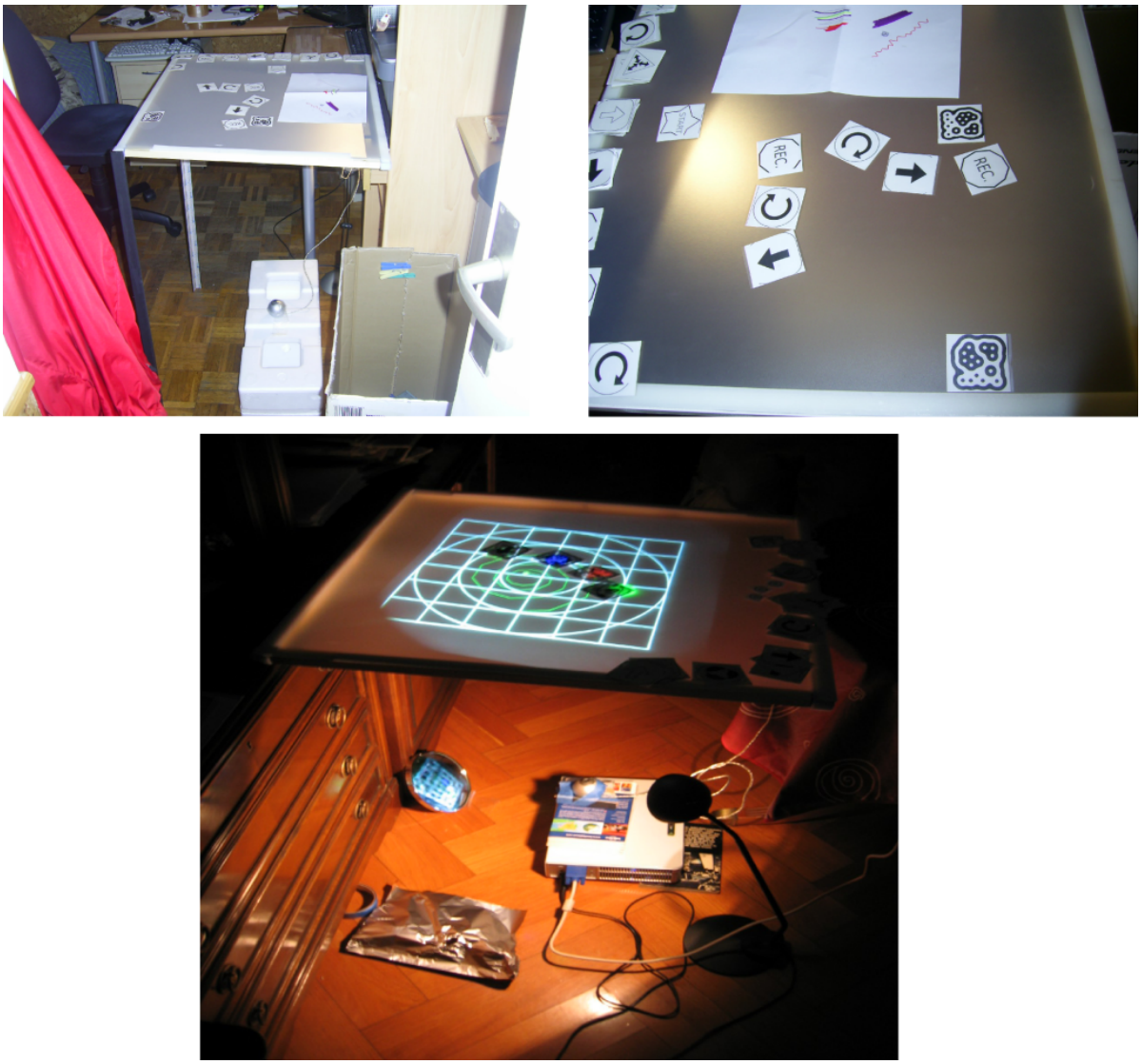


Figura C.2.: Intent B: fusta, velcro, projector.

D. Background TServer

Tot seguit un petit resum del que eren els inicis del TServer (agafat directament de la wiki del nostre projecte).

Estructura beta 1:

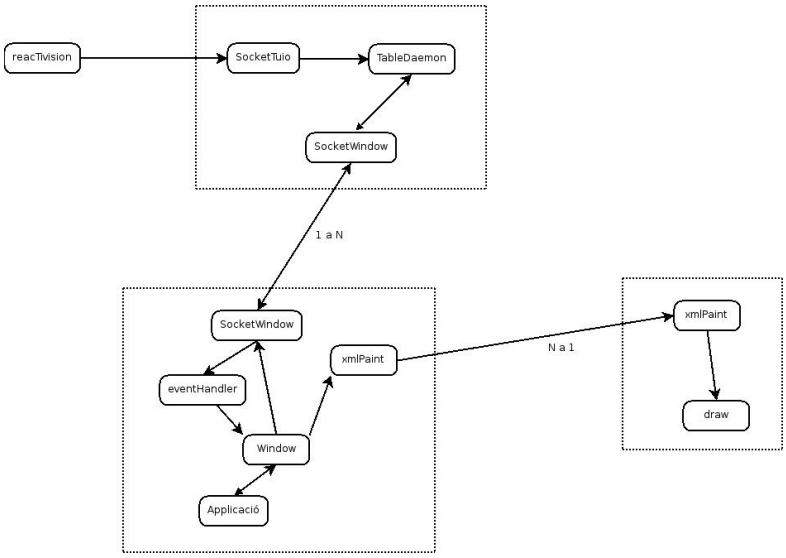


Figura D.1.: Wiki_TServer0.

```
Aquesta estructura es caracteritza per la utilització de finestres per
a fer el tractament de fiducials, cur's i widgets, la comunicació amb les
aplicacions les fa mitjançant un socket i ha de comunicar absolutament tots
els updates al "tuioDaemon". l'aplicació de pintat també es comunica
mitjançant un socket. Per tant per a cada aplicació tindrem oberts 3
sockets un d'entrada d'events un altre de sortida d'actualitzacions i
un de pintat.
Problemes:
-Necessitem comunicació ( per al tema de preferències de pintat i ordre)
entre el mòdul de pintat i el tuioDaemon, cosa que alenteix totes les
aplicacions.
-Necessita un gran volum de dades en constant refresc entre aplicacions.
```

```
-El tema del pintat queda molt limitat a primitives predefinides.
--dgg 18:19, 28 nov 2006 (CET)
```

Estructura beta 2:

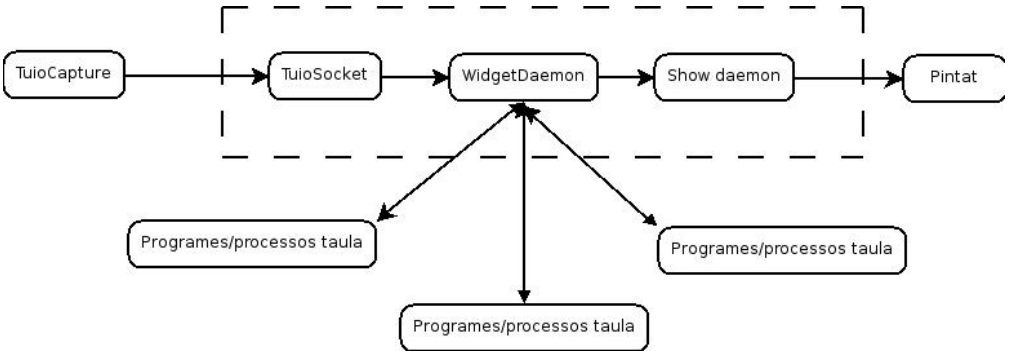


Figura D.2.: Wiki_TServer1.

```
Amb aquesta estructura, afegim tot el paradigma dels widgets, el que ens
permetrà tenir widgets d'una mateixa aplicació per qualsevol lloc de la
pantalla.
Grans diferències respecte l'anterior estructura:
-Utilització de widgets .
-Permet un ventall mes ampli de possibilitats en quant a noves formes
   d'interacció.
-Ajuntar la part de dibuixat amb la de control .
-Ens permet tenir una comunicació més fluïda i poder dibuixar figures
complexes, textures, i altres efectes.
-Widgets remots:
  Els widgets estaran continguts dins de tuioDaemon, millorarà la congestió
  entre la comunicació programa-WidgetDaemon, un widget que mostri dades no
  té perquè actualitzar-se amb el programa cada vegada que es vulgui pintar.
--dgg 18:19, 28 nov 2006 (CET)
```

Intent Fallit[1] de Dibuixar vectorialment utilitzant SVG:

```
-Perquè SVGGEN:
Degut al problema de la deslocalització dels widgets davant del seu motor
gràfic, i a les característiques de la taula, dibuixar i transmetre
primitives amb mono i mapejar-les és una tasca complicada, per això he
creat unes llibreries de primitives que he anomenat SVGGen.
```

[1] Va fracassar pel tema de la calibració i gran consum de recursos.

```
SVGgen segueix l'estàndard svg (W3C) que a grans trets es podria definir
com un llenguatge de gràfics vectorial que segueix l'estàndard xml per
a la seva codificació.
SvgGen defineix les primitives necessàries per a satisfer les necessitats
gràfiques del nostre sistema:
1.-XML s'emmagatzema en tipus text:
 -fàcil accés a primitives.
 -mínim volum .
 -ideal per a transmetre entre els missatges.
2.-Al poder dividir la imatge en gràfics:
 -ens permet transformar segons quines parts de la imatge
 -es pot construir una imatge de manera descentralitzada
 -es pot reconstruir on vulguem
Gràcies a que és un estàndard, existeixen moltes maneres de poder càrregar
aquesta imatge en diferents plataformes com a opengl, per tant, es pot
distorsionar tant com vulguem i això soluciona el problema de la calibració.
Primitives implementades:
-Cercle.
-El·lipse.
-Línia.
-MultiLínia.
-Polígon.
-Path (MultiLínia suavitzada amb corbes de Bezier)
-Rectangle.
-Text.
-Gràfic que fa la funció de contenidor de figures per aplicar
  transformacions conjuntes.
```

