Projecte Final de Carrera
Enginyeria Informàtica

TangiWheel: Disseny i Implementació d'un Control d'Exploració de Col·leccions Sobre Superfícies Interactives

Fernando García Sanjuan

Dirigit per:

Dr. Francisco Javier Jaén Martínez
Alejandro Catalá Bolós

Juny 2012

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Resum

Aquest treball presenta el disseny i la implementació de TangiWheel, un control per a l'exploració de col·leccions en forma de menú de pastís circular. El seu disseny està orientat a les superfícies interactives que permeten l'ús tant de dits com d'elements tangibles per realitzar les interaccions. A més, TangiWheel incorpora un estil d'interacció nou en aquest tipus de control que combina l'ús dels dits i dels tangibles. Per demostrar l'adequació del disseny i la comoditat de les interaccions possibles en aquest menú, s'han realitzat una sèrie d'experiments amb usuaris on s'ha arribat a la conclusió que la modalitat d'interacció híbrida resulta, per regla general, més avantatjosa que la tàctil o la tangible per separat.

Paraules clau: Superfícies interactives, Taules tàctils, Interfície Tangible d'Usuari (TUI), Interfície col·laborativa, Multitàctil, Tangible, Disseny d'un control, Menú, Exploració de col·leccions, Estil d'interacció, Estudi empíric.

Agraïments

A Raquel, a ma mare i a Richard per estar sempre al meu costat, donar-me suport en tot moment, inclús en els moments més difícils, i animar-me sempre a no rendir-me i seguir endavant.

Als meus amics i companys de classe, especialment a Adrià, Joan, Patricia i Jose, perquè no tot és treballar, i pels bons moments que hem compartit al llarg de tota la carrera i del desenvolupament d'aquest projecte.

Als companys del laboratori, per alleujar els moments d'estrès i pels seus consells.

I un agraïment especial a Javier i a Alejandro, primer per confiar en mi i brindar-me l'oportunitat de treballar amb ells en aquest projecte, per fer-me créixer tant a nivell professional com personal, i també per estar sempre ahí donant-me ànims i quan he tingut algun dubte o he necessitat alguna cosa, de vegades a altes hores de la nit o en cap de setmana, i tot i estar a molts kilòmetres de distància.

Gràcies de tot cor.

Índex

Índex de Figures

Índex de Taules

Índex d'Algorismes

Capítol 1. Introducció

1 Motivació

Les superfícies interactives estan sent molt populars aquests últims anys. Mentre que les taules multitàctils permeten la col·laboració i la interacció amb objectes virtuals mitjançant contactes amb els dits sobre la superfície, les taules tangibles afegeixen la possibilitat d'emprar objectes, permetent, així, realitzar les interaccions d'una manera més intuïtiva i eficient.

Un tipus d'aplicació molt popular sobre superfícies interactives són les anomenades aplicacions constructives, que consisteixen en assemblar una sèrie de components per formar un objecte més complex (per exemple, fer servir una sèrie de rodes, motors, portes, etc. per construir un cotxe). Tenen èxit en executar-se sobre taules d'aquest tipus perquè la pròpia naturalesa d'aquestes permet una millor gestió dels elements gràfics i, a més, la col·laboració de distints usuaris per realitzar la tasca objectiu (com si d'una taula normal es tractara). Les aplicacions constructives requereixen exploracions massives dels components esmentats anteriorment i permetre l'accés als mateixos de forma eficient. A més a més, en el cas de tractar-se d'entorns col·laboratius, cal proporcionar, entre altres, simultaneïtat i equitat a tots els usuaris, compartició dels elements d'interfície i controls sense orientació fixa de forma que tothom puga tindre'n la mateixa percepció independentment de la seua localització entorn a la taula.

Amb aquest treball es pretén presentar TangiWheel, un control d'exploració de col·leccions idoni per a aplicacions constructives sobre superfícies tangibles i que acompleix les característiques esmentades al paràgraf anterior.

TangiWheel organitza una sèrie d'elements al voltant d'un centre seguint el model de menú de pastís circular. Està pensat per a la manipulació, per part de diversos usuaris simultàniament, de col·leccions que poden contindre text, imatges, o ambdós al mateix temps. A més, suporta jerarquies de col·leccions, aconseguint una representació gràfica comú dels diferents nivells de la jerarquia, però distingint clarament les relacions pare-fill. Les interaccions en TangiWheel poden realitzar-se mitjançant els dits, l'ús d'objectes tangibles o la combinació d'ambdós.

2 Organització del Document

Aquest document està organitzat de la següent manera: En aquest capítol, a continuació s'expliquen els treballs en els quals s'ha inspirat aquest. A més, s'expliquen les característiques que s'han considerat necessàries per al disseny d'un control d'exploració de col·leccions sobre superfícies interactives i es realitza una comparació amb els altres treballs.

Al Capítol 2 es comenta la plataforma tecnològica que s'ha fet servir. També, com es gestiona l'entrada en aquesta, servint de base per a la creació de controls entre els que destaca TangiWheel. El Capítol 3 versa sobre com s'ha dissenyat el menú: quines característiques té i com funciona.

Al Capítol 4 es conta la sèrie d'experiments realitzats per demostrar les bondats del control implementat. Finalment, el Capítol 5 mostra les conclusions a les que s'ha arribat i el treball futur.

3 Treballs Relacionats

Aquest treball ha sigut influenciat per una gran varietat d'estudis sobre el disseny d'interfícies gràfiques d'usuari i de controls d'exploració d'elements o menús. En aquesta secció es mostra els estudis que han inspirat la nostra proposta i una comparació de TangiWheel amb alguns d'ells.

3.1 Interfícies Tangibles d'Usuari

Fitzmaurice et al. [12] introduïren el concepte de "Interfícies d'Usuari Aprehensibles" (de l'anglès, *Graspable User Interfaces*). Aquest tipus d'interfícies eren aquelles que permetien controlar certs components virtuals mitjançant l'ús d'objectes físics (anomenats *bricks* en anglès), fet que enriqueix la interacció de l'usuari amb els elements virtuals. Per exemple, segons els autors, permet la interacció amb les dues mans i el posicionament i establiment de l'orientació dels elements de forma paral·lela. La nostra proposta fa servir també una mena de *bricks* per controlar i manipular elements virtuals, als quals hem anomenat poms pel semblant que tenen als seus equivalents en les portes.

Un poc més tard, Ishii i Ullmer [20] amplien el concepte d'interfícies aprehensibles i les reanomenen "Interfícies Tangibles d'Usuari" (de l'anglès, *Tangible User Interfaces* o TUI). També hi plantegen l'associació d'elements físics a elements virtuals per permetre la manipulació d'aquests últims de forma més natural per a l'usuari, però no es limiten a un sol tipus d'element manipulador com era el cas dels *bricks* [12] o dels nostres poms, sinó que, amb el concepte de "bits tangibles" (*Tangible Bits*) plantegen l'associació dels elements virtuals a qualsevol element físic de la vida quotidiana. Concretament, els autors presenten uns equivalents físics als controls típics de les GUI. Substitueixen les clàssi-

ques finestres, icones, menús, manipuladors i *widgets* per lents, *ficones* (de l'anglès, *phicons* o *physical icons*), safates, *fmanipuladors* (*phandles* o *physical handles*) i instruments, respectivament (vore Figura 1). El concepte de ficones [20, 36] serveix també com a inspiració per als nostres poms. A més a més, estenen l'entorn de treball [20], fent que les TUIs puguen representar-se des de sobre superfícies interactives com ara taules (metaDESK) o pissarres (transBOARD) fins a habitacions senceres (ambientROOM). La nostra proposta es concentra sobre les taules interactives, i només contempla un sol tipus d'element físic. L'avantatge d'emprar els poms no només és fer més còmoda la manipulació dels elements virtuals en alguns casos, sinó el fet de poder associar-los-hi, permetent així el transport dels elements gràfics per a no sobrecarregar l'àrea de treball (que, en el cas de les taules sol ser prou limitada). També, el fet de que els poms siguen de dimensions reduïdes (vore Figura 2) contribueix a aquest objectiu, i el no tindre molts elements físics distints, sinó que tot es realitza amb el mateix, en facilita la comprensió per a l'usuari. Actualment, el terme "interfícies tangibles d'usuari" ha esdevingut més popular i ha reemplaçat al d'"interfícies d'usuari aprehensibles".

En la línea de les ficones, altre treball que ha influït en la nostra proposta és el de mediaBlocks [37], xicotets blocs de fusta que contenen una etiqueta electrònica amb una mena de URL, que permet al tangible contindre elements físics, transportar-los i també manipular-los.

Weiss et al. [38] presenten un conjunt de controls físics fets de silicona per a la gestió d'elements virtuals anomenats *SLAP Widgets*. Al seu treball demostren que aquests són fàcils d'usar i són més precisos que els controls virtuals, així com presenten un temps d'interacció menor. Això suporta els estudis anteriorment esmentats en el fet de què les interaccions amb elements tangibles són més naturals per a les persones. Entre altres, tenen un dispositiu al qual anomenen també pom (*knob*), i exploren el seu ús per a la manipulació d'un menú de pastís circular. Els autors conclouen que les manipulacions fetes sobre el menú emprant el pom són més ràpides que les purament virtuals.

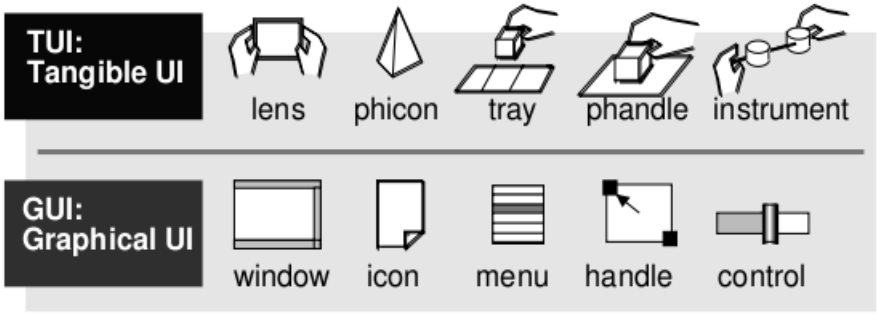


Figura 1. Elements clàssics de les GUI (baix) i els seus bits tangibles equivalents per a les TUI (dalt) proposats en [36].

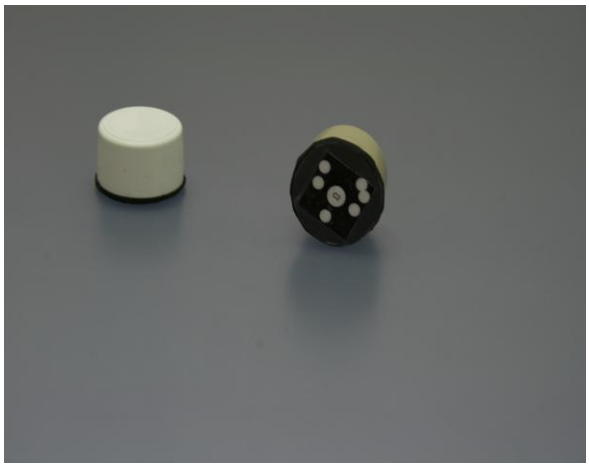


Figura 2. Poms sobre la superfície interactiva.

3.2 Disseny de Menús

L'aparició de nous tipus d'interfícies d'usuari com ara les tangibles, associades a noves plataformes tecnològiques com les superfícies interactives han provocat la necessitat de redefinir els controls d'usuari. En concret, en aquest treball es presenta TangiWheel, un menú de pastís circular que, com es podrà observar més endavant, presenta unes propietats idònies per a treballar sobre taules interactives amb més d'un usuari a la vegada.

Shen et al. [32] presenten CoR²Ds (vore Figura 3), una mena de menú on els seus ítems (propietats, accions, etc.) estan associats a un element gràfic i poden ser moguts o rotats a voluntat. D'aquesta manera pretenen donar solució als cinc problemes que identifiquen com a claus en les interfícies basades en taules interactives: oclusió, abast, associació contextual, llegibilitat i interacció concurrent. No obstant, tots els elements del "menú" estan dispersos per la superfície, provocant que, si hi ha moltes col·leccions sobre la superfície, resulte complicat saber quin ítem està associat a quin element gràfic, perjudicant la seua manipulació.

En [31] es presenten dos menús distints, cadascun sensible a un tipus diferent d'interacció, però ambdós responen només a l'ús d'elements tangibles. D'una banda, estan els "menús jeràrquics" (Figura 4-esquerra), menús de pastís on la manipulació es realitza amb les dues mans: una subjecta un tangible associat a l'element el qual es vol modificar, i amb l'altra es controla un tangible manipulador que modifica les propietats del primer o selecciona un element d'entre els possibles associat al primer tangible. El fet de tindre que realitzar totes les interaccions amb les dues mans, cadascuna prement un tangible pot resultar un poc tediós.

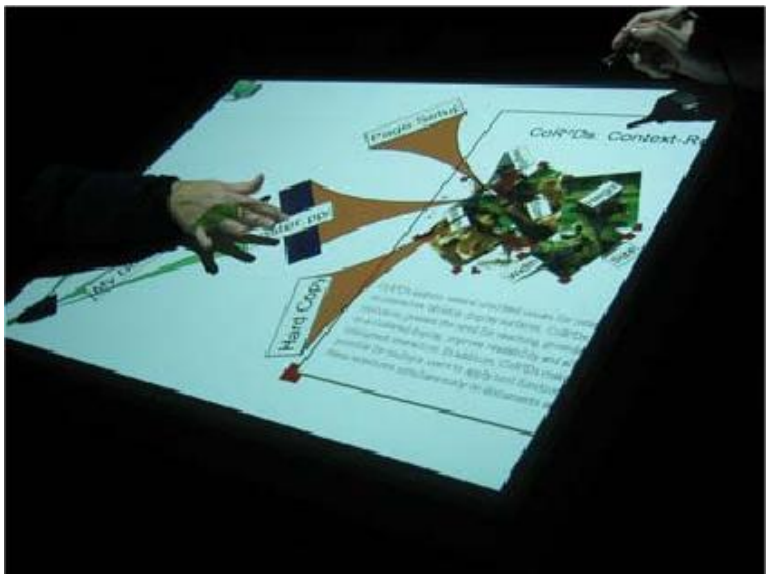


Figura 3. CoR²Ds [32].

D'altra banda, estan els "menús flotants" (Figura 4-dreta). Aquests últims no són circulars, i tant la manipulació del control (rotacions i translacions) es realitzen amb el mateix element tangible. S'ha trobat més encertat el disseny circular dels menús jeràrquics, ja que els menús de pastís han demostrat ser útils i entre un 15-20% més ràpids que els menús lineals a l'hora cercar i seleccionar elements d'una col·lecció [19, 5]. A més, considerem que aquesta forma d'interacció amb un mateix tangible pot portar a equivocacions per part de l'usuari, com ara que moga el menú quan en realitat el que volia era seleccionar un element, o viceversa.

Altre punt a destacar d'aquest treball de Patten et al. [31] és que la creació de qualsevol instància d'un menú es realitza mitjançant zones especials o *hotspots* sobre la superfície, limitant així l'accés a la instanciació dels menús.

Figura 4. Menú jeràrquic (esquerra) i menú flotant (dreta) [31].

En la línea de dissenyar menús associats a tangibles, Hancock et al. [16] en presenten un (vore Figura 5) per explorar elements que pot ser usat en dues modalitats: una tàctil, en la qual s'usen les dues mans per realitzar les interacci-

ons; i una tangible, en la qual s'associa un element físic al menú i tant el control de posició i d'orientació com l'exploració dels elements es realitzen mitjançant aquest (que es manipula amb una sola mà). En aquest segon cas, la translació i la rotació s'aconsegueixen movent i rotant l'objecte, i l'exploració, girant una *trackball* que té el dispositiu físic al damunt. En el cas tàctil, amb un dit es pot manipular el menú i, realitzant un segon toc amb un altre dit, s'efectua l'exploració d'ítems.

Aquest menú només està pensat per realitzar exploracions d'elements, així que no es considera cap altra funció ni es fa cap distinció entre elements simples o "fulla" i elements que representen subcol·leccions. Els autors realitzen un estudi en el que pretenen mesurar la percepció de facilitat d'ús en ambdues modalitats, i conclouen que l'enfocament tangible és més fàcil d'usar per a la navegació i la compartició d'informació.

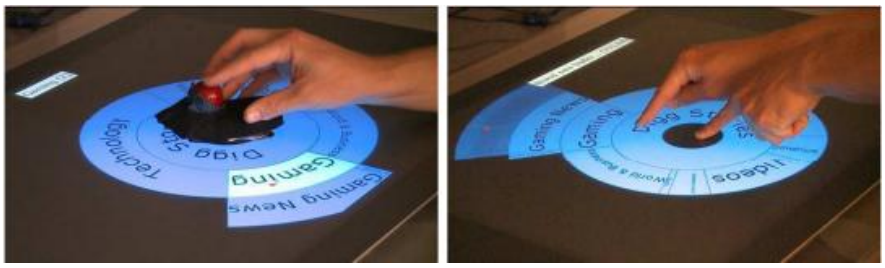


Figura 5. Menús dissenyats per Hancock et al. en [16]. Modalitat tangible (esquerra) i modalitat multitàctil (dreta).

Especialment interessant és la proposta de Gallardo i Jordà, el *Tangible Jukebox* [14]: un menú de pastís per a la navegació i gestió de llistes de música. Empra cartes físiques amb un codi imprès per emmagatzemar una col·lecció de música que, en ser dipositades sobre la superfície i reconegudes pel sistema, una sèrie d'elements s'organitzen de forma circular al seu voltant, i després la interacció pot realitzar-se tant rotant la pròpia carta com realitzant gestos amb els dits per explorar les diferents categories de música i, dins de cada categoria, les diferents cançons.

Figura 6. *Tangible Jukebox* [14].

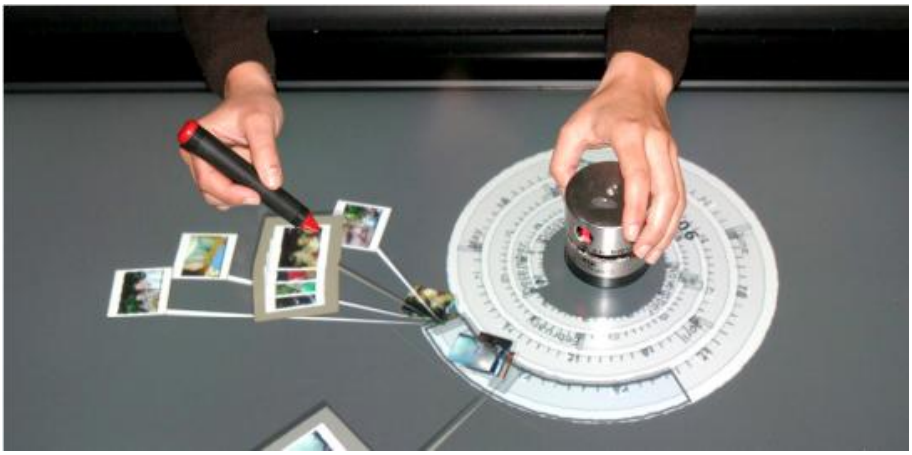


Figura 7. Photohelix [18].

Tal i com afirmen Lepinski et al. en [23], en els últims anys s'ha realitzat un gran esforç en la investigació sobre plataformes multitàctils, però no s'ha prestat molta atenció a dissenyar un conjunt estàndard d'elements d'interfície per a aquestes plataformes, com ara els menús. Sobretot, els estudis se centren en millorar la interacció en menús amb el ratolí [1, 33] o amb un bolígraf digital, on destaquen els menús de marcat (*marking menus*) [4, 40]. Aquests últims, generalment proporcionen suport per a l'exploració de col·leccions jeràrquiques i intenten donar suport a col·leccions cada vegada més grans, però no acaben de permetre un nombre virtualment il·limitat d'elements.

Tornant al treball de Lepinski et al. [23], aquests realitzen un estudi sobre un menú de marcat dissenyat específicament per a plataformes multitàctils, i els seus experiments conclouen que aquest tipus de menús permeten interaccions més ràpides que les dels menús de marcat monotàctils tradicionals a l'hora de realitzar seleccions d'elements.

Hilliges et al. dissenyen en [18] Photohelix, un menú per explorar col·leccions de fotografies que es distribueixen en forma d'espiral en torn a un element tangible que controla la seua posició i orientació (vore Figura 7). Aquesta organització permet tindre una visió completa de tots els elements, però té el problema de que a mesura que s'inclouen més fotografies el menú creix desmesuradament. La nostra proposta no presenta aquest problema, car tenim un màxim d'elements visibles alhora per a una mida invariant del control, amb la possibilitat d'explorar tota la col·lecció completa amagant elements antics i mostrant-ne de nous. Tot i això, si es desitja, TangiWheel també permet modificar aquest límit, de forma que es pot aconseguir tindre tots els elements de la col·lecció visibles al mateix temps. Per explorar subconjunts d'imatges es fa servir un bolígraf digital, però no hi ha suport per a col·leccions niades.

Un enfocament distint és el pres per Bailly et al. en els menús ona o *wave menus* [4]. Tal i com es pot vore a la Figura 8, aquests menús se centren en representar i explorar jerarquies d'elements, però els elements que es poden representar en un mateix nivell són limitats. Cada vegada que es vol explorar una

categoria d'ítems, el menú creix de mida i es crea una nova ona interior amb la col·lecció filla (o exterior en el cas de menús ona invertits o *inverted wave menus*). El creixement virtualment il·limitat del control pot suposar un problema, car pot arribar a ocupar tota la superfície disponible, impedint la instanciació de diversos controls o col·lisionant amb aquests. Els autors, però, opinen que tindre tots els elements a seleccionar en un mateix punt augmenta l'eficiència, car l'usuari no sol apartar la vista del centre del menú.

Dos problemes addicionals de tindre menús molt grans és que augmenta la taxa d'errors i disminueix la velocitat en les seleccions [4, 21, 30]. En aquests també influeix l'orientació dels elements en els menús circulars [4, 22, 40].