Tot i la gran expectativa de SvgGen i OpenGl, no va funcionar per varis motius:

1. En aquell moment nomes existien llibreries propietàries per a poder passar svg a OpenGl.
2. Les llibreries no estaven definides per a C#
3. Construït una imatge SVG consumia molta memòria.

Codi que genera la figuraD.3

```
Ellipse e = new Ellipse(200,600,60,120);
e.addStroke(3);
e.colorFill(0,0,255,6);
Polygon po = new Polygon();
```

```
po.addPoint(300,300);
po.addPoint(380,500);
po.addPoint(300,400);
po.addPoint(220,500);
po.addStroke(20);
po.colorFill(0,255,0,6);
po.colorstroke(255,0,0);
Ppath path= new Ppath();
path.addPoint(0,0);
path.addPoint(200,300);
path.addPoint(400,400);
path.colorstroke(255,255,0);
Graphic g2= new Graphic();
g2.addPrimitive(e);
g2.addPrimitive(po);
g2.addPrimitive(path);
Text t = new Text(30,10);
t.addText(300,300,"això és un gràfic");
t.colorFill(255,0,255);
Graphic g3= new Graphic();
g3.addPrimitive(t);
g3.rotate(45,300,300);
Rectangle re= new Rectangle(400,50,40,40,150,150);
re.colorFill(0,0,255,10);
re.colorstroke(255,0,0);
re.addStroke(7);
Rectangle re2= new Rectangle(400,50,40,40,150,150);
re2.colorFill(0,255,255);
re2.colorstroke(255,0,0);
re2.addStroke(7);
re2.translate(-50,-50);
viewport.addPrimitive(g2);
viewport.addPrimitive(g3);
viewport.addPrimitive(re);
viewport.addPrimitive(re2);
```

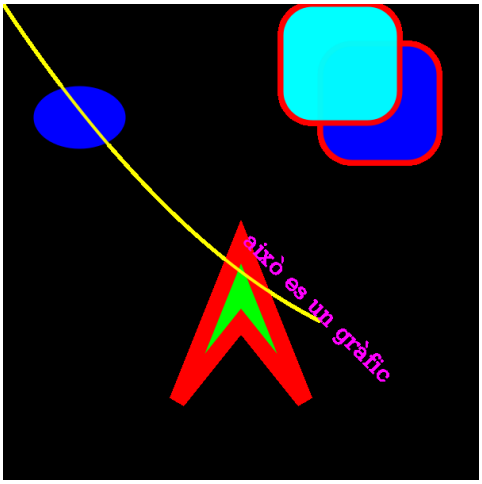


Figura D.3.: Exemple SVG.

E. Turtan, the tangible friend

TurTan és un llenguatge de programació tangible, basat en el paradigma de LOGO[1], pensat per treballar en interfícies d'usuari tangibles. Consisteix en un conjunt de fiducials o identificadors gràfics, que es poden col·locar arreu de la superfície de treball, els quals l'usuari els connecta entre si formant una seqüència d'accions. Aquesta seqüència és interpretada per una tortuga virtual, que representa les accions sobre la taula.

E.1. Objectius

La idea principal és fer una implementació tangible del llenguatge de programació Logo. A diferència del Logo original, nosaltres no volíem fer tangible la tortuga, sinó les instruccions.

Per tant varem imposar-nos aquestes premisses:

- Intuïtiu: Fàcil de fer servir sense aprenentatge previ.
- Visual: No hi ha d'haver cap text més enllà del necessari, les icones són més ràpides d'assimilar i més amigables.
- Dinàmic: Implica que hi ha d'haver un feedback instantani, ja que al ser intuïtiu i tangible, els paràmetres s'han d'ajustar i corregir dinàmicament.
- Universal: Independent de coneixements previs més enllà de bagatge cultural (com ara les icones).
- Coherent: Lògic en el seu funcionament.
- Divertit: Les tortugues són divertides si dibuixen fractals.

E.2. Dinàmica de la interacció

La interacció es fa bàsicament amb fiducials, els quals representen les diferents instruccions del TurTan i el feedback visual és instantani per a poder tenir una comunicació i control d'errors més fluïda.

Descripció passos a seguir per a crear un flux d'instruccions:

[1] Llenguatge adreçat als nens amb una sintaxis clara i fàcil d'aprendre on els objectius eren dibuixar coses mitjançant la interpretació d'una tortuga.

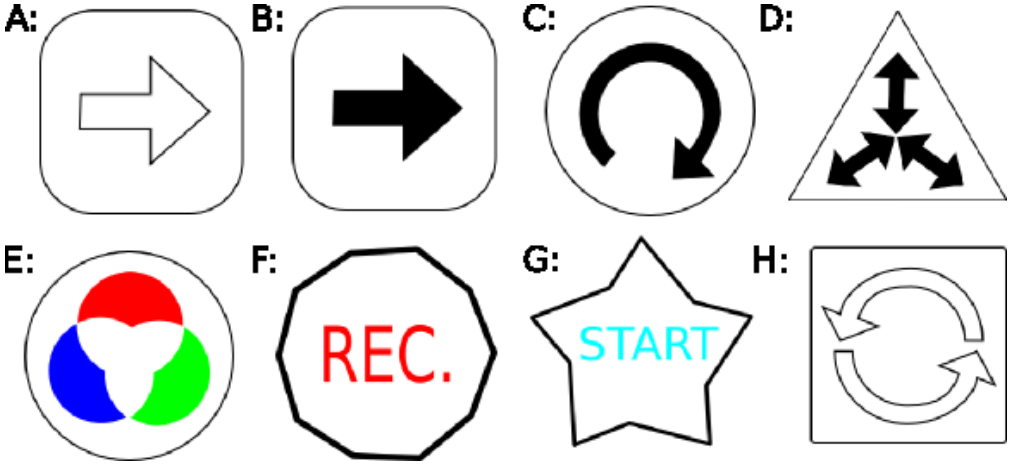


Figura E.1.: Instruccions del Turtan.

En aquesta figura es poden observar les diferents instruccions del Turtan:

Instrucció	Acció	Paràmetre a controlar
A	Avança sense pintar	Llargada del pas
B	Avança pintant	Llargada del pas
C	Gira	Angle de gir
D	Escala	Factor d'escalat
E	Canvi de color	Color
F	Grava instantània instruccions	Nombre d'iteracions
G	Torna a iniciar	—
H	Repeteix	Nombre d'iteracions

- A l'inici només apareix una tortuga al mig de la taula.
- L'usuari col·loca un objecte a la taula i immediatament aquest esdevé l'inici del programa. La tortuga farà el que indiqui la instrucció.
- Al col·locar els següents objectes a la taula aquests s'encadenaran segons un criteri de proximitat. Es poden encadenar i desencadenar a voluntat.
- Al girar qualsevol objecte es modifica la intensitat de l'acció que representa, Immediatament un retorn visual de la tortuga mostra els canvis realitzats.
- En qualsevol moment es pot traslladar i escalar l'àrea de dibuixat utilitzant els dits.

Cada fiducial a la taula representa una instrucció que és interpretada per la tortuga on seu angle representa un paràmetre de la instrucció a controlar (veure figura E.1).

E.3. Implementació

Per a simplificar, nomes comentarem la part de la implementació on es tracten les instruccions del "TurTan" i tots els gràfics relacionats, cal destacar que en uns inicis, Turtan es trobava fora del entorn TDesktop, va ser en un posterior remake que aquest llenguatge de programació es va portar a TAplic adaptant 2 widgets i convertint-los en CurMasterWidget i BlocMasterWidget.

Instruction: Base de les instruccions, conté tots els mètodes per a executar la instrucció, i per a processar l'angle de rotació de la figura associada.

InstrFactory: Crea instàncies de les instruccions excepte de InstRepeat i InstrSubrutine ja que aquestes han de ser instanciades amb llistes de referències o copia (respectivament) de les instruccions.

Bloc: Aquesta classe és la representació gràfica de les instruccions, cada bloc és la unitat d'una llista doblement enllaçada, per tant, donat un bloc tenim referències al seu anterior i al seu següent.

BlocMaster: Escolta els events dels fiducials que li arriben des del "Device", és encarregat de gestionar tots els moviments dins de la llista doblement enllaçada i conté una referència al primer element de la cadena d' instruccions.

Interpreter: És l'encarregat d'executar la cadena d'instruccions a cada event de pintat que li arriba, conté els mètodes de dibuixat i per a transformar la imatge resultant (transformacions amb els dits).

CurMaster: És qui escolta els events dels cursors que li arriben des del "Device". Conté una matriu de transformació per a modificar la imatge resultant de l'execució de les instruccions; i els mètodes per a generar-la depenent de la posició dels dits. (Només accepta 2 dits com a Input, un tercer dit serà ignorat).