També resulta interessant el treball de Hesselmann et al. [17] sobre menús de mig pastís (vore Figura 9). Estan pensats per a la navegació tàctil de col·leccions jerarquitzades sobre superfícies interactives. El punt més remarcable d'aquest disseny és que no està limitat en termes d'amplada ni de profunditat de menús (i.e. nombre d'elements en un mateix nivell i nombre de nivells, respectivament), i la seua forma no es veu modificada. Les interaccions es realitzen només amb els dits, sense necessitat d'un bolígraf virtual o altre dispositiu i, a més, és capaç de solucionar el problema de l'oclusió gràcies a la seua forma de mig pastís i que està ancorat a un costat de la taula. No obstant, només permet tocs paral·lels sobre elements fulla que estan al mateix nivell de la jerarquia; si no, la interacció ha de ser seqüencial. A diferència de TangiWheel, aquest control està concebut per a ser usat per només un usuari i no permet la seua translació.

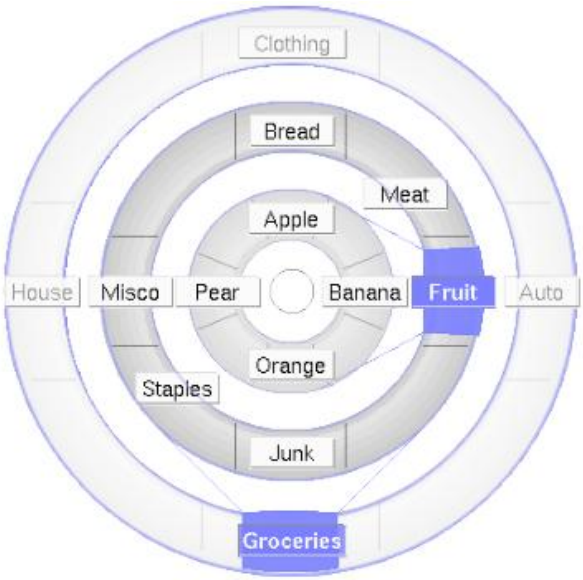


Figura 8. Instància de menú ona [4].

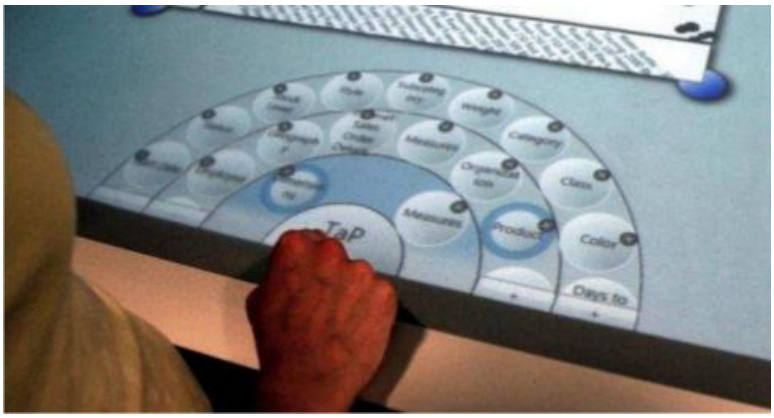


Figura 9. Menú de mig pastís [17].

3.3 Relació amb TangiWheel

Al treball de Weiss et al. [38] arriben a la conclusió de que les manipulacions fetes sobre el menú emprant un pom són més ràpides que les que usen només els dits. Al present treball també realitzem un estudi (vore Capítol 4) sobre el temps de manipulació del nostre menú de pastís, TangiWheel, i, a més, s'explora un ús híbrid del control (virtual i tangible al mateix temps), que suporta interaccions i característiques més complexes com ara la navegació en cascada de la col·lecció.

La proposta de CoR²Ds [32] presenta el problema de la dispersió dels elements. TangiWheel, per contra, manté tots els elements del menú al voltant d'un punt central com veurem al Capítol 3, augmentant així la cohesió d'aquests i permetent un major nombre de menús sobre la taula. Respecte als problemes de les TUIs identificats pels autors, TangiWheel en dóna solució a tots menys a l'oclusió dels elements amb la mà. En pròximes versions del control s'abordarà aquest problema. A continuació s'explica com es tracten aquests principis al nostre treball:

- **Abast**: Degut a les grans dimensions que poden tindre les taules interactives, pot donar-se la situació en la que un usuari necessite traslladar un element a una altra posició de la superfície. TangiWheel permet ser associat a un pom tangible, de manera que aquest pot ser alçat de la superfície i traslladat (per exemple, passant-lo de mà en mà) a una altra localització.
- **Associació contextual**: Els poms es poden vore com ferramentes genèriques, el significat específic dels quals es contextualitza sobre l'element digital al qual van associats.
- **Llegibilitat**: Com els diferents ítems del menú estan situats al voltant d'un centre mirant cap a aquest, l'orientació d'aquests no es veu afectada per la posició i orientació en la que es trobe el control, permetent una visió de 360º igual per a tots els usuaris, sense importar on estiguen situats en la taula.

- **Interacció concurrent**: Gràcies a la visió de 360º dels elements i a la capacitat del menú per ser reposicionat i rotat, es permet la interacció de diversos usuaris amb la mateixa instància de TangiWheel, permetent seleccions múltiples de diversos ítems, per exemple. També, un usuari pot ser capaç d'interactuar amb més d'un menú alhora.

Relacionant amb el treball de Patten et al. [31], en TangiWheel, com es detallarà en capítols posteriors, es pot associar un tangible al menú, i llavors a través d'aquest es realitzen les translacions o canvis d'orientació pertinents. Les seleccions poden realitzar-se amb els dits o, si es considera necessari, amb un altre pom, però no es limita la interacció a l'ús d'aquestos, de forma que l'usuari té més llibertat. Quant a la forma de crear les instàncies, les de TangiWheel poden crear-se en qualsevol punt de la superfície només situant un pom sobre la mateixa i, en cas de no disposar d'elements tangibles, també es poden usar *hotspots* específics per a aquesta tasca, de forma similar a com es fa en [31].

Igual que Hancock et al. [16], TangiWheel també funciona en dues modalitats: tàctil i tangible, però es permet combinar-les com l'usuari vulga en un mode que s'ha anomenat "híbrid". En cas d'estar manipulant el menú amb un pom, el canvi d'orientació i de posició es realitzen de forma anàloga a la proposta d'aquests autors, i la selecció es realitza tocant l'element desitjat amb un altre pom o amb el dit. D'aquesta manera, es pot elegir si emprar una mà o dues, o inclús més, en entorns multiusuari. Quan el TangiWheel està sent manipulat amb els dits només, la selecció es realitza de la mateixa forma, i la translació i la rotació es realitzen mitjançant els gestos corresponents amb un dit. De nou, no s'obliga a usar dues mans, tot i que poden emprar-s'hi si es considera oportú.

La interacció presentada en [14] és molt similar a l'adoptada en TangiWheel: en dipositar uns objectes físics amb un codi imprès a sota, es desplega el menú corresponent col·locant els elements al voltant del tangible, i la manipulació dels mateixos pot realitzar-se per mitjà de la manipulació de l'objecte físic com emprant els dits.

D'altra banda, més enllà dels esforços per part dels autors de [4] o de [40] per suportar col·leccions cada vegada més grans, TangiWheel permet gestionar col·leccions virtualment infinites (amb un nombre no acotat d'elements). Però, per evitar els problemes que poden aparéixer a l'hora de manipular menús molt grans, en TangiWheel es manté un nombre reduït d'elements visibles a la vegada (per defecte, 5), i l'orientació dels ítems cap al centre del menú fa que tots els usuaris tinguen la mateixa apreciació del control independentment de la seua localització entorn a la taula. S'aconsegueix evitar, així, l'augment de la taxa d'errors i la disminució de la velocitat en les seleccions [4, 21, 30, 22, 40].

Per suportar no només un gran nombre d'elements en amplada (i.e. en un mateix nivell de la jerarquia de col·leccions), sinó també en profunditat (i.e. un gran nombre de subcol·leccions), els menús ona [4] representen totes les sub-

col·leccions en un mateix menú, perquè els autors ho consideren avantatjós. No obstant, el creixement descontrolat del menú pot presentar problemes com ara el solapament amb altres controls. En TangiWheel, cada submenú es mostra com un control separat del pare i es permet la reubicació de cadascun i, fins i tot, ocultar-los si no són necessaris. Tot i això, també es pot aconseguir un efecte similar al dels menús d'ona si es configura el control de manera que les sub-col·leccions reemplacen els elements del menú pare en ser seleccionades. No obstant, la forma d'ocultar els fills i tornar a vore els elements del pare és menys natural i cal afegir un ítem addicional que realitze aquesta funció.

Dues característiques interessants que també presenten els menús ona [4] i que no estan implementades en TangiWheel són la previsualització dels submenús i la visualització del camí seguit en l'exploració (marcant els nodes que han sigut seleccionats i explorats). És quelcom que es tindrà en compte per a futures versions del control.

3.4 Comparació dels Estudis

Aquests treballs relacionats que s'implementen sobre superfícies interactives poden ser classificats seguint una sèrie de característiques, tal i com es mostra en la Taula 1, a fi de poder realitzar una comparació entre les diferents propostes. Aquestes característiques versen sobre aspectes com la generalitat de la solució, organització de les dades, mètodes d'entrada o d'interacció i suport multiusuari.

L'aspecte del "propòsit general" indica si la proposta es pot emprar en un àmbit genèric o, per contra, només està pensada per a un domini específic. L'"organització de la informació" versa sobre característiques que descriuen com estan suportades les dades i com s'organitzen. "Longitud" i "jerarquies niades" estan relacionades amb l'amplitud i la profunditat de les col·leccions, respectivament. La "longitud" de les col·leccions suportada es classifica en tres nivells: menuda (m), gran tot i que encara limitada (G) i virtualment il·limitada (∞) quan no hi ha cap cota superior que limite el nombre d'elements que es poden representar. El suport per a "jerarquies niades" indica si la proposta en qüestió permet tindre i representar col·leccions jeràrquiques en diferents nivells. "Jerarquies dinàmiques" es refereix a la possibilitat d'establir jerarquies de col·leccions dinàmicament, no sols de forma predeterminada o estàtica. La "compactació" senyala si la proposta conté alguna característica per limitar la seua mida (i evitar que es faça indefinidament gran a mesura que el nombre d'elements que representa creix). "Col·leccions massives" apunta si existeix la possibilitat d'estar manipulant i visualitzant diverses col·leccions d'elements al mateix temps. Per últim, "representació visual" mostra el tipus d'organització gràfica que segueixen els elements.

Quant a les característiques que tracten el tema dels "mètodes d'entrada", "modalitat" descriu els tipus d'interacció suportats. Es refereix a si l'usuari rea-

litza les accions de forma purament multitàctil ("tàctil"), només mitjançant objectes físics ("tangible") o de forma "combinada" o "híbrida". La diferència entre aquestes dues últimes modalitats és que la primera requereix una combinació ad hoc de modalitats "tàctil" i "tangible" per part de l'usuari a fi d'aconseguir els objectius desitjats, mentre que la segona suporta ambdues modalitats per a realitzar qualsevol acció, i és l'usuari qui decideix quina usar en tot moment segons li convinga. Algunes propostes proveeixen un control usant una única modalitat d'entrada o una combinació d'entrades tangible i tàctil per suportar les funcions del menú. Altra possibilitat és proveir les tècniques duals de forma separada, oferint una entrada multitàctil i altra usant tangibles. És, llavors, necessari distingir la "semblança" entre aquestes tècniques duals en la homogeneïtat del comportament observable del sistema de gestió de la col·lecció respecte a la modalitat d'entrada. Per evitar esforços cognitius innecessaris, és desitjable tindre un nivell de semblança alt. Altra característica important en el disseny de modalitats d'interacció és quan el tangible utilitzat ha sigut dissenyat expressament per al control ("tangible específic") o si, per contra, és un tangible més genèric que es pot usar en diversos contextos (com ara els poms o bé les cartes de [14]). Açò implica una acceptació i una adopció més ràpides en les plataformes existents [10].

Altre punt d'interés és el "suport multiusuari". Aquestes característiques determinen si la proposta presentada ha sigut dissenyada per a contextos multiusuari o per a tasques que només en requerisquen un. El disseny de "control 360º" expressa si la solució adoptada permet una visió comú des de qualsevol punt de la taula. En complement a altres característiques del grup "organització de la informació" que poden contribuir a un suport multiusuari (com ara permetre "col·leccions massives"), "seleccions paral·leles" indica si la proposta permet seleccionar al mateix temps diversos elements dins d'una col·lecció. "Replicació de col·leccions" es refereix a l'habilitat de clonar el contingut d'un menú per a ser explorat separadament per diversos usuaris o, simplement, crear dues vistes distintes. Finalment, "Instanciació flexible" mostra quan la solució proposada suporta un mecanisme flexible per facilitar l'accessibilitat a les col·leccions i la seua instanciació.

L'anàlisi comparativa dels estudis reportats revela que gairebé totes les propostes estan dissenyades per a dominis genèrics i usen una representació en forma de pastís o similar entorn a un centre. Quasi la meitat de les propostes proveeix suport només per a col·leccions menudes, i moltes no consideren contextos multiusuari. Com a conseqüent, no faciliten la representació massiva de col·leccions, replicació de col·leccions o seleccions paral·leles. Quant a la modalitat d'entrada, els treballs normalment es basen en una combinació de tècniques tàctil i tangible i només una ofereix dues versions del control suportant l'exploració en ambdues modalitats.

	Propòsit general	Organització de la informació						Mètodes d'entrada			Suport multiusuari			
		Longitud	Jerarquies niades	Jerarquies dinàmiques	Compactació	Col·leccions massives	Representació visual	Modalitat	Semblança	Tangible específic	Seleccions paral·leles	Replicació de col·leccions	Instanciació flexible	Control 360º
CoR²Ds [32]	✓	m	✓	✗	✓	✗	Sense sistema (entorn a un centre)	Tàctil			✓	✗	✓	✓
Audiopad [31]	✓	m	✓	✗	✓	✗	Pastís / Lineal flotant	Combinada		✗	✗	✗	✓	✓
SLAPs [38]	✓	m	✗	✗	✗	✗	Pastís	Tangible		✓	✗	✗	✓	✗
MTMM [23]	✓	m	✓	✗	✗	✗	Menús d'acords (entorn a un centre)	Tàctil			✗	✗	✗	✗
Tableball [16]	✓	G	✓	✗	✗	✗	Radial (basat en pastís)	Tàctil / Tangible	✗	✓	✗	✗	✓	✓
Photohelix [18]	✗	∞	✗	✗	✓	✗	Espiral (pastís)	Combinada		✓	✗	✗	✓	✓
Stacked [17]	✓	∞	✓	✗	✗	✗	Mig pastís	Tàctil			✗	✗	✗	✗
Jukebox [14]	✗	∞	✓	✓	✓	✓	Pastís	Combinada		✗	✓	✓	✓	✓
TangiWheel	✓	∞	✓	✓	✓	✓	Pastís	Tàctil / Tangible / Híbrida	✓	✗	✓	✓	✓	✓

Taula 1. Comparació dels treballs relacionats sobre superfícies interactives.

Basant-se en l'anàlisi prèvia, TangiWheel ha sigut conceptualment dissenyat per suplir les deficiències dels sistemes anteriors i proveir suport total per a noves característiques, incloent l'amplada i la profunditat de les col·leccions, que és desitjable tindre una longitud il·limitada i col·leccions niades. A més a més, suporta l'ús massiu de col·leccions per part de múltiples usuaris així com la manipulació de diverses col·leccions alhora per part d'una mateixa persona. La compactació ha de ser considerada juntament amb altres característiques per proveir flexibilitat i facilitat d'ús, com ara replicació de col·leccions i vista de 360º. També es proveeixen tècniques d'interacció duals, semblant-se l'una a l'altra tot el possible. Llavors, emergeix així una modalitat híbrida, en oposició a una simple combinació de les modalitats tàctil i tangible, permetent a l'usuari elegir la que més li convinga en cada moment sense cap esforç cognitiu addicional. I, per acabar, el dispositiu tangible (el pom) és estàndard i genèric en compte de ser específic per a una aplicació concreta.

Capítol 2.
Gestió de l'Entrada

1 Entorn Tecnològic: Microsoft Surface

Aquest treball ha sigut realitzat amb una Microsoft Surface[1] 1.0 [9, 34]. És una superfície interactiva retroprojectada amb una pantalla de 30" i una resolució de 1024 x 768 (*aspect ratio* de 4:3). Presentada en maig de 2007, té un processador Intel Core 2 Duo a 2.13 GHz, 2GB DDR2 de memòria RAM i una targeta gràfica ATI Radeon X1650 de 256MB, i un sistema operatiu Windows Vista de 32 bits. Amb un pes total de 90Kg, les seues dimensions es poden vore en la Figura 10.

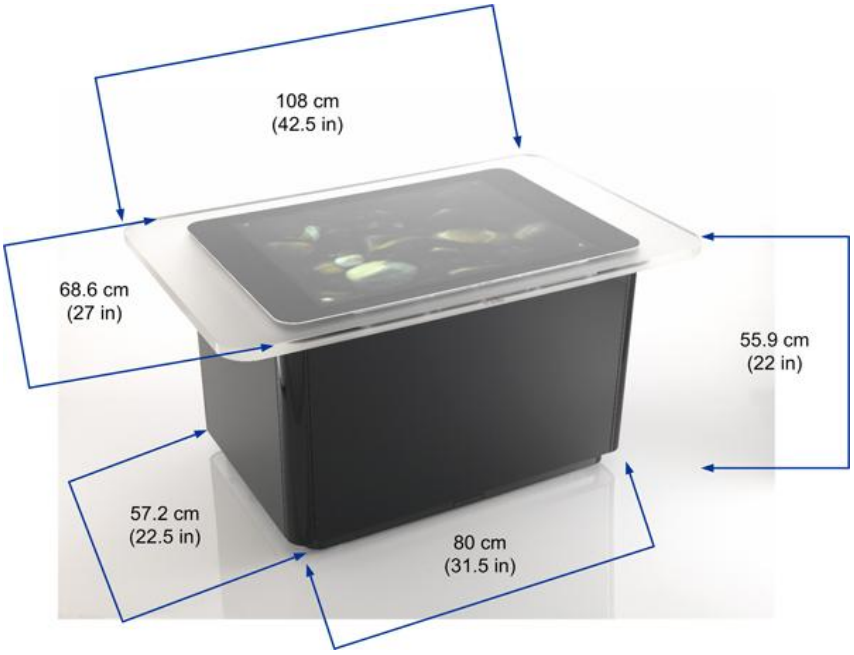


Figura 10. Dimensions de la Microsoft Surface 1.0 [34].

Juntament amb la taula interactiva, la Microsoft Surface SDK 1.0 permet obtindre informació referent a l'entrada per part dels usuaris. Com s'extrau aquesta informació serà explicat a la secció 2, i en les posteriors es comentarà com es processa aquesta.

La SDK proporciona dues formes de treballar i crear interfícies: la capa de presentació i la capa nucli. La primera està orientada a la creació d'aplicacions

[1] Des del 18 de juny de 2012, la Microsoft Surface 2.0 passa a anomenar-se Microsoft PixelSense. Com el treball d'aquest projecte és anterior a aquesta data i realitzat sobre la versió 1.0, es mantindrà el nom anterior.

tàctils amb un *framework* per al desenvolupament d'interfícies d'usuari anomenat *Windows Presentation Foundation* (WPF) [28], que ja proporciona una sèrie d'elements d'interfície i controls predefinits com ara botons, panels o llistes. En aquest treball s'ha fet servir la capa nucli que, més a baix nivell i de forma més complexa però més potent, permet desenvolupar aplicacions amb gràfics mitjançant el *framework* Microsoft XNA [29], pensat per al desenvolupament de videojocs. XNA no proporciona el concepte de control i de gestió de l'entrada de la Surface. No obstant, la SDK conté alguns exemples que s'han pres com a referència per crear els nostres propis controls, com s'explica en les seccions 3 i 4 d'aquest capítol.

2 Extracció de la Informació

Com ja s'ha comentat a la secció anterior, la MS Surface és una superfície retroprojectada. Baix la superfície, hi ha una sèrie de fonts de llum infraroja que il·luminen aquesta. Llavors, quan un dit o un objecte són dipositats sobre la mateixa, refracten més llum, la qual és percebuda per cinc càmeres infraroges (situades també a sota) que capturen la imatge a una freqüència de 60 vegades per segon, aproximadament. Els principis de funcionament dels sistemes com l'esmentat reben el nom d'il·luminació difusa (de l'anglès, *Diffused Illumination* o DI) [39]. L'estructura interna queda representada en la Figura 12. D'aquesta manera, el sistema de visió és capaç de reconèixer fins a 52 punts de contacte simultàniament i també captar objectes que estiguen propers a la superfície encara que no arriben a tocar-la. Per exemple, a la Figura 11 esquerra i centre es pot observar que el palmell de la mà no arriba a tocar la superfície però, no obstant, és captat per les càmeres.

Tal i com es mostra a la Figura 11, les càmeres recullen una imatge dels contactes (centre) i, posteriorment, aplicant un programari de reconeixement, extreuen la informació dels dits (dreta), com ara la posició, l'orientació i les dimensions de cadascun dels dits. De forma similar s'extrauria la informació per a un objecte o per a unes etiquetes especials que el sistema és capaç de reconèixer i de les que es parlarà en la secció 3.

Figura 11. Procés de captura de la informació de contactes. Contactes situats sobre la superfície (esquerra), imatge captada per les càmeres (centre) i informació dels dits extreta (dreta).