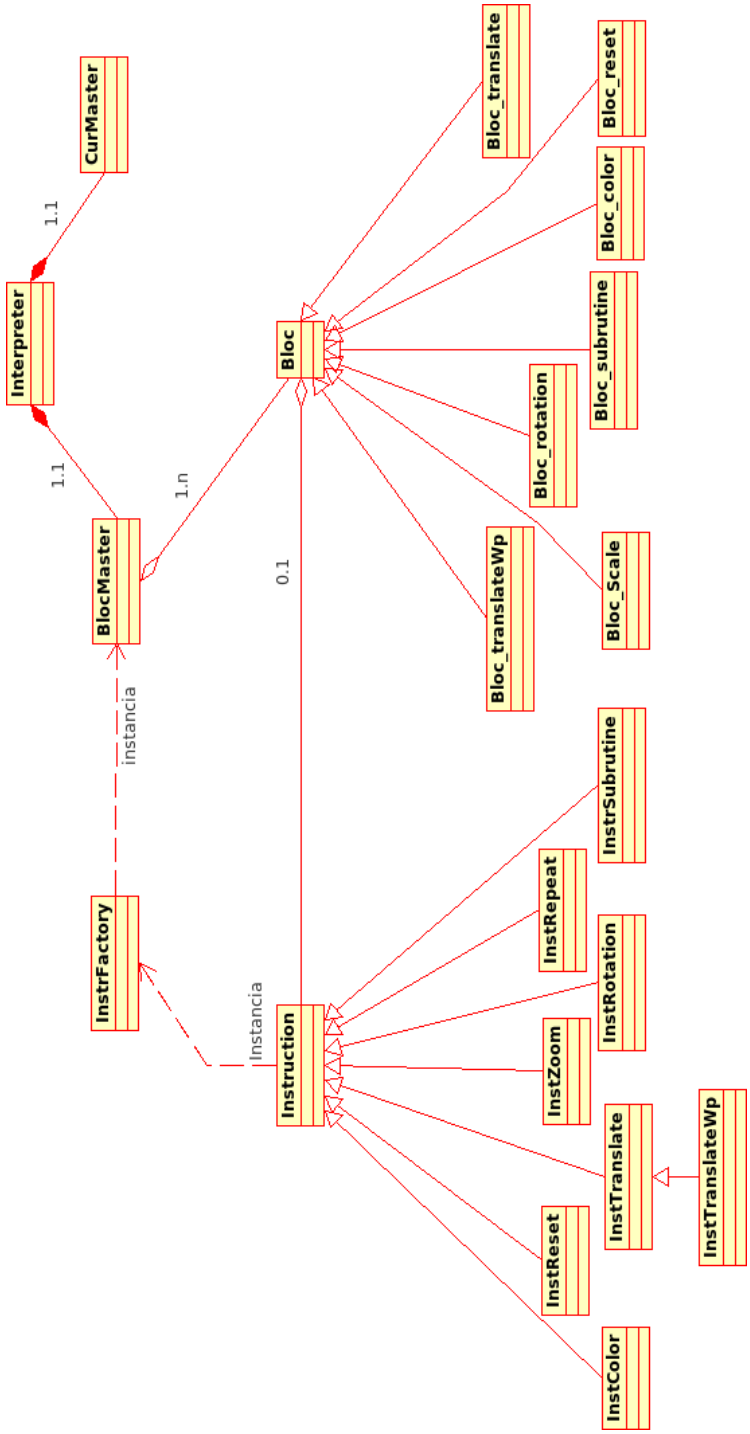


Figura E.2.: Esquema Turtan.