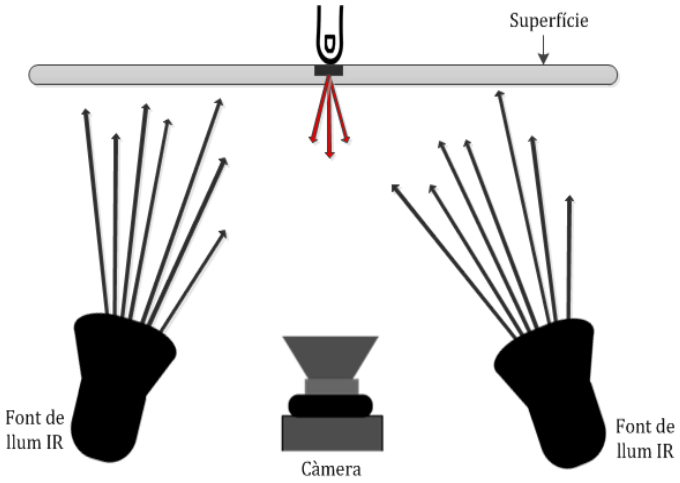


Figura 12. Esquema d'un sistema de visió d'il·luminació difusa.

3 Manejador de Contactes

La Microsoft Surface proporciona tres mecanismes d'interacció els quals s'engloben dins del concepte de contacte (o *contact*, en anglès). Aquests són: dits (*fingers*), etiquetes amb un codi imprès que és llegit pel sistema de visió de la Surface (*tags*) i qualsevol objecte que no es puga catalogar dins de les dues anteriors classificacions (*blobs*). En aquest treball només fem ús dels dos primers.

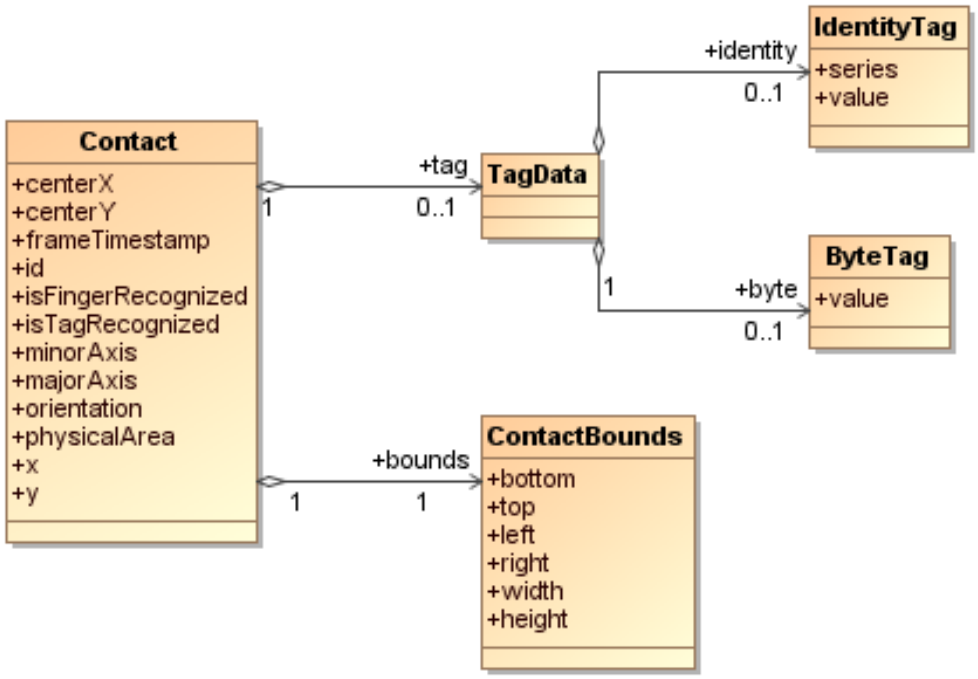


Figura 13. Jerarquia de classes dels contactes.

Cada contacte és unívocament identificat i té associada una informació com ara la posició, la rotació, etc. Un contacte no és un punt concret, sinó que es tracta d'una superfície (és a dir, una regió delimitada pels paràmetres de la propietat *bounds*). A més, si el contacte resulta ser una etiqueta, tindrà un TagData associat. Les etiquetes poden ser de dos tipus: de tipus byte o de tipus *identity*. Les primeres, com el seu nom indica, emmagatzemen 8 bits d'informació, mentre que les últimes consisteixen en dues propietats, sèrie i valor, de forma que es tenen 64 bits per a identificar les sèries i 64 bits més per donar un valor dins de cada sèrie (el que fa un total de 128 bits d'informació emmagatzemable). A la Figura 13 es pot vore un diagrama de classes que il·lustra aquesta informació.

La SDK de la Microsoft Surface proporciona una sèrie de classes que permeten treballar amb la informació extreta pel sistema de visió. No obstant, com s'explicarà més avall, treballar amb aquesta pot resultar un poc costós. El nostre objectiu en aquesta secció i la següent serà, per tant, presentar una abstracció d'aquest conjunt de classes per simplificar la gestió de l'entrada a través d'una classe nova, *Control*, que permetrà realitzar les operacions a més baix nivell de forma transparent per al desenvolupador. D'aquesta manera, sempre que es necessite crear una classe que responga a contactes, en fer que herete d'aquesta *Control* s'aconseguirà simplificar molt la gestió.

Centrant l'atenció primer en la SDK de la Surface, el processament dels contactes és dut a terme per la classe *UIController*, de la qual només hi ha una instància en tota l'aplicació seguint un patró de disseny *singleton*. Cada vegada que el sistema de visió detecta un contacte (siga del tipus que siga) en la superfície, l'emmagatzema en una cua. Després la processa i determina sobre quin objecte ha impactat. A més, gràcies a que manté una col·lecció de contactes ordenats per l'instant en el que han tingut lloc i l'objecte sobre el que han impactat, pot determinar quin esdeveniment ha ocorregut per a aquest contacte determinat. Aquest esdeveniment pertany a un tipus enumerat anomenat *ContactEventType* que pot ser dels següents tipus:

- **Enter:** El contacte ha entrat en un nou objecte de la interfície.
- **Leave:** El contacte ix de l'objecte de la interfície en el que estava.
- **Added:** El contacte entra en la unitat Surface.
- **Removed:** El contacte ix de la Surface.
- **Changed:** El contacte canvia alguna de les seues propietats, com ara la posició o l'orientació.

En la Figura 14 es pot vore un diagrama d'estats que indica la gestió que l'*UIController* fa del cicle de vida corresponent a l'entrada produïda per cada contacte depenent dels esdeveniments *ContactEventType* que hagen ocorregut. Cada contacte segueix una instància d'eixe diagrama, és a dir, cada vegada que entra un contacte nou compleix el seu propi diagrama de transició d'estats. Els possibles estats assolibles són els següents:

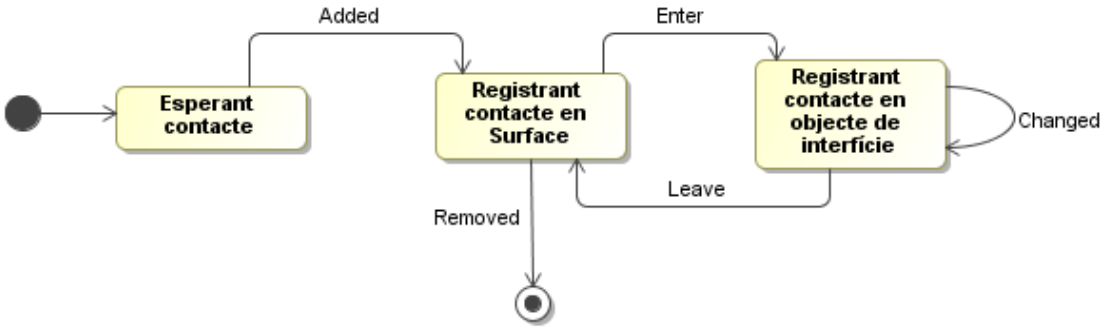


Figura 14. Gestió del cicle de vida d'un contacte determinat per part de l'*UIController* davant els possibles *ContactEventType* ocorreguts.

- **Esperant contacte**
  L'*UIController* està esperant que algun contacte es pose sobre la superfície. Quan açò ocorre, passa a l'estat "Registrant contacte en Surface".
- **Registrant contacte en Surface**
  Si ve de l'estat "Esperant contacte", significa que ha tingut lloc un nou contacte sobre la superfície. Llavors, a continuació es detecta sobre quin objecte de la interfície està i, quan ocórrega un esdeveniment Enter, passa a l'estat "Registrant contacte en objecte de interfície". Per contra, si ve de l'estat "Registrant contacte en objecte de interfície" significa que el contacte ha eixit de la regió delimitada per l'objecte sobre el que es trobava, i ara poden ocórrer dos esdeveniments: 1) Si s'ha retirat el contacte de la superfície, es produirà un esdeveniment *Removed* i s'haurà acabat el cicle de vida per a aquest contacte; 2) si s'ha retirat el contacte d'un objecte per fer-lo entrar en un altre, es produirà un esdeveniment *Enter* i tornarà a l'estat "Registrant contacte en objecte de interfície".
- **Registrant contacte en objecte de interfície**
  Entra en aquest estat quan l'*UIController* detecta que un nou contacte està impactant sobre un objecte de la interfície. Si el contacte canvia de posició o d'orientació sense eixir d'eixe objecte, es produirà un esdeveniment *Changed*, mentre que si es retira el dit (ja siga perquè s'ha retirat de la superfície o perquè se n'ha eixit dels límits de l'objecte actual per entrar-ne en un altre), l'esdeveniment que es llançarà serà *Leave* i anirem a l'estat "Registrant contacte en Surface".

Una vegada ha establert l'esdeveniment associat al contacte actual, l'*UIController* encamina eixe esdeveniment cap a l'objecte impactat. La Figura 15 mostra de forma conceptual com es realitza aquest procés: Primer, l'aplicació crea una instància d'*UIController*. En fer-ho, se li passa un delegat, anomenat *HitTestCallback*, a una funció que haurà de ser implementada per la pròpia aplicació, i l'objectiu de la qual serà identificar sobre quin objecte ha impactat un determinat contacte. Aquest objecte en qüestió ha d'implementar la interfície *IInputElementStateMachine* (també de la SDK de la Surface), que conté operacions per a la gestió dels contactes. Llavors, per a cada contacte nou que proces-

se l'*UIController*, farà una crida al manejador del delegat *HitTestCallback* de l'aplicació passant la informació del contacte (entre altres coses, la posició d'aquest). L'aplicació, amb aquestes dades i les referències a tots els objectes de la interfície, buscarà sobre quin s'ha impactat i li'l passarà al controlador. Per últim, aquest encaminarà l'esdeveniment ocorregut associat al contacte cap a l'objecte impactat, fent ús del mètode *Update* de la interfície *IInputElementStateMachine*, que rep com a paràmetre una cua de contactes amb els esdeveniments corresponents).

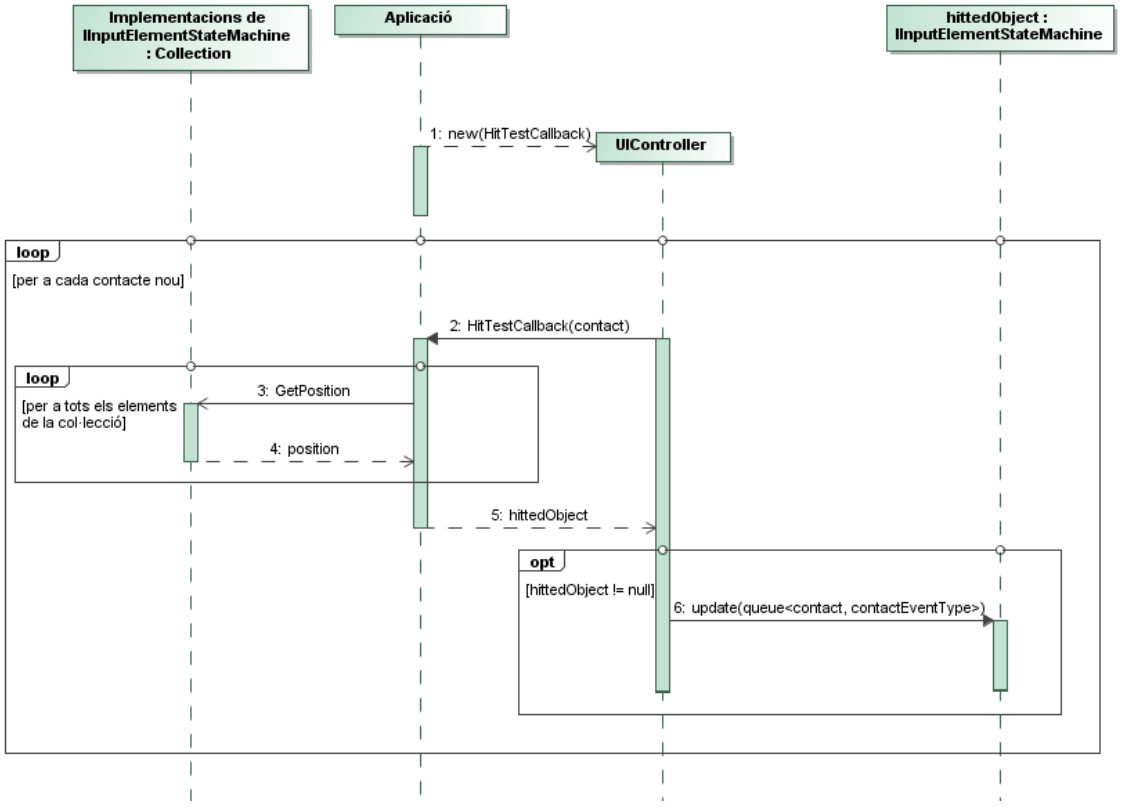


Figura 15. Encaminament dels esdeveniments de la classe *UIController* cap als diferents *IInputStateMachine*.

Per tant, per desenvolupar una aplicació amb diferents controls sobre Microsoft Surface, s'ha de tindre una classe que continga una referència a tots els controls, a fi d'implementar el mètode que és cridat pel delegat *HitTestCallback*. Cadascun d'aquests objectes, a més, ha d'implementar la interfície *IInputElementStateMachine* per a què, quan l'*UIController* detecte un contacte sobre ell, puga encaminar-lo correctament i, a més, per a què gestione els seus propis contactes una vegada ja sap quins han impactat sobre ell. La nostra proposta introdueix una capa (que s'explicarà més en detall en la secció 4) entre el controlador i els diferents controls concrets (vore Figura 16) que permet l'abstracció total d'aquests, de manera que l'*UIController* és invisible per als controls, fet que en facilita la implementació de nous. En aquesta figura, a mesura que es puja de capa s'aconsegueix un major grau d'abstracció, és a dir, que l'última capa és a més baix nivell (més propera a la màquina) que la primera (o la de més amunt),

que és la més propera a l'usuari o més abstracta. Les fletxes entre els distints elements representen dependència o comunicació direccional. Així, per exemple, els controls concrets (Control 1, Control 2 i Control 3) interactuen amb l'Aplicació i depenen o es comuniquen amb la capa intermèdia que, al seu torn, depèn dels elements de la capa més baixa. No obstant, elements d'una capa són totalment independents de capes superiors. D'aquesta manera, els canvis en una capa no afecten per a res a les capes que estan per sota d'aquesta.

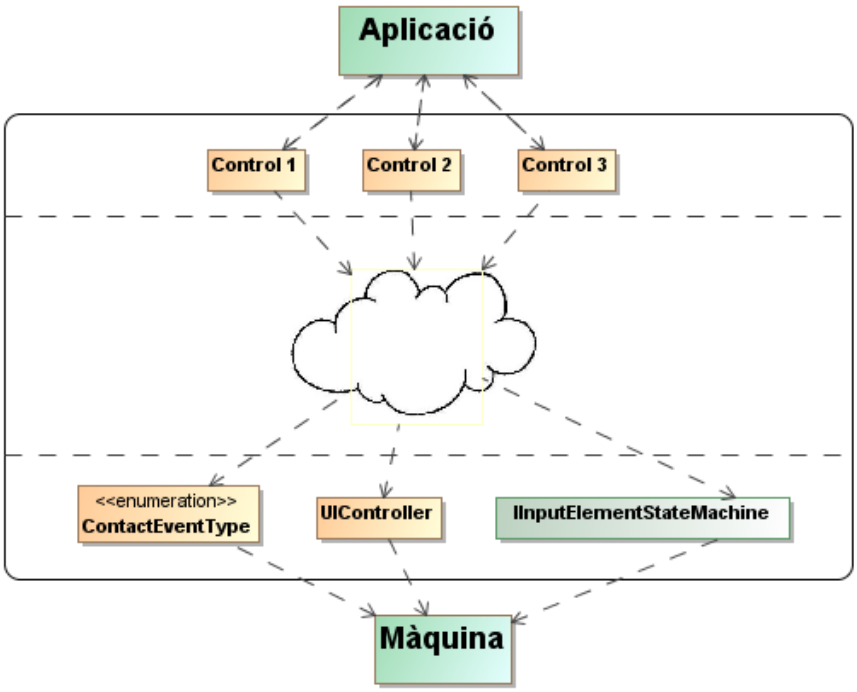


Figura 16. Estructura en tres capes que representa l'abstracció de l'*UIController* per als diferents controls concrets de l'aplicació.

4 Abstracció del Manejador de Contactes

Com s'ha comentat anteriorment, es desitja que el manejador de contactes (última capa de la Figura 16) siga transparent per als diferents controls que es vulguen desenvolupar en les nostres aplicacions. D'aquesta manera, s'aconsegueix aïllar la lògica de les aplicacions de la part més depenent de la màquina, aconseguint així una major reutilització del codi i facilitant una possible portabilitat dels nostres controls cap a altres plataformes tecnològiques. A més, el fet de què cada control concret que es desenvolupe (primera capa de la figura Figura 16) no tinga coneixement del que passa a capes més enllà de la immediatament inferior facilita la construcció d'aquests, ja que no han de preocupar-se per detalls molt concrets dependents de la infraestructura, concentrant-se més en "què es vol fer" que en "com s'ha de fer".

Llavors, es presenta la necessitat d'una capa intermèdia que servisca de pont entre aquestes dues capes que s'han comentat. Les dues classes que formen

aquesta capa s'anomenen *UIElement* i *Control*. La primera, tot i ser proporcionada amb els exemples de la SDK de la Microsoft Surface, ha sigut modificada per fer-la més adient als nostres propòsits.

UIElement representa la part de la vista, és a dir, s'encarrega de la representació gràfica dels elements. D'altra banda, el paper de *Control* és la gestió dels esdeveniments de contacte que proporciona l'*UIController*. Així, la Figura 17 completa la representació gràfica de l'estructura en capes exposada en la Figura 16 amb la inclusió d'aquestes dos noves classes, i la Figura 18 mostra el diagrama de classes per a aquesta capa intermèdia.

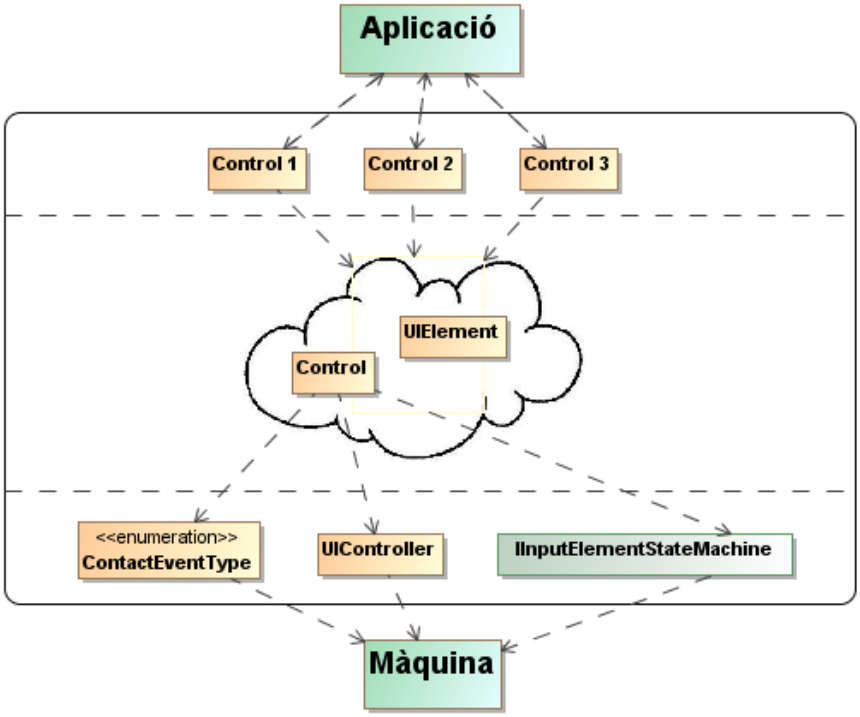


Figura 17. Estructura en tres capes que representa l'abstracció de l'*UIController* per als diferents controls concrets de l'aplicació. Detall de la capa intermèdia.

Així, tot control que es desenvolupe heretarà de la classe *Control* que, al seu torn, hereta d'*UIElement* i implementa la interfície *IInputElementStateMachine*. Com s'ha comentat prèviament, *UIElement* és una classe abstracta (i.e. no hi ha instàncies directes d'aquesta, sinó de les seues extensions) que s'encarrega del dibuixat dels elements. A més, controla les propietats de les dimensions (altura, amplada) així com la posició, la rotació i l'escala de cada control. Per tant, també és l'encarregat de definir el delegat *HitTestCallback* que es va introduir en la secció 3, que crida al mètode *HitTest* també definit en aquesta classe. L'objectiu d'aquest mètode és determinar quan un contacte està dins de la regió delimitada per cada control (per a això fa servir les diferents propietats que defineixen les dimensions de l'objecte gràfic). Per suposat, aquest mètode haurà de ser sobreescrit per cadascun dels controls concrets, ja que cadascun d'aquestos tindrà una forma i unes dimensions diferents.

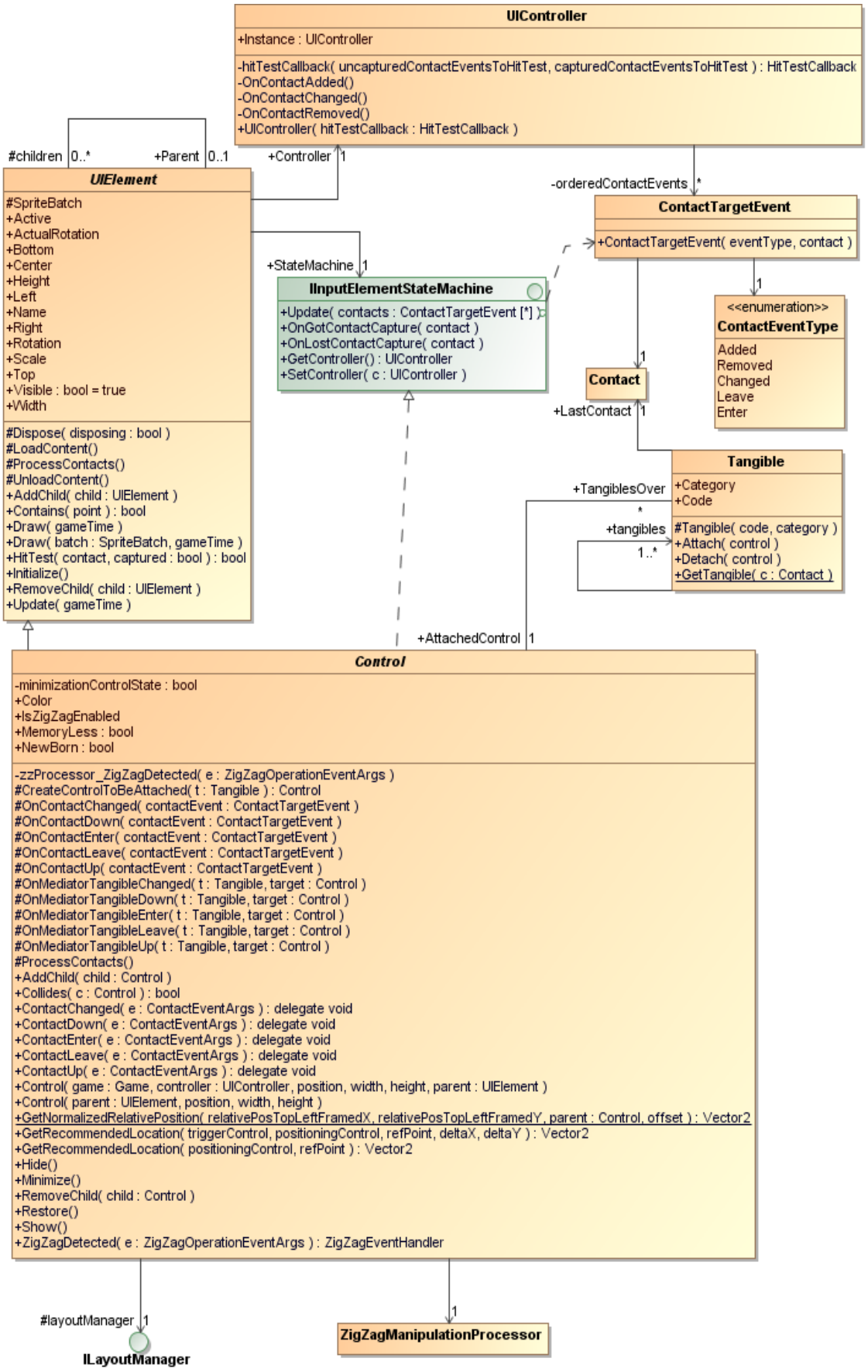


Figura 18. Disseny de les classes que abstrauen el manejador de contactes.

UIElement també s'encarrega de mantindre una jerarquia d'elements gràfics: Cada control pot contindre'n d'altres a dintre seu, formant una jerarquia pare-fills entre ells, de manera que el control contenidor és el pare d'aquells qui estan continguts a dintre seu, però no d'aquells dintre dels seus fills. Aquesta jerarquia és útil, per exemple, per determinar sobre quin control ha entrat un contacte. Si el mètode *HitTest* detecta que un contacte està dins d'un control, llavors aquest crida al *HitTest* dels seus fills, i així recursivament, fins que es detecta el control més baix dins de la jerarquia sobre el que ha entrat el contacte. A la Figura 19 es mostra un exemple d'uns controls rectangulars continguts dins d'altres (esquerra), i com seria la representació de la seua jerarquia en un diagrama d'instàncies (dreta).

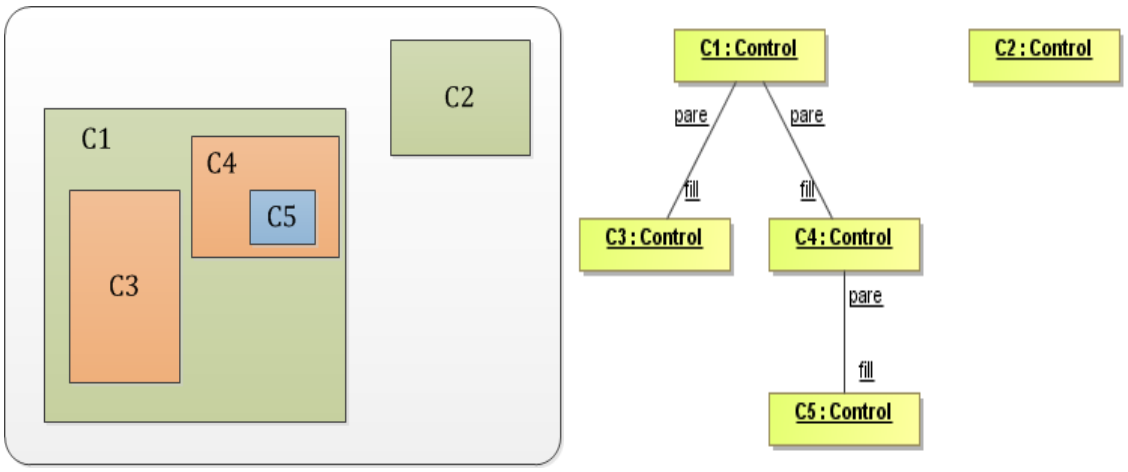


Figura 19. Exemple de jerarquia de controls. Representació gràfica (esquerra) i diagrama d'objectes associat (dreta).

Quant a la classe *Control*, els seus objectius són els següents:

- Gestió de contactes
- Gestió de tangibles
- Distribució dels fills (*layout*)
- Detecció de gestos (zig-zag)

4.1 Gestió de Contactes

Un dels principals objectius de la classe *Control* és la gestió d'esdeveniments de contacte proporcionats per l'*UIController*. Per fer-ho, quan una instància és creada, se li passa una referència a l'objecte *UIController*. Aquest és creat per un control especial anomenat *Surface* que, tal i com s'explicarà a la secció 4.3, serveix de contenidor per a tota la resta d'elements que se situen sobre la taula. L'*UIController*, com ja s'ha explicat, proporciona una sèrie d'esdeveniments de contacte que van dirigits a l'*IInputElementStateMachine* sobre el qual ha impactat. Per això, *Control* implementa aquesta interfície, per a rebre aquests esdeveniments.

Quan l'*UIController* detecta que ha impactat un contacte sobre un control determinat, crida al mètode *Update* d'aquest (vore Figura 15) passant-li una cua amb els esdeveniments ocorreguts. El control, llavors, depenent de quin esdeveniment de contacte haja rebut, llançarà uns altres esdeveniments propis que seran visibles per a tots els controls concrets que hereten de *Control* i per a totes les altres classes de l'aplicació que tinguen una referència a aquests controls (aconseguint, així, l'aïllament de l'*UIController*). Aquests tenen associats uns manejadors dins de la pròpia classe *Control*, que són cridats quan el seu esdeveniment corresponent ocorre, i poden ser sobreescrits per afegir accions pròpies de cada control específic en notar la presència d'un contacte. Els nous esdeveniments són els següents:

- **ContactEnter:** Ocorre quan un contacte entra en aquest control, ja siga sent arrastrat o bé posant un contacte sobre la superfície. Correspon a l'esdeveniment Enter de l'*UIController*. El manejador associat s'anomena *OnContactEnter*.
- **ContactLeave:** Ocorre quan un contacte ix d'aquest control. Pot ser que siga perquè el contacte ha eixit de la superfície o perquè s'ha arrastrat fora de la regió que delimita el control. Correspon a l'esdeveniment *Leave* de l'*UIController* i el seu manejador associat és *OnContactLeave*.
- **ContactDown:** Llançat quan un contacte entra en la superfície sobre aquest control determinat. Correspon a l'esdeveniment *Added* de l'*UIController*. El seu manejador és *OnContactDown*.
- **ContactUp:** És llançat quan un contacte que estava sobre aquest control abandona de sobte la superfície (s'alça). Correspon a l'esdeveniment *Removed* d'*UIController* i el seu manejador s'anomena *OnContactUp*.
- **ContactChanged:** Aquest esdeveniment ocorre quan un contacte que estava dins del control és mogut o rotat i continua estant-hi dins. Correspon al Changed d'*UIController* i el seu manejador associat és anomenat *OnContactChanged*.

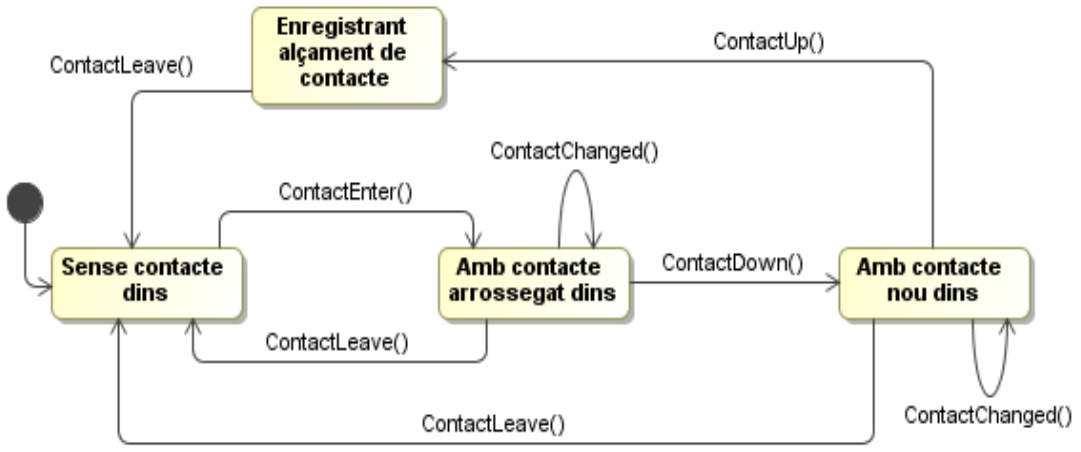


Figura 20. Estats del control per a la gestió d'esdeveniments de contacte.

La Figura 20 mostra per quins estats propis de la gestió de contactes passa el control davant aquests nous esdeveniments. Segons mostra aquesta figura, un *ContactDown* sempre ha de vindre precedit per un *ContactEnter*, i un *ContactUp* sempre ha d'anar seguit d'un *ContactLeave*. Açò podria passar si els contactes simplement es posen sobre la superfície i després s'alcen. Per contra, un *ContactEnter* no implica un *ContactDown*, igual que un *ContactLeave* no implica un *ContactUp*, perquè podria donar-se el cas de què el contacte s'arrossegara per la superfície i isquera d'un control, però es mantinguera dins de la superfície.

4.2 Gestió de Tangibles

En la nostra proposta, es poden tindre elements físics (tangibles), anomenats poms, que continguen elements virtuals. Així, un control pot estar associat a un tangible. Per desvincular-lo, hi ha una propietat anomenada *MemoryLess* a la classe *Control* que, si està activada, en produir-se un *ContactUp*, el tangible quedarà lliure i el control serà esborrat. Aquesta situació es pot donar en escenaris on es vulga controlar un control amb un tangible simplement per comoditat. Per contra, si *MemoryLess* val fals (que és el comportament per defecte), el control queda permanentment associat al tangible fins que es produïsca un gest en zig-zag amb el pom sobre la superfície (similar a com si esborrarem en una pissarra). Per a més informació sobre aquest gest vore la secció 4.5.

Un tangible és representat per la classe *Tangible* (vore Figura 18), que conté informació sobre el codi de l'etiqueta que té apegada, l'últim contacte que ha produït aquest tangible, el control al qual està associat (si ho està) i, per tant, té mètodes per vincular o desvincular un control a aquest tangible: *Attach* i *Detach*.

Quan un tangible "buit" provoca un *ContactEnter* sobre el control, aquest últim crida al mètode *CreateControlToBeAttached* que, sobreescrit en cada subclasse, permet a aquestes crear altres controls que estiguen associats a eixe tangible. A més, existeix una sèrie de mètodes que són cridats quan un tangible amb un control associat genera un esdeveniment de contacte sobre un altre control: *OnMediatorTangibleEnter*, *OnMediatorTangibleLeave*, *OnMediatorTangibleDown*, *OnMediatorTangibleUp* i *OnMediatorTangibleChanged*. El per què d'aquests mètodes ve motivat perquè, amb el disseny adoptat, la lògica de resposta del sistema està dispersa dins de cada control. Amb aquesta solució s'aconsegueix ajuntar la lògica corresponent a un control determinat dins d'ell mateix, encara que el tangible associat, efectivament, impacte sobre un altre objecte. Així, el control sobre el que ha impactat cridarà al corresponent d'aquests mètodes (segons el tipus d'esdeveniment llançat —*ContactEnter*, *ContactLeave*, etc.—) passant-li una referència al control associat, que obté mitjançant la propietat *AttachedControl* del tangible que ha generat el contacte. Vegem el funcionament amb un exemple: Suposem que un tangible *T* té un control

associat *A*. Llavors, produeix un contacte sobre un altre control *B*. *B* cridarà al seu mètode *OnContactDown* en algun moment (anàlogament es cridaria al mètode corresponent en cas de tractar-se d'un altre tipus d'esdeveniment de contacte) i, aleshores, *B* cridarà a *OnMediatorTangibleDown* sobre *A*, tal i com es mostra a la Figura 21. Per obtindre la referència de *A*, *B* segueix l'Algorisme 1 dins del seu manejador *OnContactDown*:

Classe Control
Mètode OnContactDown
Entrada

- contactEvent (ContactTargetEvent): Informació sobre el contacte que ha impactat sobre el control.

Algorisme

```
Tangible t = Surface.Instance.GetTangible(contactEvent.Contact);
if (t != null) {
 if (t.IsAttached) {
  t.AttachedControl.OnMediatorTangibleDown(t);
 }
}
```

Algorisme 1. Fragment del mètode *OnContactDown* d'un control *B* per realitzar la invocació del mètode *OnMediatorTangibleDown* sobre un control *A* associat a un tangible que ha impactat sobre *B*.

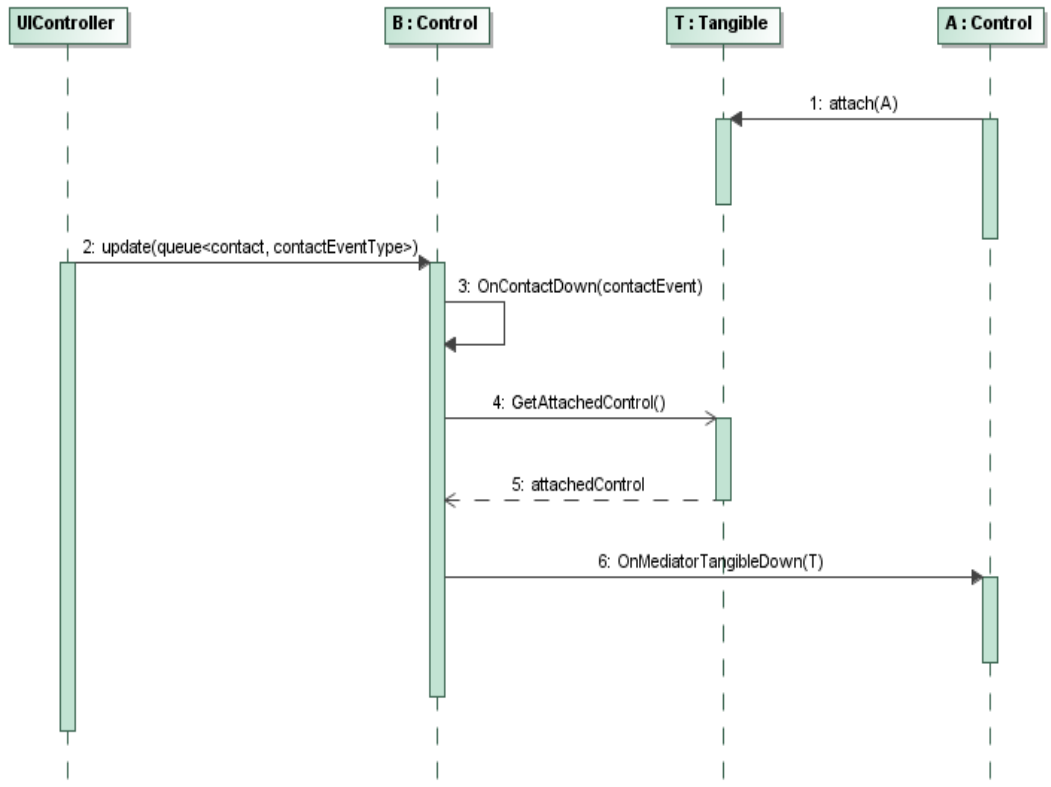


Figura 21. Exemple de funcionament del mètode *OnMediatorTangibleDown* quan un tangible té associat un control *A* i fa un contacte sobre altre control *B*.

4.3 Control Surface

Segons el model de gestió de contactes que s'ha adoptat, on és *Control* qui s'encarrega de la gestió de tots els contactes que impacten sobre ell, necessitem un control especial que faça el paper de base (és a dir, que tota la resta de controls se situen damunt d'ell i tot contacte impacte, o bé sobre aquests o bé sobre el control base). A aquest control base l'hem anomenat *Surface* (vore Figura 22) i té les dimensions de la superfície física per evitar que cap contacte quede sense ser capturat. Seguint de nou un patró *singleton*, només hi ha una instància d'aquesta classe en tota l'aplicació.

Surface, a més, s'encarrega de la creació de la instància d'*UIController* i de passar-li el delegat *HitTestCallback*. Així, el controlador es comunicarà amb el control *Surface* quan detecte algun contacte i li comunicarà la seua evolució.

Quant a l'aspecte visual del control, ja que ocupa tota la superfície i representa el "sòl", se li pot assignar un color de fons gràcies a la propietat *Color* de la superclasse *Control*. A més, com a efecte visual addicional, *Surface* respon als dits creant una animació quan aquests impacten sobre ell. Aquesta consisteix en uns anells al voltant del dit que el segueixen al moure's i, amb el temps, van fent-se més menuts fins desaparèixer (vore Figura 23).

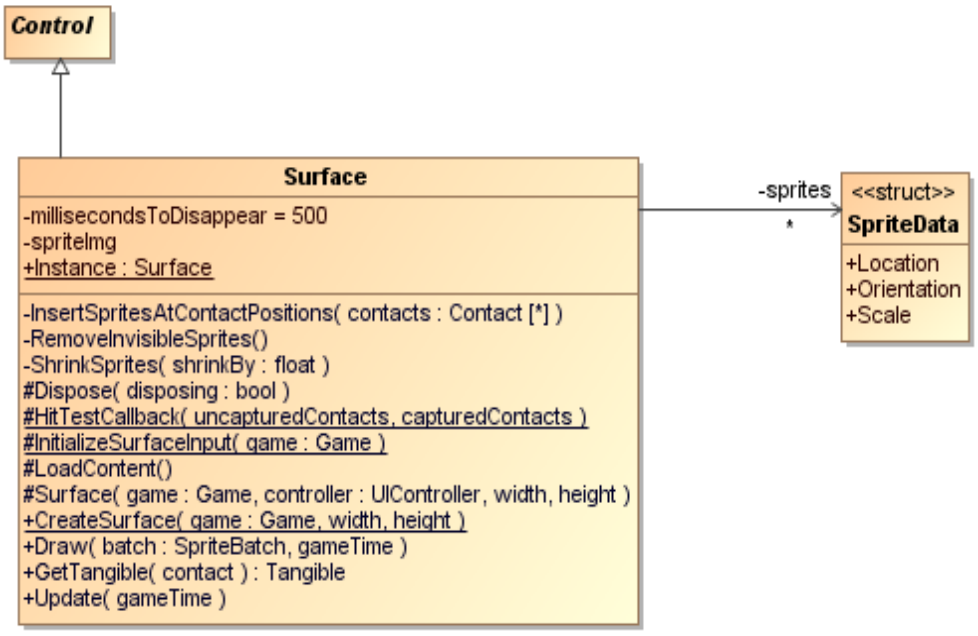


Figura 22. Diagrama de classes del control *Surface*.

Figura 23. Animació del control *Surface* davant la interacció amb els dits.

4.4 Distribució dels Fills (*Layout*)

Al tindre una jerarquia de controls on uns poden estar continguts dins d'altres (vore Figura 19), necessitem un mecanisme que permeta a un control distribuir els seus fills dins l'espai que delimita la seua regió visible, de manera que tots els fills estiguen dins la regió del pare evitant que ocupen la mateixa posició i se solapen. Això no sempre és possible, ja que, al tindre els controls una regió fixa, el nombre de fills que poden contindre és limitat i, a mesura que se'n creen de nous, arribarà un moment on inevitablement es produiran solapaments.

Per gestionar aquesta distribució, cada objecte *Control* té associat un manejador que implementa la interfície *ILayoutManager*. Cada vegada que es crea un control amb un pare (i.e. contingut dins d'un altre), es registra el nou control dins del *ILayoutManager* del pare mitjançant el mètode *Register* especificat en la interfície. Abans d'eliminar un control, l'objecte que l'elimine haurà de cridar al mètode *RemoveChild* del pare del control a eliminar que, entre altres coses, eliminarà el registre d'aquest en l'*ILayoutManager* (mètode *Unregister*) per a què no el tinga en compte a l'hora de calcular la posició en la que s'ha de col·locar un nou control. A banda de registrar-lo, quan es cree un nou control que tinga un pare, el pare farà ús del mètode *GetRecommendedLocation* del *ILayoutManager* per calcular la posició del nou fill.