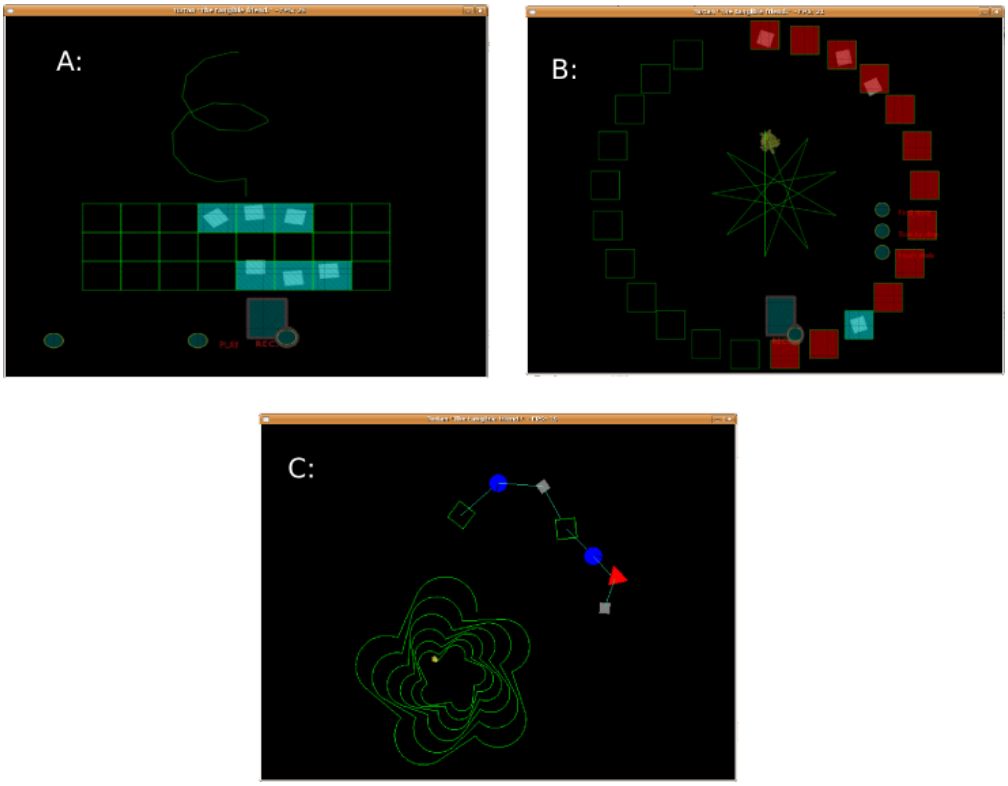


Figura E.3.: Evolució del Turtan.

A Les instruccions es disposaven sobre una graella, seguien un ordre preestablert i estaven limitades al numero de cel·les.

B Mes cel·les però mateixos problemes de llibertat.

C Instruccions lliures enllaçables: Tantes instruccions com es volguessin i llibertat total per afegir / treure'n.

F. Reconeixement de Text

Per Daniel Gallardo.

Nota: El reconeixement de text en aquests tipus D'Interfícies, és bastant complicat d'experimentar, per tant en aquest apèndix, només comentarem els dos mètodes que utilitzem: Teclat i reconeixedor de gestos. No pretenem dir que aquests dos mètodes són els mes eficients, els hem utilitzat perquè necessitàvem introduir text i no hem parat a fer els estudis d'usabilitat pertinents ja que es podria estendre a un PFC apart dedicat només a aquesta tasca.

F.1. Teclat

El teclat existeix des de ja fa molt de temps com a mètode principal d'entrada de dades, ha sigut bàsicament aquest el motiu que ens ha dut a portar-lo a TDesktop. Però un cop portat a tàctil, ens hem adonat del potencial que podria suposar un teclat en una interfície d'aquest tipus, no solament com a eina d'introduir text[18] sinó com a mètode de selecció d'eines.

Cal remarcar que la eina del teclat, tal i com la coneixem és una eina de control remot, per tant *no es podria qualificar a l'aplicació que la utilitzi com a aplicació tangible.* Així que s'ha d'anar amb molt de compte a l'hora d'utilitzar un teclat virtual i no caure en l'error de declarar-lo abans d'hora com a eina indispensable per a segons quines aplicacions

Perquè pot tenir potencial una eina d'aquest tipus? Doncs perquè pot fàcilment variar el contingut de les seves entrades, d'aquesta forma es poden dissenyar mascares per al teclat orientades a qualsevol aplicació. Actualment, existeixen teclats comercials capaços de variar la informació de les tecles dinàmicament, però aquests són poc usuals i massa cars (veure Figura F.2). Fins ara, a TDesktop, el teclat que tenim està només orientat a entrada de text, amb la peculiaritat de que si premem el bloquejador de les majúscules, les tecles del teclat canvien la seva representació.

Durant experimentació amb el teclat virtual, hem arribat a extreure certes peculiaritats:

- Al no disposar de feedback tàctil, n'hem hagut de posar un de visual, per tant al polsar una tecla ha de canviar de color. És un problema a resoldre, en principi, l'usuari expert no mira el teclat quan el fa servir.

- Al ser multi-tàctil, com identifiquem que un usuari prem dues tecles voluntàriament o en prem dues amb el mateix dit accidentalment? això ens ha dut a dissenyar-lo "uni-tàctil".
- Teclat multi-usuari?

F.2. Reconeixedor Gestual

El reconeixedor gestual, és utilitzat actualment en la majoria de dispositius amb pantalles tàctils com a principal mètode d'entrada de text. En aquesta secció no entrarem en detall de quin és el millor mètode per entrar text[26] ni compararem els estudis d'usabilitat, només explicarem com hem dut a terme el que tenim integrat a TDesktop.

Com passa amb el Teclat, també és un mètode de control remot, mes amagat, però ho és ja que no s'interactua directament amb l'aplicació.

F.2.0.4. Detecció

La detecció que hem utilitzat es penja del widget inputArea, el seu funcionament es basa en zones de control. Exactament, hem dividit l'àrea d'entrada en 4 zones on la seva funció és detectar quantes vagades hi passa un dit per damunt per a posteriorment crear una cadena del recorregut.

A	B
C	D

Taula F.1.: Taula de zones de control.

Exemple de cadena de recorregut:

- ABACD
- ACDB
- AB
- ...

Un cop es té aquesta cadena, és fàcil fer-la coincidir amb una base de dades de cadenes per a determinar quin recorregut se segueix per a crear una lletra específica (veure Figura F.3).

Per exemple la 'c ' : BACD

Figura F.1.: ArtWork Teclats.
Diferents Mascares de teclat previstes.

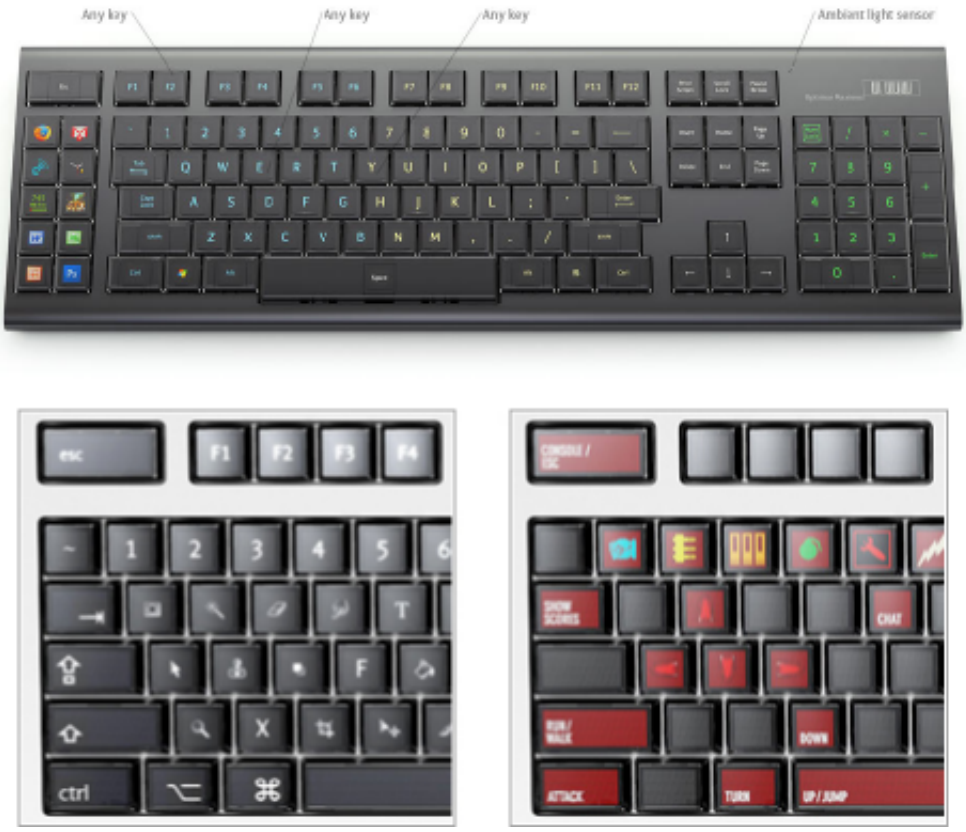


Figura F.2.: State of Art, teclat Optimus.

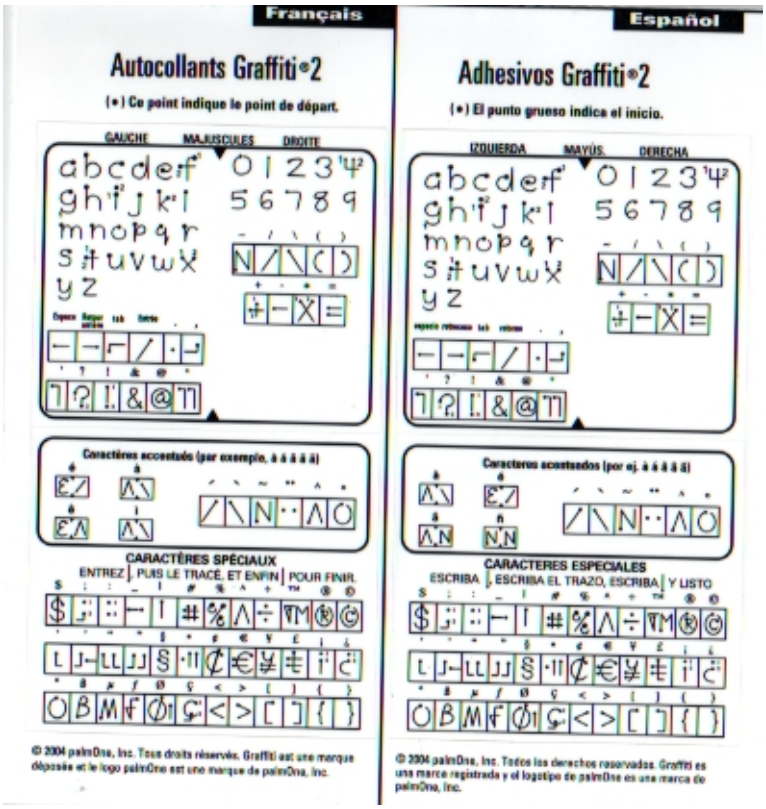


Figura F.3.: Graffiti, reconeixedor caràcters de Palm.