Seguint el patró de disseny *bridge*, s'han implementat dues versions de manejadors de distribució que implementen *ILayoutManager*: una versió simple (*NaiveLayoutManager*) i una altra més complexa (*LeveledSweepLayoutManager*), que es poden observar a la Figura 24. L'objectiu principal era partir de la simple, que era més ràpida amb pocs controls sobre la superfície perquè elegia la posició aleatòriament, i, posteriorment, implementar una versió més avançada que substituïra la primera, però, al resultar aquesta més lenta, s'han mantingut les dues, podent canviar entre una o altra fàcilment. No obstant, en la pràctica no s'ha trobat que la versió més complexa fora tan lenta que arribara a mo-

lestar a l'usuari, així que aquesta és la que s'està usant actualment i la que s'ha fet servir per als experiments del Capítol 4.

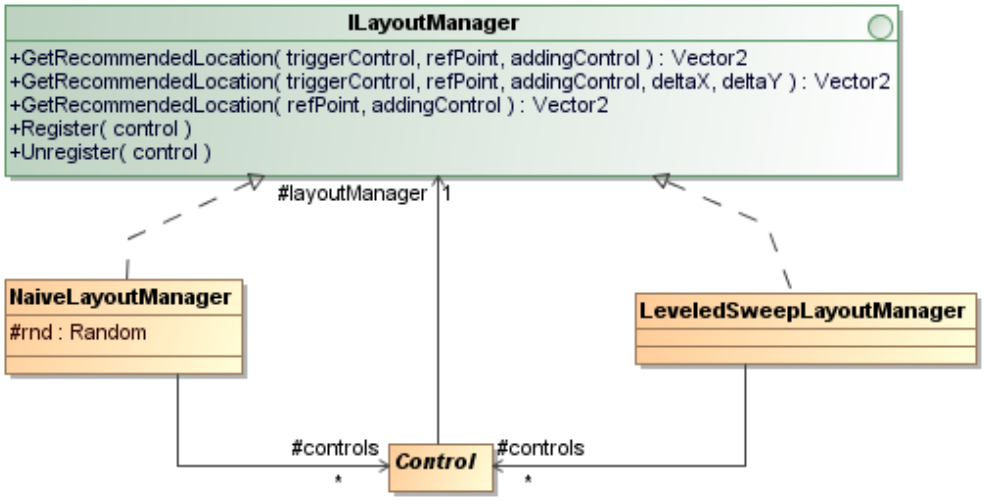


Figura 24. Diagrama de classes dels *layout managers*.

L'algorisme utilitzat per a calcular la posició d'un fill amb el *NaiveLayoutManager* es basa bàsicament en trobar una posició dins de la regió visual del pare de manera que no el faça col·lisionar amb cap altre control, tal i com es mostra en l'Algorisme 2.

D'altra banda, el *LeveledSweepLayoutManager* empra un algorisme (vore Algorisme 3) que, a diferència de l'Algorisme 2, intenta trobar una posició propera al control que invoca al *layout manager* (sempre amb la restricció de que no es produïsquen col·lisions). Per a fer-ho, realitza una exploració dibuixant un rectangle al voltant d'aquest tal i com es mostra a la Figura 25. Tal i com s'observa en aquesta, la posició del nou control estarà, idealment, al rectangle de primer nivell (i.e. el rectangle imaginari més proper al control invocador). Si no pot col·locar-se en aquest perquè col·lisiona amb algun altre control, passarà al segon, i així successivament. La distància entre nivells ve donada per dos paràmetres, δ_x i δ_y, que se li passen com entrada al mètode *GetRecommendedLocation*. Si el control no es pot posicionar en cap lloc dins del control pare, llavors se li dóna una posició qualsevol, malgrat que col·lisionarà amb algun altre control. L'alternativa seria no permetre que es col·locara cap control nou, però no s'ha considerat acceptable donat que limitaria la llibertat dels usuaris. En un futur s'intentarà dissenyar una forma més intel·ligent de distribuir els elements com, per exemple, permetre que els controls es reposicionen automàticament per deixar lloc al nou.

Classe NaiveLayoutManager
Mètode GetRecommendedLocation
Entrada

- triggerControl: Control que provoca la invocació del mètode.
- addingControl: Control que serà posicionat dins del control pare. Precisa estar ja creat i tindre les seues dimensions ben determinades (i.e. les propietats Width, Height, Top, Bottom, Left, Right i Center sobreescrites).

Eixida

- Posició (x,y) recomanada per a col·locar el menú fill addingControl.

Variables auxiliars

- XMax: Amplada màxima del control pare en píxels.
- YMax: Altura màxima del control pare en píxels.

Algorisme

```
do {
 collides = false;
 x = random(XMax);
 y = random(YMax);
 addingControl.Center = (x,y);

 if (addingControl.Collides(triggerControl))
  collides = true;
 else {
  for (i = 0; i < controls.count && !collides; i++) {
   if (controls[i] != addingControl &&
       controls[i].Collides(addingControl))
    collides = true;
  }
 }
} while (collides);

return (x,y);
```

Algorisme 2. Mètode GetRecommendedLocation de la classe NaiveLayoutManager.

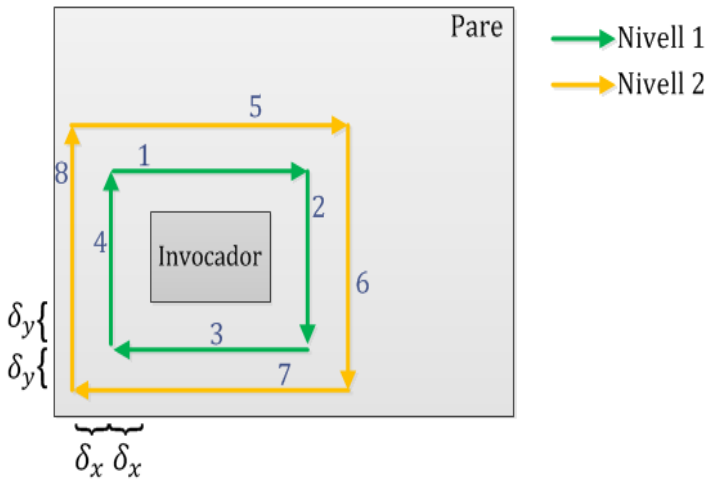


Figura 25. Trajectòria de cerca de la posició on col·locar el control fill al mètode *GetRecommendedLocation* de la classe LeveledSweepLayoutManager. Els nombres damunt de les fletxes indiquen l'ordre que segueix la trajectòria.

Classe LeveledSweepLayoutManager
Mètode GetRecommendedLocation
Entrada

- triggerControl: Control que provoca la invocació del mètode.
- addingControl: Control fill que serà posicionat dins del control pare. Precisa estar ja creat i tindre les seues dimensions ben determinades (i.e. les propietats Width, Height, Top, Bottom, Left, Right i Center sobreescrites).
- refPoint: Punt de referència al voltant del qual es calculen els rectangles imaginaris per posicionar addingControl. Generalment serà el centre del control invocador però, de forma genèrica, podria ser qualsevol altre punt.
- δ_x: Distància sobre l'eix X entre nivells i entre el primer nivell i el punt refPoint.
- δ_y: Distància sobre l'eix Y entre nivells i entre el primer nivell i el punt refPoint.

Eixida

- Posició (x,y) recomanada per a col·locar el menú fill addingControl.

Constants
Algorisme

```
// Establir límits dins del control pare
Wmin = addingControl.Width / 2;
Hmin = addingControl.Height / 2;
Wmax = triggerControl.Parent.Width - Wmin;
Hmax = triggerControl.Parent.Height - Hmin;

ret = (-1, -1);

level = 0;
exploration_completed = false;

do {
 level++;

 boundYU = max(Hmin, refPoint.Y - (level * δy));
 boundYL = min(Hmax, refPoint.Y + (level * δy));
 if (boundYU > boundYL)
  boundYU = boundYL;

 boundXL = max(Wmin, refPoint.X - (level * δx));
 boundXU = min(Wmax, refPoint.X + (level * δx));
 if (boundXL > boundXU)
  boundXL = boundXU;

 // Mirem costat superior del rectangle imaginari
 for (x = boundXL, y = boundYU; x <= boundXU; x += δx) {
  collides = false;
  // Buscar col·lisions
  for (i = 0; i < controls.Count && !collides; i++) {
   addingControl.Center = (x, y);
   if (controls[i].Collides(addingControl) &&
       addingControl != controls[i])
    collides = true;
```

```
  }
  // Si no col·lisiona amb ningú, parem de cercar
  if (!collides) {
   ret = (x, y);
   return ret;
  }
 }

 // Mirem costat dret del rectangle imaginari
 for (x = boundXU, y = boundYU; y <= boundYL; y += δy) {
  collides = false;
  // Buscar col·lisions
  for (i = 0; i < controls.Count && !collides; i++) {
   addingControl.Center = (x, y);
   if (controls[i].Collides(addingControl) &&
       addingControl != controls[i])
    collides = true;
  }
  // Si no col·lisiona amb ningú, parem de cercar
  if (!collides) {
   ret = (x, y);
   return ret;
  }
 }

 // Mirem costat inferior del rectangle imaginari
 for (x = boundXU, y = boundYL; x >= boundXL; x -= δx) {
  collides = false;
  // Buscar col·lisions
  for (i = 0; i < controls.Count && !collides; i++) {
   addingControl.Center = (x, y);
   if (controls[i].Collides(addingControl) &&
       addingControl != controls[i])
    collides = true;
  }
  // Si no col·lisiona amb ningú, parar de cercar
  if (!collides) {
   ret = (x, y);
   return ret;
  }
 }

 // Mirem costat esquerre del rectangle imaginari
 for (x = boundXL, y = boundYL; y >= boundYU; y -= δy) {
  collides = false;
  // Buscar col·lisions
  for (i = 0; i < controls.Count && !collides; i++) {
   addingControl.Center = (x, y);
   if (controls[i].Collides(addingControl) &&
       addingControl != controls[i])
    collides = true;
  }
  // Si no col·lisiona amb ningú, parem de cercar
  if (!collides) {
   ret = (x, y);
```

```
    return ret;
   }
  }

  if ((boundYL >= Hmax && boundYU <= Hmin) &&
      (boundXL <= Wmin && boundXU >= Wmax))
   exploration_completed = true;
} while (!exploration_completed);

if (ret == (-1, -1)) {
 // No hi ha lloc. Donar posició qualsevol:
 // Intentem a la dreta del control invocador...
 ret.X = triggerControl.Center.X + triggerControl.Width;
 // Si se n'eixim del pare, a l'esquerra
 if (ret.X > Wmax)
  ret.X = triggerControl.Center.X - triggerControl.Width;
 // Intentem dalt del control invocador...
 ret.Y = triggerControl.Center.Y - triggerControl.Height;
 // Si se n'eixim del pare, baix
 if (ret.Y < Hmin)
  ret.Y = triggerControl.Center.Y + triggerControl.Height;
}

return ret;
```

Algorisme 3. Mètode *GetRecommendedLocation* de la classe *NaiveLayoutManager*.

4.5 Detecció de Gestos (Zig-Zag)

A les nostres aplicacions que fan ús de TangiWheel, per permetre desvincular un pom d'un control virtual, l'usuari ha de realitzar un gest de Zig-Zag sobre el control *Surface*. No obstant, cada subclasse de *Control* pot tindre un comportament específic associat a aquest gest, sense la necessitat de què siga esborrar res.

La classe *ZigZagManipulationProcessor* s'encarrega de detectar si un conjunt d'esdeveniments de contacte *ContactChanged* corresponen a un gest de zig-zag. Aquests esdeveniments són passats per un *Control* determinat (no necessariament *Surface*). Llavors, quan un gest d'aquest tipus siga detectat, el processador de zig-zag llançarà un esdeveniment *ZigZagDetected* que, si és capturat pel seu *Control* associat, aquest últim podrà decidir quina operació realitzar com a manejador de l'esdeveniment (no necessariament esborrar). A continuació s'explica de quina manera es detecta el gest desitjat.

Per a cada contacte processat per l'objecte *Control* associat, el *ZigZagManipulationProcessor* crea un *ZigZagManipulator* (vore Figura 26), que conté la posició (x, y) del contacte més un *timestamp* amb el moment en el que aquest ha realitzat un *ContactChanged*. A més, conté un atribut *direction*, que indica la direcció en codi Freeman [13] que porta el contacte actual. Tots els manipuladors se'ls guarda en la llista *manipulatorPool*.

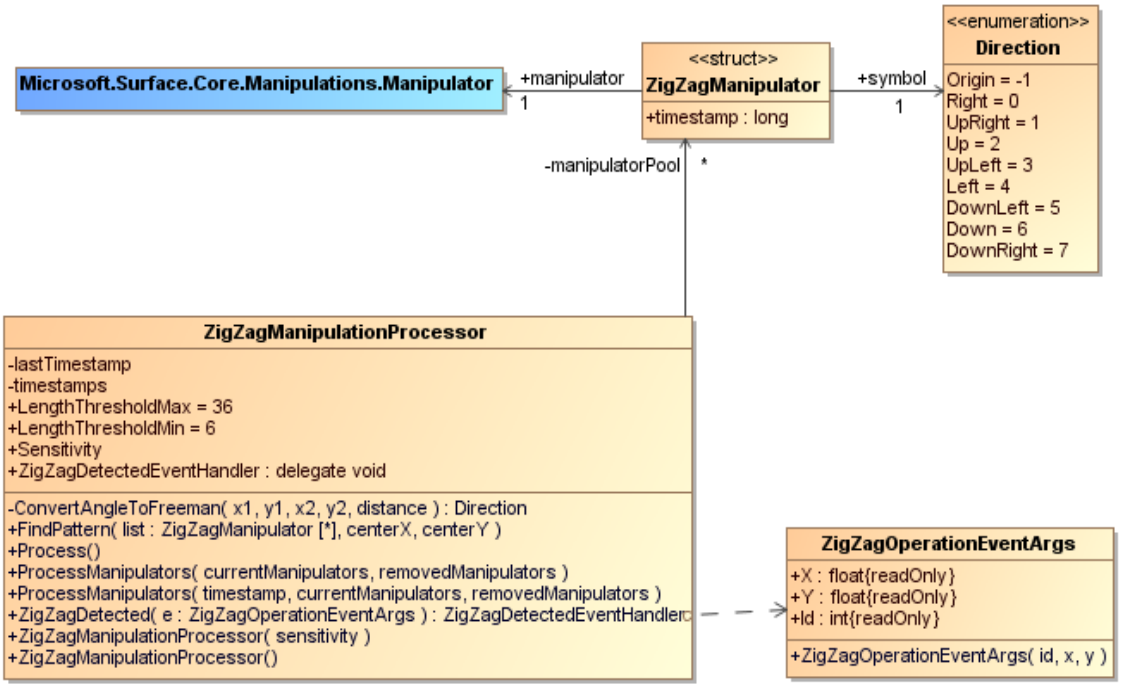


Figura 26. Estructura de classes per al *ZigZagManipulationProcessor*.

El següent pas és calcular la direcció que porta cada contacte (o manipulador) de la llista. Calculant el vector que formen les posicions de dos contactes consecutius, a continuació es transforma l'angle que formen en el codi Freeman corresponent, tal i com indica la Figura 27. D'aquesta forma s'obté una representació sintàctica de la llista de posicions que el contacte ha ocupat al llarg del temps seguint codis Freeman.

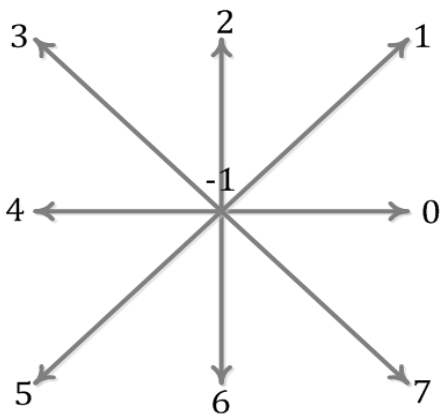


Figura 27. Codi Freeman associat a un angle simbolitzat per les fletxes.

En aquest punt, es realitzen una sèrie de filtratges als manipuladors, per obtindre només aquells que són representatius. Primer, es recorre la llista de manipuladors eliminant aquells que estiguen molt propers. Per a això s'empra l'atribut *sensitivity*: tot manipulador que estiga a una distància de l'anterior menor que *sensitivity* és descartat. També es descarta si ha passat prou de temps (500ms) des de l'últim contacte, ja que un gest de zig-zag ha de ser relati-

vament ràpid. Tot seguit, si el nombre total de manipuladors és menor que un cert llindar (*lengthThresholdMin*), aquests són descartats (com ocorre a la Figura 28-dreta), car es considera que es necessita un nombre de contactes mínim per a què realment es produïsca un zig-zag.

Així, en aquest moment es té una cadena de direccions pertanyents a un alfabet Σ de símbols de codi Freeman que, per a què siga considerada un zig-zag, ha de coincidir amb el següent patró p. D'aquesta comprovació s'encarrega el mètode *FindPattern*.

$$p = (s^m r^q s^k r^t)^+$$
$$m, q, k, t > T$$
$$s, r \in \Sigma$$
$$T \in \mathbb{N} \text{ llindar}$$
$$\Sigma = \{-1,0,1,2,3,4,5,6,7\}$$

Si el nombre de símbols de cada direcció supera un llindar vol dir que el gest és suficientment llarg per a ser considerat un zig-zag i no soroll i, aleshores, el següent pas és produir una nova cadena de direccions Freeman "simplificada", de forma que cada símbol tinga només un element. Així, el nou patró que seguiria aquesta cadena seria $p' = (srsr)^+$. És a dir, agafant grups de quatre símbols (o, el que és el mateix, de quatre direccions distintes de moviment d'un contacte), el primer ha de coincidir amb el tercer i el segon amb el quart, com es pot observar a la Figura 28-esquerra, on es tenen dos grups de quatre direccions amb el codi Freeman associat a cada seqüència de punts de contacte. Els punts representen tots els contactes enregistrats, però només els grossos són significatius (els menuts són soroll).

Finalment, l'últim pas és comprovar si els símbols anteriors s i r són corresponen a direccions oposades (vore Figura 27). Açò és senzill ja que, gràcies als codis emprats, r i s seran oposats si, i només si, $s = (r + 4)\ mod\ 8$. D'aquesta manera, si es troba que la llista de manipuladors correspon amb un patró de zig-zag, el *ZigZagManipulationProcessor* llança l'esdeveniment *ZigZagDetected* anteriorment mencionat, permetent que el *Control* que tinga associat el capture per realitzar l'operació pertinent.

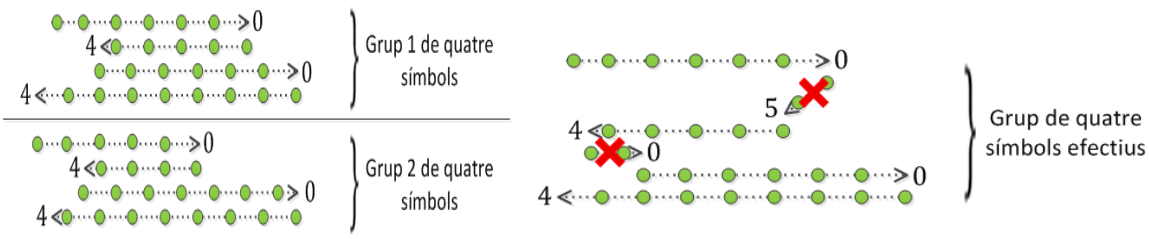


Figura 28. Representació d'una seqüència de punts de contacte com a dos grups de quatre direccions Freeman que representen un gest de zig-zag amb soroll (esquerra) i d'un altre grup de direccions Freeman en el qual s'han descartat seqüències per contindre un nombre massa menut de contactes (dreta).

Capítol 3.
Disseny i Implementació

1 Organització dels Elements

TangiWheel organitza una sèrie d'elements al voltant d'un centre seguint el model de menú de pastís circular. S'ha considerat apropiada aquesta organització car no només és compacta, sinó que, a més, redueix el temps de cerca i fixa la distància entre els elements, fet que disminueix els errors de selecció [19]. En contraposició, no s'ha considerat una organització lineal (com les usades en aplicacions de finestres mono usuari), tot i que podria proporcionar una distribució dels elements més còmoda per a molts usuaris. Aquesta decisió s'ha pres perquè TangiWheel està pensat per a la manipulació, per part de diversos usuaris simultàniament, de col·leccions que poden contindre text, imatges, o ambdós al mateix temps. Representar les col·leccions en forma de llista lineal podria resultar més còmode per a un sol usuari, però limitaria l'accés als elements a altres usuaris situats en diferents costats d'una taula interactiva.

2 Gestió dels Elements Representats

TangiWheel mostra només un subconjunt dels elements d'una col·lecció. D'aquesta manera s'aconsegueix que la grandària del control siga independent del nombre d'elements que continga. Aquesta compacitat és important, ja que ens permet tindre diverses col·leccions representades sobre la superfície sense que les més grans ocupen tot l'espai disponible. Per explorar els elements no visibles, TangiWheel té una regió especial, la "*bridge mark*" (vore secció 2.2), localitzada a la part de dalt del menú que, amb un funcionament similar al d'un botó, permet canviar entre mode d'"exploració" i mode de "rotació". Així, en mode rotació, rotar el control implica la rotació física de tot el menú, mentre que, en mode exploració, provoca que els elements llisquen sobre la superfície del TangiWheel en el sentit de la rotació, ocultant els visibles sota un costat de la regió especial i mostrant-ne els ocults per l'altre costat.

Mantenint polsada la *bridge mark* durant dos segons, el control passa a un estat de minimització on tots els elements s'amaguen i només és visible aquesta marca, obtenint així una màxima compactació en el cas de massificació de controls sobre la superfície.

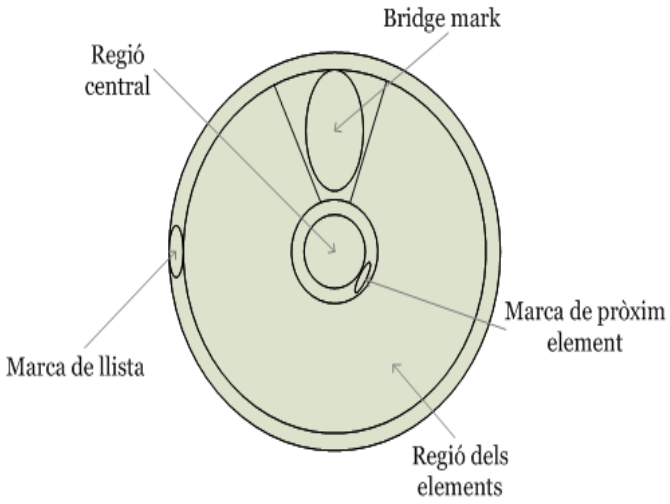


Figura 29. Estructura d'una instància de TangiWheel.

Altra característica important inclosa en TangiWheel és un punt brillant en el perímetre del pastís anomenat "marca de llista" (explicada a la secció 2.4). Com se suporten col·leccions virtualment il·limitades, aquest indicador dóna als usuaris una pista de quina és la posició relativa dels ítems visibles dins de la llista completa (similar a la caixa que proveeixen les tradicionals barres de desplaçament). Tot i no estar implementat actualment, una millora addicional consistiria en mostrar una previsualització d'uns pocs elements ocults, amb la finalitat d'ajudar a decidir a l'usuari cap a quina direcció han d'explorar. Altra millora de la representació dels elements quan es treballe amb llistes molt grans podria ser representar una llista d'elements recentment accedits per poder-hi accedir directament. Aquests afegits seran inclosos en futures versions del menú.

La "marca de pròxim element" (vore apartat 2.5) proporciona informació de quant cal continuar realitzant un moviment de rotació per a què es mostre un nou ítem ocult.

Finalment, la "regió central" només és útil quan tenim instàncies del control que no són controlades mitjançant un tangible, ja que és la receptora de les interaccions amb els dits de l'usuari que permeten la seua manipulació. Més detalls a la secció 2.6.

2.1 Distribució dels Elements Visibles

2.1.1 Tipus d'Elements i Orientacions

Tal i com es pot observar a la Figura 30, TangiWheel suporta tres tipus d'elements, depenent de la seua representació visual:

- Imatge: Ítems representats només per una imatge. Els formats suportats són .bmp, .dds, .dib, .hdr, .jpg, .pfm, .png, .ppm i .tga. L'orientació d'aquests ítems és l'anomenada "normal" (i.e. mirant cap al centre).

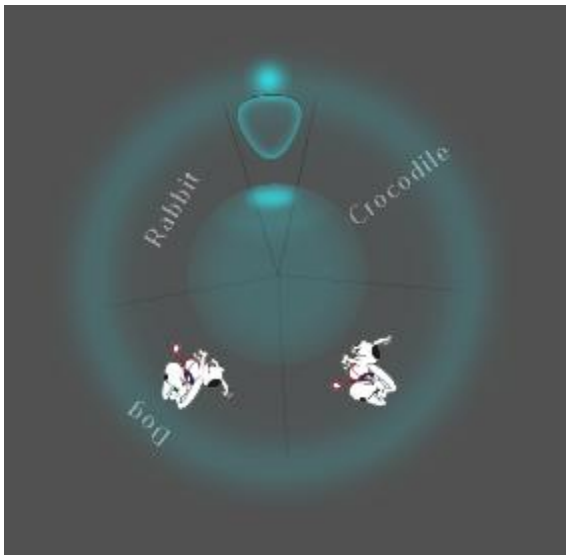


Figura 30. TangiWheel amb els distints tipus d'elements que pot contindre (imatge, imatge i text, text normal i text radial).

- Imatge + Text: Aquests elements estan formats per una imatge juntament amb un text situat al davall. L'orientació és normal.
- Text: Elements representats només amb un text. Aquest tipus suporta dos tipus d'orientació. La normal i també una anomenada "radial", on el text es col·loca apuntant cap al centre, de forma que, si és molt llarg, se n'ix del menú. Aquesta orientació és útil quan es tenen textos llargs que no caben bé al sector. A la Figura 31 es pot observar aquesta situació.

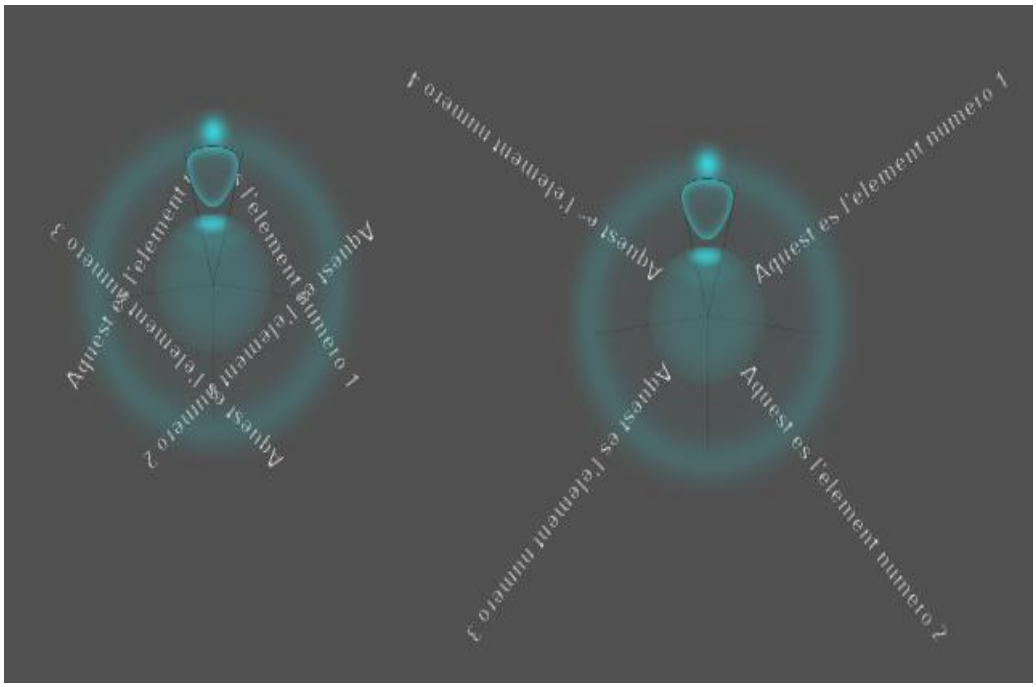


Figura 31. Problemes amb l'orientació normal i textos llargs (esquerra) i solució adoptada amb l'orientació radial (dreta).

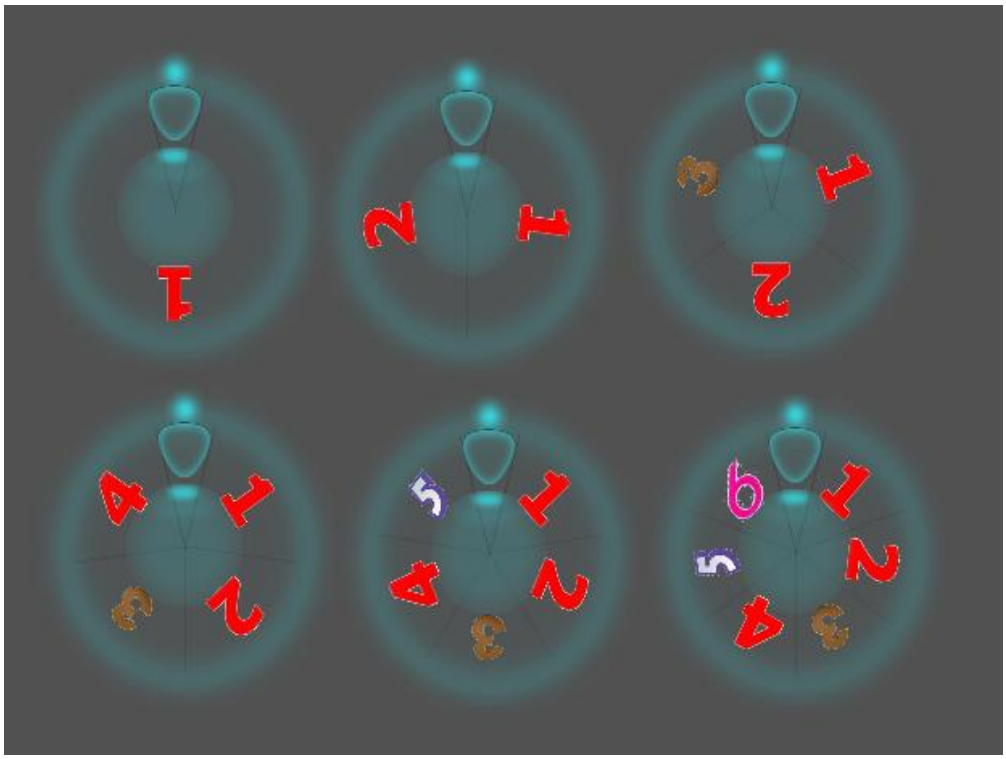


Figura 32. Instàncies TangiWheel variant el nombre de sectors.

2.1.2 Establiment de les Posicions

Els ítems d'un TangiWheel són representats al voltant del centre del menú organitzats en sectors (vore Figura 32). El nombre de sectors $nSectors$ depèn tant del nombre d'elements que continga la col·lecció representada $nItems$ com del màxim nombre d'elements $maxVisibleItems$ que s'han establert com per a ser visibles al mateix temps. Aquest últim paràmetre es pot establir com una propietat del control, permetent al programador establir quant sobrecarregat estarà el mateix (si no s'altera, per defecte val 5). Així, el nombre de sectors es calcula com segueix:

$$nSectors = \min(nItems, maxVisibleItems)$$

Una vegada establert el nombre de sectors ja es pot calcular l'angle θ de cadascun dels sectors que seran dibuixats, les unitats del qual són radians. Aquest depèn també de l'angle de la regió de la *bridge mark* Θ_{bm}(que val $\frac{\pi}{6}$ radians —30º— si no és modificat pel programador mitjançant una propietat) i es calcula d'aquesta manera:

$$\theta = \frac{2\pi - \Theta_{\mathrm{bm}}}{nSectors}$$

Arribats a aquest punt, només queda calcular les posicions dels elements visibles i de les línies que delimiten els sectors per a ser dibuixats posteriorment. Donada la representació circular del menú, aquestes posicions es donaran en coordenades polars relatives al centre del control. Per calcular-les, es calcula un

angle addicional, Θ_{L0}, que indica l'angle sobre l'eix X del sistema de coordenades (cartesià, amb origen en el centre del control) en el que comença la regió de la b*ridge mark*. A partir de Θ_{L0} es calculen tots els demés angles que es faran servir per situar els ítems i les línies. La Figura 33 mostra el sistema de coordenades relatiu al control del que es parlava anteriorment, sobre el qual s'ha situat Θ_{L0}i els angles corresponents a les línies que s'haurien de dibuixar en una situació hipotètica de tindre $nSectors = 2$. El càlcul d'aquest angle inicial es realitza de la següent forma:

$$\Theta_{L0} = \frac{\pi}{2} - \frac{\Theta_{bm}}{2}$$

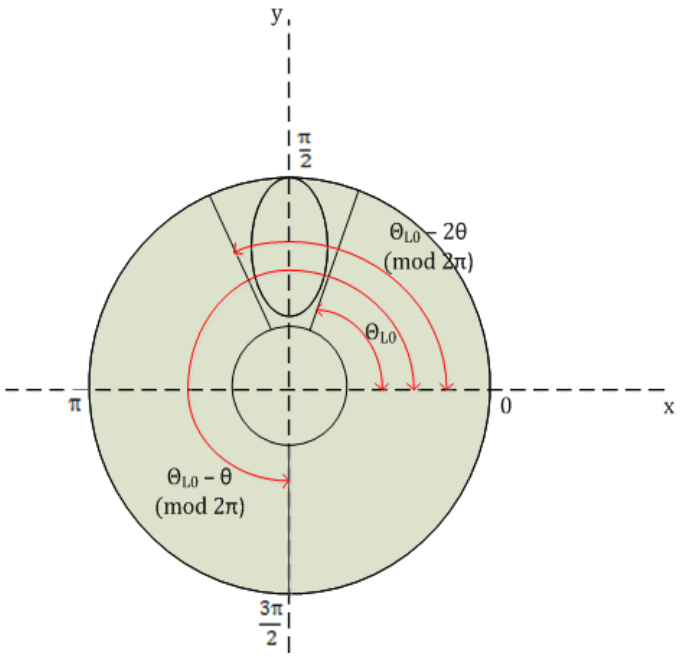


Figura 33. Sistema de coordenades local d'un TangiWheel.

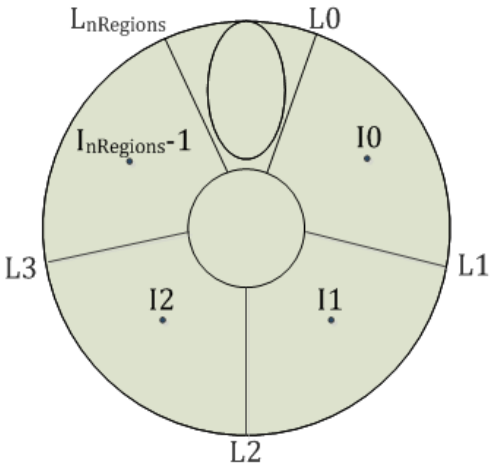


Figura 34. Numeració de les línies que delimiten els sectors i dels ítems.

Cadascuna de les línies a dibuixar (les que delimiten els sectors) té una longitud igual al radi R del TangiWheel, i el seu extrem esquerre està situat sobre el

centre, amb la rotació corresponent. Les seues posicions P_L (on P_{L0} és la posició de la línia que correspon a l'angle de referència Θ_{L0} i la resta es numeren correlativament, tal i com es mostra en la Figura 34) són, per tant:

$$P_{Ll} = (0, \Theta_{Ll})$$
$$\Theta_{Ll} = (\Theta_{L0} - l \cdot \theta) \bmod 2\pi$$
$$l \in [0, nRegions]$$

De forma similar, els diferents elements de la col·lecció són situats en la bisectriu de l'angle del seu sector corresponent, a una distància d del centre. A la Figura 35 es pot vore gràficament quina és aquesta distància, que ve donada en funció de R i de r (el radi de la regió central). Així, les posicions dels ítems P_I es calculen com segueix:

$$P_{Ii} = (d, \Theta_{Ii})$$
$$d = \begin{cases} r + \dfrac{R - r}{2}, & \text{si orientació normal} \\ r, & \text{si orientació radial} \end{cases}$$
$$\Theta_{Ii} = (\Theta_{L0} - \frac{\theta}{2} - i \cdot \theta) \bmod 2\pi$$
$$i \in [0, nRegions - 1]$$

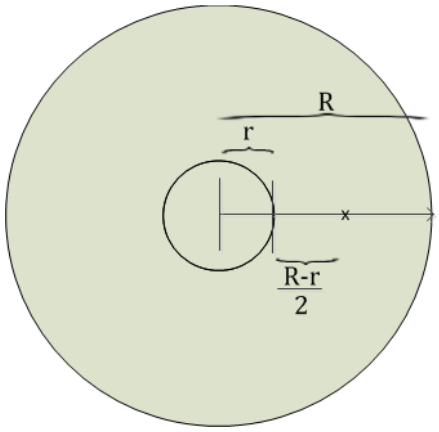


Figura 35. Representació gràfica de d en funció de R i r.

2.2 Bridge Mark

Tal i com es pot vore a la Figura 29, la *bridge mark* és un botó virtual situat en una regió especial de tota instància TangiWheel, i té un objectiu doble. Per una part, permet a l'usuari canviar entre mode l'exploració i de rotació. D'altra banda, permet realitzar la minimització del control (i la seua posterior restauració), de manera que ocupe menys espai i permeta la instanciació de més TangiWheel sobre la superfície.

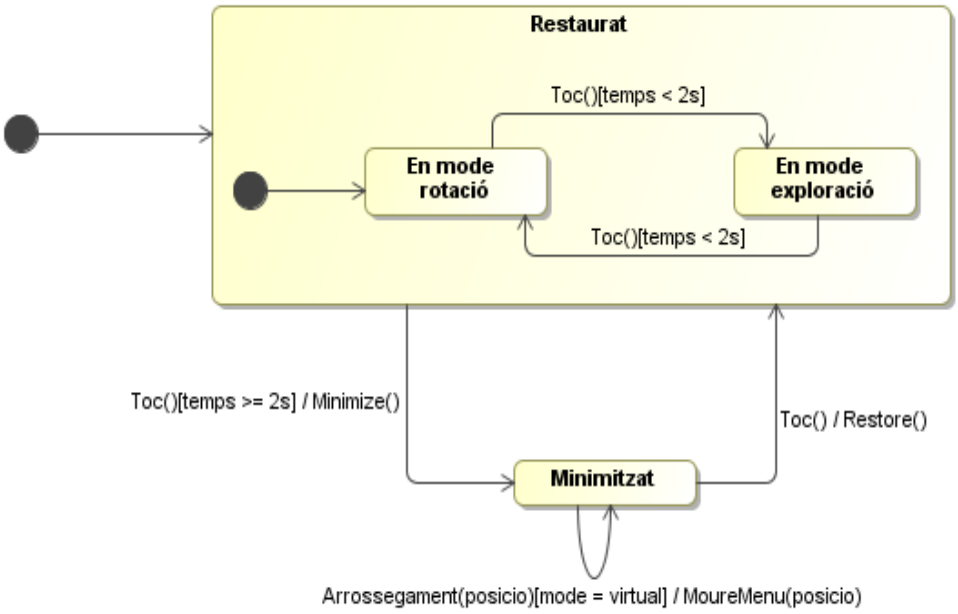


Figura 36. Comportament del menú davant d'interaccions de l'usuari sobre la *bridge mark*.

La Figura 36 mostra un diagrama de transició d'estats (STD) amb el comportament del control davant interaccions de l'usuari sobre la *bridge mark* i els estats pels que passa. Com es pot observar, el control pot passar per dos estats principals: "Restaurat" o "Minimitzat". El primer és aquell que apareix per defecte, on tots els elements són visibles, mentre que el segon deixa només visible aquesta regió especial, obtenint una major compactació (vore Figura 37). Dins de l'estat restaurat, es pot estar en mode de rotació o mode d'exploració. En rotació, el control respon amb una reorientació de tots els seus elements davant un gest de rotació, mentre que, en exploració, aquest gest provoca l'aparició d'elements fins ara ocults. Les transicions d'un estat a un altre es comenten a continuació:

- **En mode rotació → En mode exploració:** Realitzant un toc amb el dit sobre aquesta regió especial durant menys de 2 segons.
- **En mode exploració → En mode rotació:** Realitzant un toc amb el dit sobre la regió del *bridge mark* durant menys de 2 segons.
- **Restaurat → Minimitzat:** Realitzant un toc amb el dit durant 2 segons o més. Abans de canviar d'estat, s'invoca al mètode *Minimize*, que prepara el control per a la minimització tant a nivell lògic com visual.
- **Minimitzat → Restaurat:** Realitzant un toc amb el dit. Abans de canviar l'estat, s'invoca al mètode *Restore*, que prepara al control per a la seua restauració (torna a mostrar tots els ítems).
- **Minimitzat → Minimitzat:** Aquesta transició només té sentit quan el control no té cap tangible associat i és controlat només amb els dits. Si, en compte d'un toc, es realitza un gest d'arrossegament sobre la *bridge mark* quan el control està minimitzat, provoca la translació d'aquesta seguint al dit que l'ha provocada.

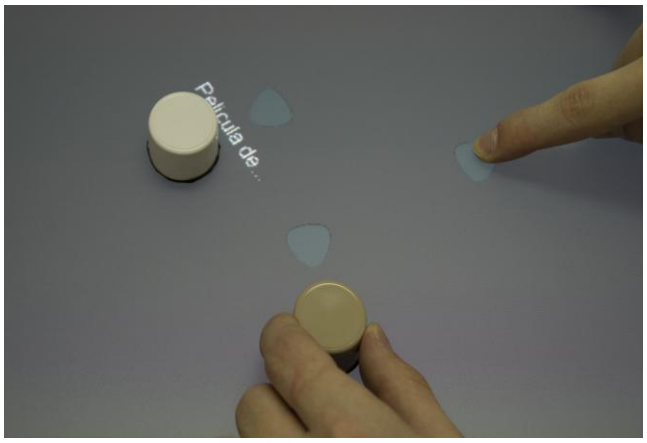


Figura 37. Menú minimitzat controlat per un tangible i controlat pels dits.

Per a aquesta última transició, la forma de saber la diferència entre un esdeveniment de toc i altre d'arrossegament s'explica en la Figura 38. S'hi fan ús dels esdeveniments *ContactDown*, *ContactChanged* i *ContactLeave* explicats al Capítol 2.

Cal destacar que, malgrat els estats marcats com a inicials al diagrama i les interaccions comentades, el programador pot canviar l'estat mitjançant la propietat booleana *RotationMode* (que, posada a vertader indica que el mode actual és de rotació i, a fals, d'exploració) i mitjançant la invocació als mètodes *Minimize* i *Restore* per canviar entre els modes minimitzat i restaurat, respectivament.