G. Escalar amb els dits

Un dels problemes al principi del projecte, però sorgit d'una aplicació tipus Google maps[1] per a la Taula. Per al tipus de navegació, inspirats en els exemples de la Taula interactiva multi-tàctil de Jeff Han[15] vam intentar implementar l'escalat a partir de dits.

És fàcil trobar una forma efectiva d'escalar i rotar el mapa a través dels dits , una forma és, un cop hi ha els dos dits posats, escalar el mapa proporcionalment al augment de la distància dels dits i rotar el mapa calculant la diferència d'angle entre aquests.

Però un cop implementat aquest mètode, un se n'adona que no és això el que realment està buscant. El que l'usuari espera de l'escalat bàsicament és que els *dits sempre estiguin en les mateixes posicions del mapa* (4.1.2.2 a la pàgina 69). L'escalat i la rotació en són només conseqüències.

Per a escalar el moviment amb els dos dits, vam decidir dividir-ho en dos segments: primer amb un dit traslladar i després amb l'altre dit, rotar respecte al primer, també traslladat, i escalar (s'entén molt millor a la Figura G.1). D'aquesta manera podíem rotar i escalar respecte al dit quiet, així els dits sempre seguien en el mateix lloc relatiu.

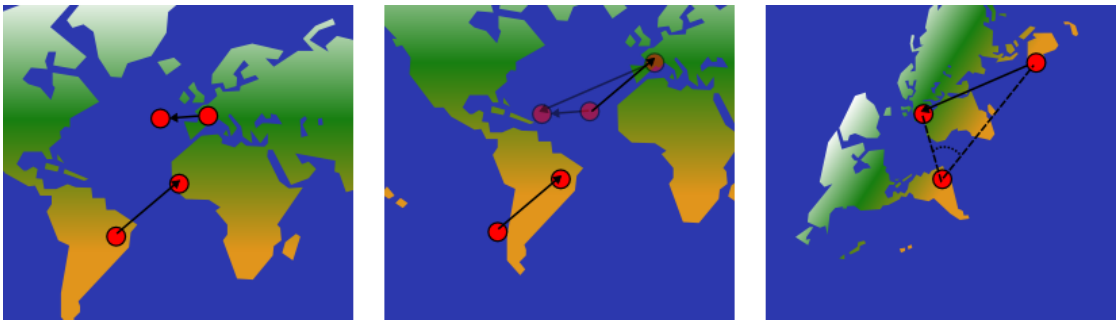


Figura G.1.: L'escalat en dos passos: un per cada dit.
En l'exemple primer calculem la translació del dit de l'esquerra respecte a ell mateix. Després calculem el nou angle i el rotem i escalem respecte del dit esquerre nou. D'aquesta manera els dits sempre es mantenen al mateix lloc del mapa.

Així reduiríem un moviment complex en dos moviments senzills que sabíem fer. Assumirem que les transformacions les fem d'aquesta forma:

$$\overrightarrow{q} = A\overrightarrow{p}$$

[1] Google Maps per qui no ho conegui és una aplicació web de navegació per un mapa realista fet a partir de imatges preses per satèl·lit. http://maps.google.com

on $\overrightarrow{p}$ és cada punt de la imatge original i $\overrightarrow{q}$ cada punt de la imatge destí. Com que estem treballant amb OpenGl, només caldrà aplicar la transformació a cada un dels punts del quadrat contenint la imatge.

La transformació es composa d'una translació, d'un gir respecte un punt i d'un escalat:

$$\overrightarrow{q} = TPGZP^{-1}\overrightarrow{p} = X\overrightarrow{p}$$

T és la translació del primer dit Δd_1, P és la translació del primer dit $d_1$al segon dit d_2, G és el gir segons l'angle calculat, Z és l'escalat segons l'angle calculat. Anomenem X al conjunt de aquestes operacions.

La matriu resultant la guardem per multiplicar-la a la següent iteració. Però la seva aplicació no és fàcil. Suposem que X_1 és la matriu de la primera iteració i X_2la de la segona. Si les apliquéssim en ordre les coordenades quedarien tergiversades per les operacions anteriors. Així si fem la multiplicació al revés es forma una pila on cada transformació $X_n$només afecta les transformacions anteriors:

$$M_n = X_1X_2 \ldots X_{n-1}X_n \Longrightarrow M_{n+1} = M_nX_{n+1}$$

Per això és important recordar M_n per cada iteració, ja que les transformacions X no són acumulatives.

Al final de cada iteració n per pintar només cal transformar tots els punts:

$$\overrightarrow{q_n} = M_n\overrightarrow{p}$$