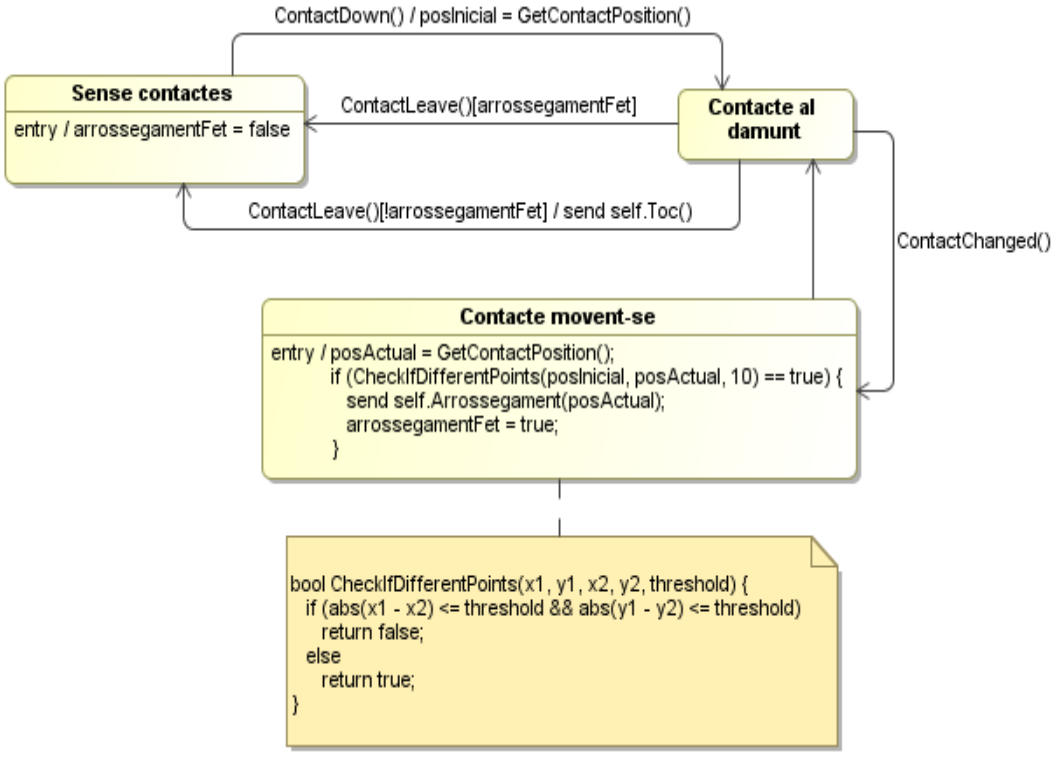


Figura 38. Forma de diferenciar entre un esdeveniment de toc i altre d'arrossegament sobre la *Bridge mark* quan el TangiWheel no té cap tangible associat.

2.3 Exploració dels Elements

Com ja s'ha comentat prèviament, l'objectiu principal de TangiWheel és explorar una sèrie d'elements dins d'una col·lecció. En l'apartat 2.1 s'ha contat com són dibuixats els elements visibles d'un TangiWheel (que es col·loca un dins de cadascun dels $nSectors$ sectors del menú). Però, què passa amb la resta?

A la secció 2.2 s'ha explicat com arribar a un estat d'exploració d'elements. Aquesta secció versa sobre com s'organitzen els elements de la col·lecció que representa TangiWheel i com es visualitza la ja esmentada exploració. També s'explicarà com s'aconsegueix l'efecte de què els ítems vells s'amaguen sota la regió de la *bridge mark* i els nous n'hi ixen (vore Figura 39).

Els elements s'emmagatzemen en una llista, $items$. Com no tots poden ser visibles a la vegada, es manté una finestra d'elements dibuixats controlada per un índex $drawItemIndex$, que indica l'inici de la finestra (i.e. l'índex del primer element a dibuixar en el primer sector en sentit antihorari). El final de la finestra és, simplement, $(drawItemIndex + nSectors - 1) \bmod nItems$, per aconseguir l'efecte de llista circular. Inicialment, la finestra està situada al principi de la llista d'elements i, en realitzar una exploració, el que es fa és moure aquesta finestra en el sentit en què l'usuari indique. Així, si l'exploració es realitza amb un gest en sentit horari, la finestra avançarà (incrementant el valor de $drawItemIndex$ en mòdul $nItems$), mentre que, si es fa en sentit antihorari, es mourà cap enrere (decrementant-ne el valor en mòdul $nItems$). La Figura 40 mostra un exemple de com canvia aquesta finestra davant d'exploracions en ambdós sentits, amb un nombre d'elements en la col·lecció $nItems = 10$ i un nombre d'elements visibles a la vegada $nSectors = 5$.

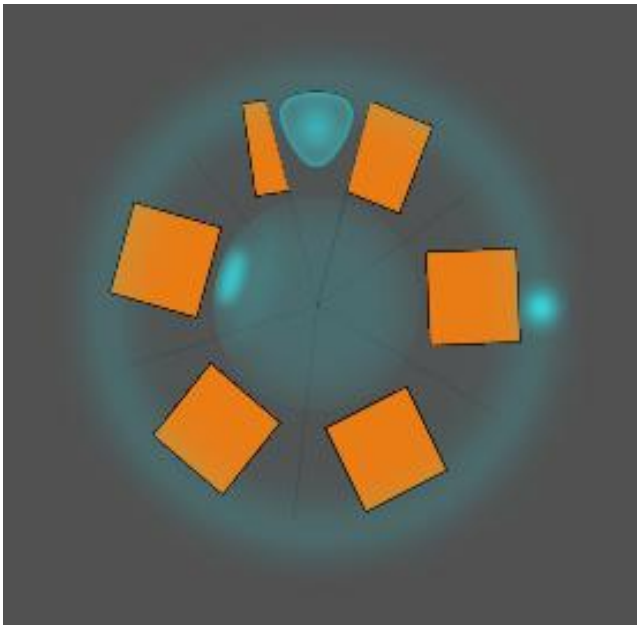


Figura 39. TangiWheel explorant elements. Efecte d'oclusió sota la *bridge mark*.

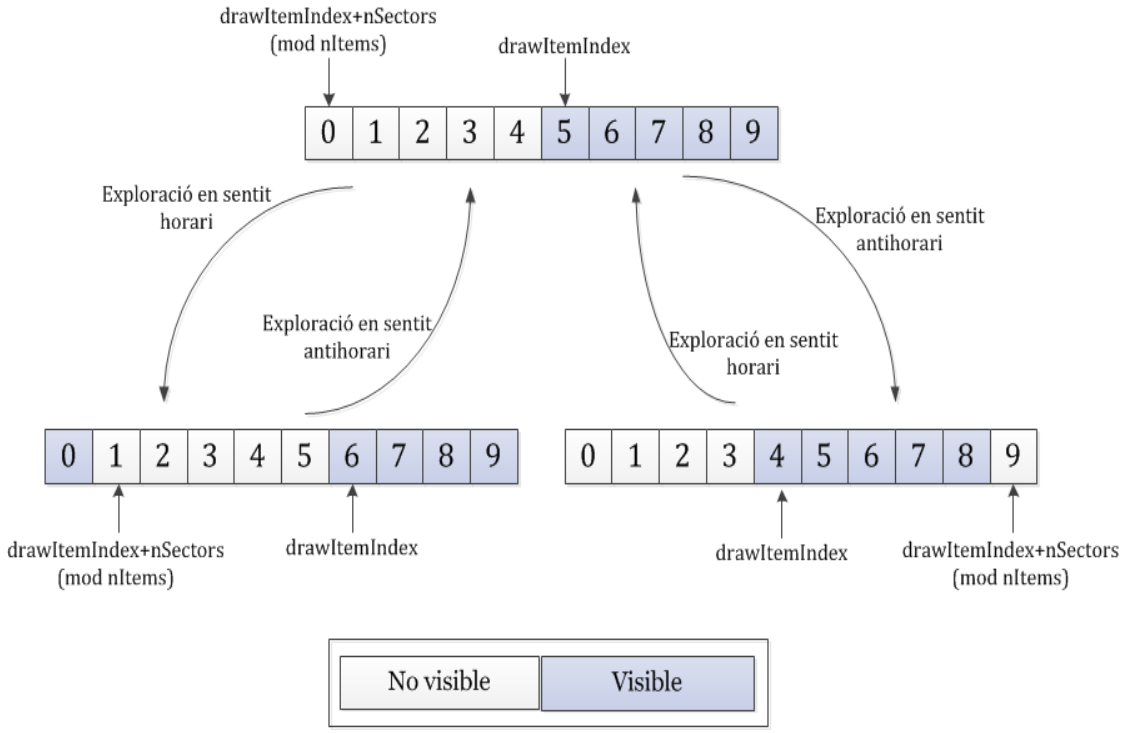


Figura 40. Exemple de representació interna dels elements dibuixats davant d'una exploració.

Per aconseguir l'efecte visual que es mostra en la Figura 39 d'oclusió dels elements, s'ha hagut de combinar el codi amb un *pixel shader* escrit en el llenguatge HLSL [27] (concretament, la versió 2.0[2] del *pixel shader*), que és tractat com un efecte d'XNA. Bàsicament, l'animació consisteix en rotar els elements sobre l'àrea circular del menú (així com les línies delimitadores dels sectors) i, quan la imatge d'un ítem "entre" en la *bridge mark*, llavors s'activa l'efecte del *shader* per descartar[3] els píxels de la imatge que ha entrat en aquesta regió, evitant que la GPU els dibuixe. Per aquesta raó, per tal de permetre al *shader* realitzar aquestes operacions, és necessari que tots els ítems siguen tractats com a imatges, independentment de si eren textos o no. D'aquesta manera s'aconsegueix ocultar els ítems que s'amaguen sota la *bridge mark*. Per als ítems que hi ixen, s'usa també el mateix *shader*, i el que es fa és, en el moment en què un element comença a amagar-se, el que ha d'eixir es col·loca baix completament de la *bridge mark*, i se li aplica la rotació corresponent com a la resta d'ítems. Així, en estar sotmès a l'efecte del *pixel shader*, els seus píxels només es dibuixaran a mesura que vagen sortint d'aqueta regió. Les funcions utilitzades al fitxer HLSL es descriuen a continuació.

El *shader* conté dues funcions principals: *PixelShaderForClockwiseItems* i *PixelShaderForCounterclockwiseItems*. La primera s'aplica a aquells elements que eixirien de la *bridge mark* en cas d'explorar en sentit horari i aquells que

[2] La raó d'emprar aquesta versió i no una posterior amb noves funcions és deguda a què la 2.0 és la versió més nova que suporta la targeta gràfica de la MS Surface 1.0.
[3] Tot i que es parla de descartar un píxel, la versió 2.0 de HLSL no permet realitzar aquesta acció. Per això, el que es fa en realitat és fer-lo totalment transparent de manera que no es veja i l'efecte resultant siga el mateix. La versió 2.a ja permet descartar píxels.

s'amaguen per la mateixa regió, és a dir, per la dreta de la *bridge mark*. La segona funció s'aplica per als ítems situats a l'esquerra. La resta d'elements, com que és impossible que s'oculten perquè no estan en sectors adjacents a l'especial, no es sotmeten a l'acció del *shader* i es dibuixen amb tots els seus píxels. L'Algorisme 4 i l'Algorisme 5 mostren la implementació d'aquestes dues funcions.

Funció PixelShaderForClockwiseItems
Entrada
- textureCoordinates: Coordenades de la imatge que representa a l'element a retallar.

Eixida
- Color resultant del píxel processat. Si cal descartar-lo, es fa transparent posant la component *alpha* a 0; si no, es manté el color original.

Variables auxiliars
- coloredTextureSampler: Permet el dibuixat de les textures.

Algorisme
```
output = tex2D(coloredTextureSampler, textureCoordinates);
pixelPos = GetPixelPosition(textureCoordinates);
if (IsPixelCrossingRightSpecialRegionLine(pixelPos))
 output.a = 0;
return output;
```

Algorisme 4. Algorisme de la funció *PixelShaderForClockwiseItems* del *pixel shader* per descartar píxels.

Funció PixelShaderForCounterclockwiseItems
Entrada
- textureCoordinates: Coordenades de la imatge que representa a l'element a retallar.

Eixida
- Color resultant del píxel processat. Si cal descartar-lo, es fa transparent posant la component *alpha* a 0; si no, es manté el color original.

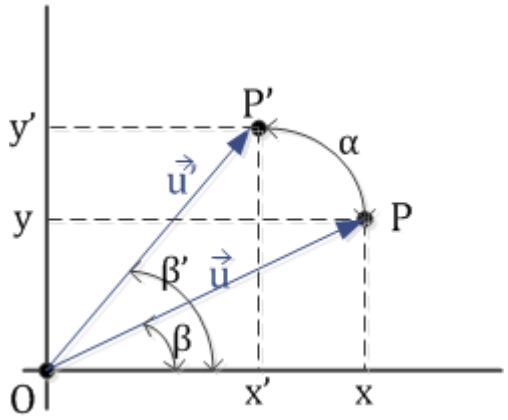
Variables auxiliars
- coloredTextureSampler: Permet el dibuixat de les textures.

Algorisme
```
output = tex2D(coloredTextureSampler, textureCoordinates);
pixelPos = GetPixelPosition(textureCoordinates);
if (IsPixelCrossingLeftSpecialRegionLine(pixelPos))
 output.a = 0;
return output;
```

Algorisme 5. Algorisme de la funció *PixelShaderForCounterclockwiseItems* del *pixel shader* per descartar píxels.

L'única informació que es pot obtindre sobre el píxel referent a la seua posició són les seues coordenades dins de la textura (*textureCoordinates*), que van de 0 a 1 en X i en Y partint del cantó superior esquerre de la imatge. Per tant, és necessari passar-li certs paràmetres al *shader* des del programa principal. Aquests es descriuen a l'Algorisme 6, on s'explica la implementació de la funció

GetPixelPosition. En aquesta es fa ús d'una altra funció, *RotateAboutOrigin*. L'objectiu d'aquesta és, donat dos punts P i O, calcular quina serà la posició P' de P després de rotar-lo α radians al voltant de O. A la Figura 41 es mostra com es realitza aquest càlcul.



Passos:
1. Obtindre $\vec{u} = P - O = (x, y)$
2. $\vec{u}$ es pot vore com:
$$\vec{u} = \begin{cases} x = |\vec{u}| \cos(\beta) \\ y = |\vec{u}| \sin(\beta) \end{cases}$$
3. Com $|\vec{u}| = |\vec{u}'|$, $\vec{u}'$ es pot vore com:
$$\vec{u}' = \begin{cases} x' = |\vec{u}| \cos(\beta') \\ y' = |\vec{u}| \sin(\beta') \end{cases}$$
4. $\beta = \text{atan}\left(\frac{y}{x}\right)$. Per tant, es pot calcular $\beta' = \alpha + \beta$, i ja es té $\vec{u}'$.
5. Obtindre $P' = \vec{u}' + O$

De forma simplificada, es pot obtindre:
$$\vec{u}' = \begin{cases} x' = x \cos(\alpha) - y \sin(\alpha) \\ y' = y \cos(\alpha) + x \sin(\alpha) \end{cases}$$
$$P' = \vec{u}' + O$$

Figura 41. *RotateAboutOrigin*: Rotació d'un punt P al voltant d'un altre O.

Funció GetPixelPosition
Entrada
- textureCoordinates: Coordenades de la imatge que representa a l'element a retallar.

Eixida
- Posició en coordenades del món del píxel representat per les coordenades textureCoordinates.

Variables auxiliars
- topLeftCorner: Posició del cantó superior-esquerre de la textura en coordenades del món.
- width: Amplada de la textura.
- height: Altura de la textura.
- center: Posició central de la textura en coordenades del món.
- rotation: Rotació de la textura en radians. S'usa conjuntament amb center per determinar la posició del píxel quan la textura està rotada.

Algorisme
```
ppos.x = topLeftCorner.x + textureCoordinates.x + width;
ppos.y = topLeftCorner.y + textureCoordinates.y + height;
ppos = RotateAboutOrigin(ppos, center, rotation);
return ppos;
```

Algorisme 6. Algorisme de la funció *GetPixelPosition* del *pixel shader* per obtindre la posició d'un píxel.

A continuació entren en joc les funcions *IsPixelCrossingRightSpecialRegionLine* i *IsPixelCrossingLeftSpecialRegionLine*, explicades a l'Algorisme 7 i a l'Algorisme 8, respectivament. La funció d'aquestes dues és determinar si, real-

ment, el píxel està dins de la regió de la *bridge mark*. Com podria donar-se el cas de tindre un element més gran que aquesta regió, si només s'oculta el que està "baix" d'esta podria passar que aquest ítem en qüestió se n'eixira per l'altre costat del que ha entrat, quan hauria de quedar-se ocult. Per això, en aquestes funcions, es fa que la regió d'oclusió siga més gran que el sector de la *bridge mark*, concretament, es consideren els sectors adjacents: Per a l'ítem de la part dreta es considera el sector adjacent de l'esquerra, i viceversa.

Cal destacar que, de nou, necessitarem informació que no podem obtenir de la targeta gràfica i, per tant, se li haurà de passar per programa. És el cas dels dos angles que delimiten la regió especial respecte al sistema de coordenades local del menú: *cwSpecialAngle* i *ccwSpecialAngle*. Aquests angles es calculen emprant el mètode *WrapAngle* de la classe *MathHelper* d'XNA, que redueix un angle a l'interval $[-\pi, \pi]$. En les funcions anteriors, és necessari conèixer l'angle en el que es troba el píxel. Per a fer-ho, es fa ús de la funció *GetAngle* (vore Algorisme 9). La funció arctangent de HLSL torna, però, l'angle en $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. Llavors, s'ha de transformar aquest valor per a què estiga en el primer interval.

Funció IsPixelCrossingRightSpecialRegionLine
Entrada
- pixelPos: Coordenades de la posició del píxel.

Eixida
- True si el píxel està dins de la regió especial; false en cas contrari.

Variables auxiliars
- cwSpecialAngle: Angle que forma la línia del costat dret de la regió especial.
- ccwSpecialAngle: Angle que forma la línia del costat esquerre de la regió especial.
- regionAngle: Angle que formen les diferents seccions dels ítems del menú.

Algorisme
```
pixelAngle = GetAngle(pixelPos);
ccwAngle = ccwSpecialAngle + regionAngle;
diff = ccwAngle - π;
if (diff > 0)
 ccwAngle = diff - π;
in = IsAngleInsideRegion(pixelAngle, cwSpecialAngle, ccwAngle);
return in;
```

Algorisme 7. Algorisme de la funció *IsPixelCrossingRightSpecialRegionLine* del *pixel shader* per comprovar si cal descartar un píxel.

Tal i com es pot observar a l'Algorisme 7 i a l'Algorisme 8, per determinar si un píxel està dins de la regió de la *bridge mark* es delega en una funció *IsAngleInsideRegion*, que rep com a entrada l'angle del píxel (*pixelAngle*) i els dos angles delimitadors de la regió d'oclusió (*cwAngle* i *ccwAngle*), tots entre $-\pi$ i π. La decisió que pren aquesta funció es basa en mirar si *pixelAngle* està comprès entre els altres dos valors. No obstant, donat l'interval de representació,

aquesta decisió és un tant complicada, i es necessita actuar de forma distinta segons el quadrant on es troben els angles. A mode de recordatori, la Figura 42 mostra els distints quadrants d'un cercle. La Taula 2 recull les condicions que s'han d'acomplir per a què *pixelAngle* estiga comprès entre els altres dos angles delimitadors depenent del quadrant en què es troben aquests últims.

Funció IsPixelCrossingLeftSpecialRegionLine
Entrada
- pixelPos: Coordenades de la posició del píxel.

Eixida
- True si el píxel està dins de la regió especial; false en cas contrari.

Variables auxiliars
- cwSpecialAngle: Angle que forma la línia del costat dret de la regió especial.
- ccwSpecialAngle: Angle que forma la línia del costat esquerre de la regió especial.
- regionAngle: Angle que formen les diferents seccions dels ítems del menú.

Algorisme
```
pixelAngle = GetAngle(pixelPos);
cwAngle = cwSpecialAngle + regionAngle;
diff = cwAngle + π;
if (diff < 0)
 cwAngle = diff + π;
in = IsAngleInsideRegion(pixelAngle, cwAngle, ccwSpecialAngle);
return in;
```

Algorisme 8. Algorisme de la funció *IsPixelCrossingLeftSpecialRegionLine* del *pixel shader* per comprovar si cal descartar un píxel.

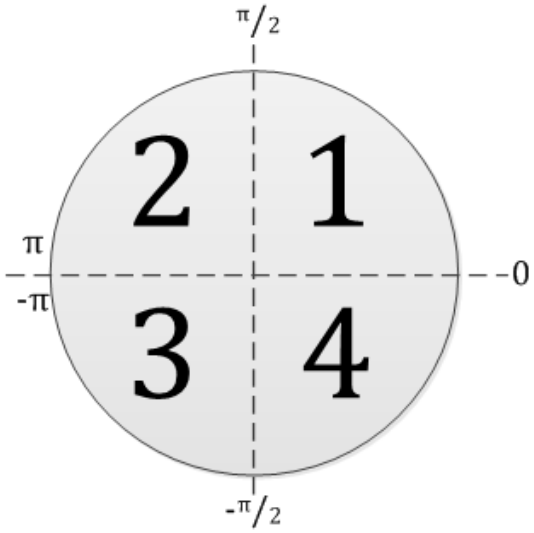


Figura 42. Quadrants d'un cercle.

Funció GetAngle
Entrada
- pos: Coordenades de la posició de l'ítem.

Eixida
- Angle en l'interval $[-\pi,\pi]$ que forma pos dins del sistema de coordenades local del menú.

Variables auxiliars
- origin: Punt en coordenades del món que indica el centre del menú.

Algorisme
```
xAboutOrigin = pos.x – origin.x;
yAboutOrigin = pos.y – origin.y;
// Es canvia el signe per a passar a coordenades cartesianes,
// on l'eix Y creix cap amunt, ja que en coordenades del món la
// Y creix cap avall.
yAboutOrigin *= -1;
angle = atan(yAboutOrigin / xAboutOrigin);

// Depenent del quadrant on es trobe el punt, cal ajustar
// l'angle o no per a què estiga en l'interval desitjat.
if (xAboutOrigin < 0) {
 if (yAboutOrigin >= 0)  // quadrant 2
  return angle + π;
 else                    // quadrant 3
  return angle - π;
}
else                     // quadrant 1 o 4
 return angle;
```

Algorisme 9. Algorisme de la funció *GetAngle* del *pixel shader* per calcular l'angle d'un píxel dins del sistema de coordenades local d'un TangiWheel.

cwAngle	*ccwAngle*	Condició
1	1,2	$pixelAngle \geq cwAngle \wedge pixelAngle \leq ccwAngle$
1	3,4	$pixelAngle \geq cwAngle \vee pixelAngle \leq ccwAngle$
2	2	$pixelAngle \geq cwAngle \wedge pixelAngle \leq ccwAngle$
2	1,3,4	$pixelAngle \geq cwAngle \vee pixelAngle \leq ccwAngle$
3	1,2,3,4	$pixelAngle \geq cwAngle \wedge pixelAngle \leq ccwAngle$
4	3	$pixelAngle \geq cwAngle \vee pixelAngle \leq ccwAngle$
4	1,2,4	$pixelAngle \geq cwAngle \wedge pixelAngle \leq ccwAngle$

Taula 2. Condicions que comprova la funció *IsAngleInsideRegion* per a què un angle *pixelAngle* estiga comprès entre altres dos (*cwAngle* i *ccwAngle*) depenent del quadrant en el que es troben.

2.4 Marca de Llista

A l'inici de la secció 2 es comentava l'existència d'una marca situada a la perifèria del control que indica, dins de la llista d'elements que s'estan explorant, la posició relativa que ocupen aquells que són visibles en el moment actual (concretament, la posició del primer que es mostra). S'anomenà a aquesta, marca de

llista (vore Figura 29). A continuació s'explicarà el funcionament de la mateixa mitjançant un exemple.

Figura 43. Funcionament de la marca de llista amb un exemple amb dos elements en la col·lecció i només un element visible.

Suposem un cas com el de la Figura 43: Tenim un TangiWheel que representa una col·lecció de dos elements (els números "1" i "2") on el màxim nombre d'elements visibles alhora és 1. L'objectiu de la marca de llista és mostrar la posició relativa del primer ítem mostrat dins de la col·lecció, de forma que, si el primer visible coincideix amb el primer de la llista, llavors la marca estarà dalt de la *bridge mark* a una posició $P_0 = (R, \frac{\pi}{2})$ en coordenades polars relatives al sistema de coordenades centrat al centre del menú, on R és el radi del mateix (Figura 43-esquerra). Al realitzar l'exploració completa de tots els elements i tornar al primer, la marca hauria d'estar en la posició P_0 inicial i, al realitzar els gestos necessaris per mostrar el següent (i últim) element, la marca estarà baix del tot, a la posició $P_1 = \left(R, -\frac{\pi}{2}\right) = (R, \frac{3\pi}{2})$, com es mostra a la Figura 43-dreta.

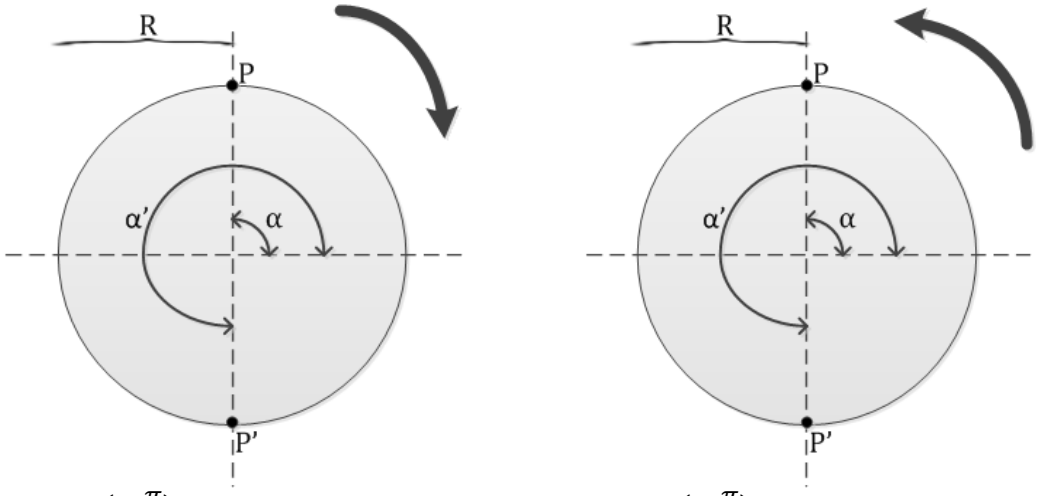
La distància al centre sempre serà R, ja que, siga quina siga la posició de la marca, sempre està sobre el perímetre. Així que l'únic que variarà serà l'angle, al qui anomenarem $\alpha \in [0,2\pi[$ i que, inicialment, valdrà $\frac{\pi}{2}$.

Siga $P = (R, \alpha)$ una posició qualsevol de la marca. Davant un gest de rotació que implica una exploració, cada vegada que un ítem nou es mostre s'ha d'actualitzar la posició modificant l'angle. Com realitzar una exploració de tots els $nItems$ elements implica una rotació de 2π radians, per a cada nou element mostrat α sofrirà un desplaçament de $\frac{2\pi}{nItems}$ radians. El sentit d'aquest desplaçament dependrà del sentit del gest realitzat (horari o antihorari), llavors, suposarà realitzar un increment sobre α en sentit antihorari, ja que és el sentit creixent dels angles, o un decrement en cas d'explorar en sentit horari, car és el sentit decreixent dels angles. Per tant, el càlcul de la posició després de mostrar un element nou, P', seria la següent:

$$P' = (R, \alpha')$$

$$\alpha' = \left(\alpha + \delta \frac{2\pi}{nItems}\right) \bmod 2\pi$$
$$\delta = \begin{cases} -1, & \text{si rotació horària} \\ 1, & \text{si rotació antihorària} \end{cases}$$

A la Figura 44 es mostra gràficament el càlcul anterior per a l'exemple plantejat anteriorment.



$$P = \left(R, \frac{\pi}{2}\right)$$
$$P' = \left(R, \left(\frac{\pi}{2} - \frac{2\pi}{2}\right) \bmod 2\pi\right) =$$
$$= \left(R, -\frac{\pi}{2} \bmod 2\pi\right) =$$
$$= \left(R, \frac{3\pi}{2}\right)$$

$$P = \left(R, \frac{\pi}{2}\right)$$
$$P' = \left(R, \left(\frac{\pi}{2} + \frac{2\pi}{2}\right) \bmod 2\pi\right) =$$
$$= \left(R, \frac{3\pi}{2} \bmod 2\pi\right) =$$
$$= \left(R, \frac{3\pi}{2}\right)$$

Figura 44. Càlcul de P' per a una exploració en sentit horari (esquerra) i antihorari (dreta), en el cas de tindre dos elements en la col·lecció i només un element visible.

No obstant, l'efecte visual que es produiria així és molt brusc, ja que el canvi de posició de la marca es realitzaria de forma discreta (i.e. només cada vegada que un element nou fóra completament visible), i l'efecte resultant seria el de la marca "pegant bots". Llavors, per fer-ho més acord a l'efecte d'exploració "suau" explicat en la secció 2.3, es desitja que el canvi de posició siga continu. La forma d'aconseguir-ho és emprar les velocitats d'exploració. Així, de forma definitiva, el càlcul de P' es realitzarà cada vegada que l'usuari realitze un gest de rotació estant el control en mode d'exploració, i el càlcul es vorà afectat per esta velocitat, v, fent que es necessiten v angles reals rotats per a què un ítem nou aparega completament i, com a conseqüència, per aconseguir l'increment o el decrement de $\frac{2\pi}{nItems}$ radians comentats anteriorment:

$$P' = (R, \alpha')$$
$$\alpha' = \left(\alpha + \delta \frac{\frac{2\pi}{nItems}}{v}\right) \bmod 2\pi$$

$$\delta = \begin{cases} -1, & \text{si rotació horària} \\ 1, & \text{si rotació antihorària} \end{cases}$$
$$\nu = \begin{cases} 13, & \text{si el TangiWheel té tangible associat} \\ 5, & \text{si no té tangigle associat} \end{cases}$$

2.5 Marca de Pròxim Element

Al realitzar l'exploració dels elements, els visibles realitzen una rotació sobre la regió circular (vore secció 2.3) però, a més, s'ha afegit altre indicador de què s'està fent una exploració i que, a més, indica quant falta per rotar per a què un element nou siga totalment mostrat: la marca de pròxim element (vore Figura 29).

Aquesta marca realitzarà una rotació mentre un element estiga sortint (i, per tant, altre amagant-se) fins completar a un gir complet de 2π radians, tornant a situar-se just baix de la *bridge mark*, en el moment en què aquest element és totalment visible. Tal i com s'ha comentat en la secció 0, existeix una velocitat d'exploració ν que indica el nombre d'angles que s'ha de rotar (amb el pom o el dit, segons el mode d'interacció escollit) per aconseguir aquesta rotació completa de la marca de pròxim element. Per tant, siga γ la rotació actual de la marca (que, inicialment, val $\frac{\pi}{2}$ perquè està situada a sota de la *bridge mark*), el càlcul de la rotació de la marca γ' en resposta a un sol gest de rotació de l'usuari es realitza com segueix:

$$\gamma' = \left(\gamma + \delta\frac{2\pi}{\nu}\right) \bmod 2\pi$$
$$\delta = \begin{cases} -1, & \text{si rotació horària} \\ 1, & \text{si rotació antihorària} \end{cases}$$
$$\nu = \begin{cases} 13, & \text{si el TangiWheel té tangible associat} \\ 5, & \text{si no té tangigle associat} \end{cases}$$

2.6 Regió Central

Si un TangiWheel té associat un tangible, el canvi de posició del primer és controlat pel segon. A més, quan el pom és retirat de la superfície, el control desapareix, tornant a mostrar-se si l'usuari el torna a posar sobre la taula. Això és perquè es crea un vincle entre el tangible i el TangiWheel. Per contra, si no s'està fent ús dels tangibles, es diu que el menú és virtual i ha de ser controlat només amb els dits. Òbviament, la vinculació anterior no es possible, i si es vol ocultar el control s'ha d'eliminar per complet. Per realitzar aquesta acció, així com per moure el menú, s'empra la regió central (vore Figura 29), que realitza una operació o l'altra depenent del gest que realitze l'usuari.

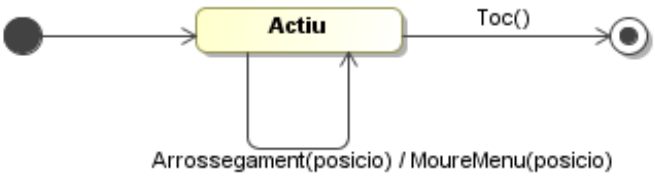


Figura 45. Comportament del TangiWheel virtual davant d'interaccions de l'usuari sobre la marca central.

La Figura 45 mostra com es comporta aquest control davant de les interaccions de l'usuari sobre la marca central quan es troba en mode virtual. Bàsicament, realitzar-hi un toc implica la destrucció de l'objecte mentre que un arrossegament implica realitzar una translació de tot el control a la posició de destí del dit. A la Figura 38 s'explica com distingir entre aquests dos gestos. S'hi fan ús dels manejadors d'esdeveniments *OnContactDown*, *OnContactChanged* i *OnContactLeave* explicats al Capítol 2.

3 Subcol·leccions

TangiWheel permet la definició d'un nombre virtualment il·limitat de col·leccions jeràrquiques, de manera que una llista d'elements pot contindre diverses subcol·leccions niades, on cadascuna d'aquestes últimes es representa com un ítem del pare que, en ser seleccionat, crea una altra instància TangiWheel amb la col·lecció filla. Açò crea una relació entre dos instàncies concretes, i aquesta és representada gràficament per una cadena que uneix els dos controls quan ambdós són visibles. D'aquesta manera, si dos menús amb aquesta relació jeràrquica estan en llocs diferents de la superfície, hi ha una representació visual de la seua unió.

Una subcol·lecció és representada com un menú a banda per diverses raons. Primer, és una forma senzilla de suportar la replicació de col·leccions i instanciació flexible mitjançant l'ús de tangibles (i.e. associació directa), fet que permet a qualsevol element (ja siga una altra col·lecció o un element simple) ser associat a tangibles per a la seua manipulació directa. En TangiWheel, un tangible buit (no associat a cap control) pot ser col·locat sobre un ítem en qualsevol moment. En aquest moment, el tangible queda associat a aquest element, i pot contindre o bé una llista d'elements (representada com un altre TangiWheel, per exemple) o un sol element (una fulla de la jerarquia, si veiem aquesta com un arbre). Per desvincular un tangible contenidor, es pot realitzar un moviment de zig-zag sobre la superfície amb aquest o, simplement, col·locar-lo sobre altre element (llavors, el contingut del contenidor se sobreescriu). Aquest mecanisme proporciona accés directe a subcol·leccions que estan a un nivell molt profund en una jerarquia de menús i redueix el temps de cerca quan els ítems han de ser visitats de forma repetida, ja que no cal explorar totes les col·leccions intermèdies fins arribar a l'objectiu. Aquesta mateixa associació pot aconseguir-se situant un

tangible contenidor en la regió central d'un TangiWheel virtual (i.e. que no té ja un tangible associat perquè ha sigut mostrat com a resultat d'interaccions purament dactilars). El fet de representar col·leccions i subcol·leccions de la mateixa manera fa que el sistema d'associació anteriorment descrit siga homogeni des del punt de vista de les accions que un usuari ha de dur a terme (simplement col·locar un tangible contenidor en la regió central d'una instància TangiWheel).

Altra raó per obrir diverses col·leccions niades en diferents instàncies TangiWheel és que es manté el mateix procés d'exploració de col·leccions jeràrquiques i, per tant, les interaccions permeses per a la col·lecció "arrel" (tornant a la metàfora de l'arbre) són les mateixes que per a les subcol·leccions. A més a més, en situacions on es tinga un gran nombre de menús en la superfície, els usuaris podran controlar més fàcilment l'orientació, la posició i la visibilitat de les subcol·leccions representades.

Aquesta característica proporciona flexibilitat als usuaris per decidir com es representen les subcol·leccions en l'espai disponible sobre la taula. Un disseny alternatiu al proposat en aquest treball per gestionar jerarquies niades seria expandir la col·lecció filla entorn a l'element seleccionat per obtindre una distribució més compacta. No obstant, aquest enfocament entraria en conflicte amb l'acompliment d'altres característiques. Per exemple, s'haurien de canviar les tècniques d'interacció per a la col·lecció arrel i les diferents filles, afectant així a la similitud d'interaccions entre les diferents col·leccions. Si es dóna molta importància a la compactació en un escenari concret, llavors es pot considerar una altra alternativa en la que, mantenint una sola instància, els elements hi canvien dinàmicament depenent de la col·lecció seleccionada. D'aquesta manera no s'afecta la similitud d'interaccions. Una futura versió d'aquest control serà sensible al context i serà capaç de decidir dinàmicament si desplegar un nou menú per a representar els elements de la subcol·lecció seleccionada o bé utilitzar el menú pare per fer-ho, evitant la creació d'un element gràfic nou que ocupe més espai.

3.1 Representació Visual de la Jerarquia de Menús

Per representar gràficament la unió entre dos TangiWheels que tenen una relació jeràrquica de pare i fill, es dibuixa una "cadena" entre ambdós. Aquesta pot ser sòlida i/o tindre una sèrie de punts animats que es mouen en la direcció del pare al fill (vore Figura 47). La forma que té aquesta cadena és, per defecte, la d'una funció sinusoïdal, però es pot canviar implementant la funció desitjada i passant-ne un punter (mitjançant delegats) a la classe *ChainGen* (vore la seua estructura a la Figura 46), que és la que s'encarrega del dibuixat de la cadena. Aquesta funció, $chainFunction(x)$, ha d'estar definida, però, per a valors positius de x. A més, cal tindre en compte que un sol menú pare només pot tindre

un tipus de cadena (i.e. amb una sola funció). La Figura 48 mostra, a banda de l'estàndard, altres dues formes possibles que pot prendre la cadena:

- Dalt, un moviment sinusoïdal, que és el que es troba per defecte.
$$chainFunction(x) = \sin(x)$$
- Al centre, un moviment constant.
$$chainFunction(x) = 0$$
- Baix, un moviment oscil·latori amb atenuació.
$$chainFunction(x) = \frac{5 \cdot \sin(x)}{e^{0.03x}}$$

Cal destacar que els punts d'unió dels dos controls relacionats es recalculen a cada moviment de qualsevol dels dos, a fi d'aconseguir que la cadena tinga la mínima longitud possible (per a no sobrecarregar en excés la superfície). Aquests punts corresponen als punts d'intersecció de la recta que uneix els centres dels menús amb el perímetre del cercle de cadascun, si estan ambdós no minimitzats. Si algun control està minimitzat, el punt d'unió corresponent a la seua part és el seu mateix centre, tal i com es mostra a la Figura 49 o en la Figura 47 centre-dreta.

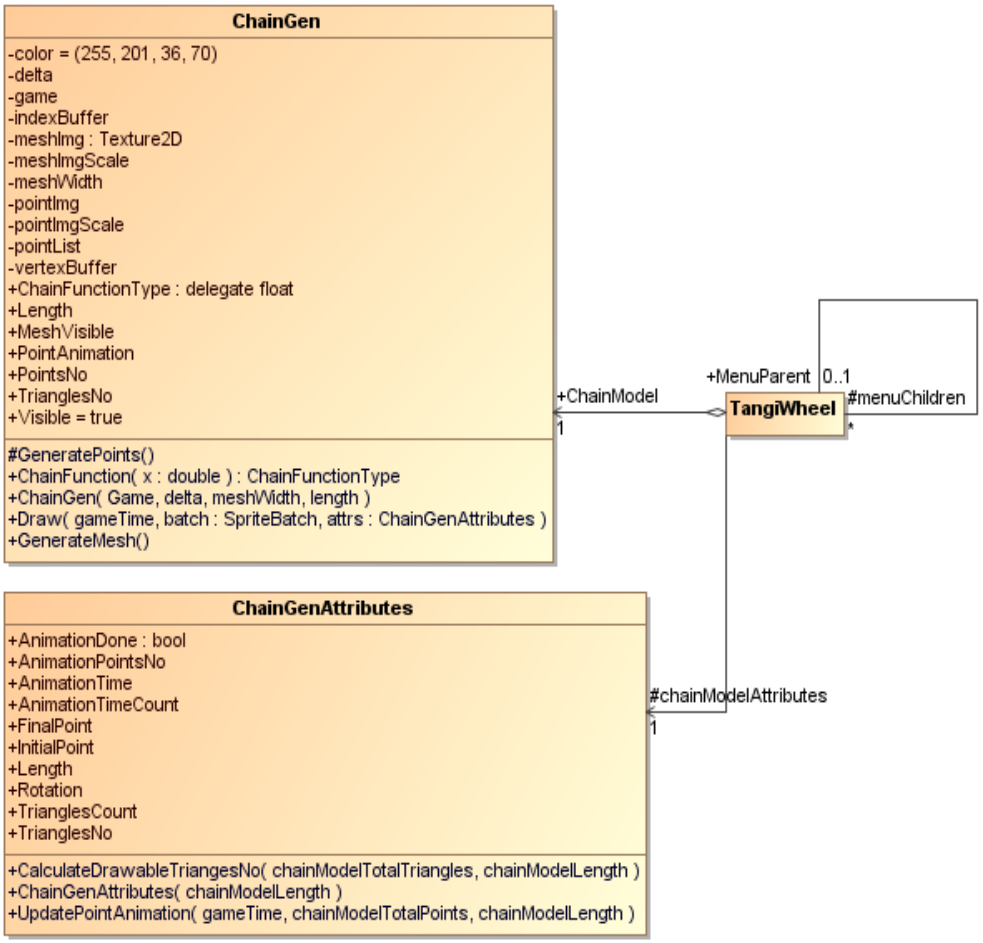


Figura 46. Diagrama de classes de la cadena.

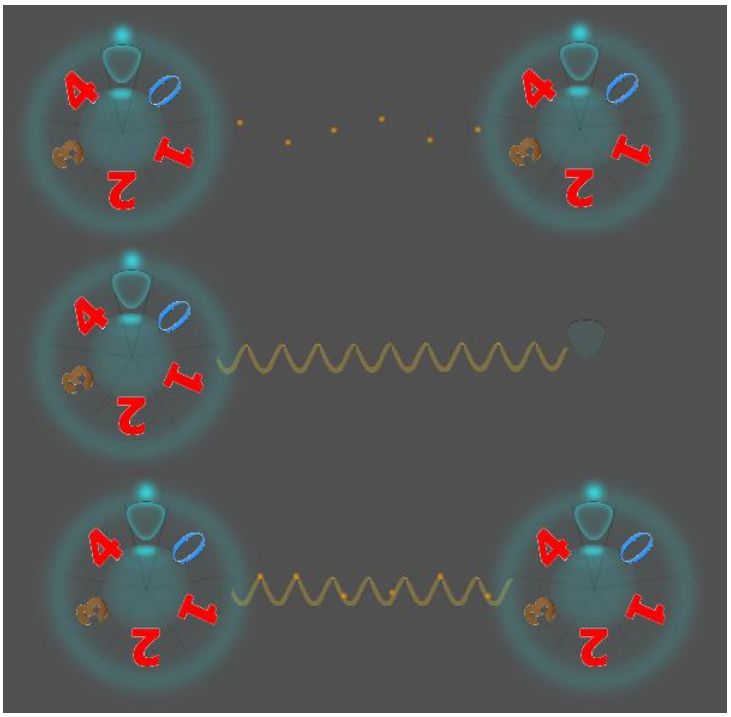


Figura 47. Representacions possibles de la cadena: Amb punts només (dalt), sòlida (centre) o sòlida amb punts (baix).

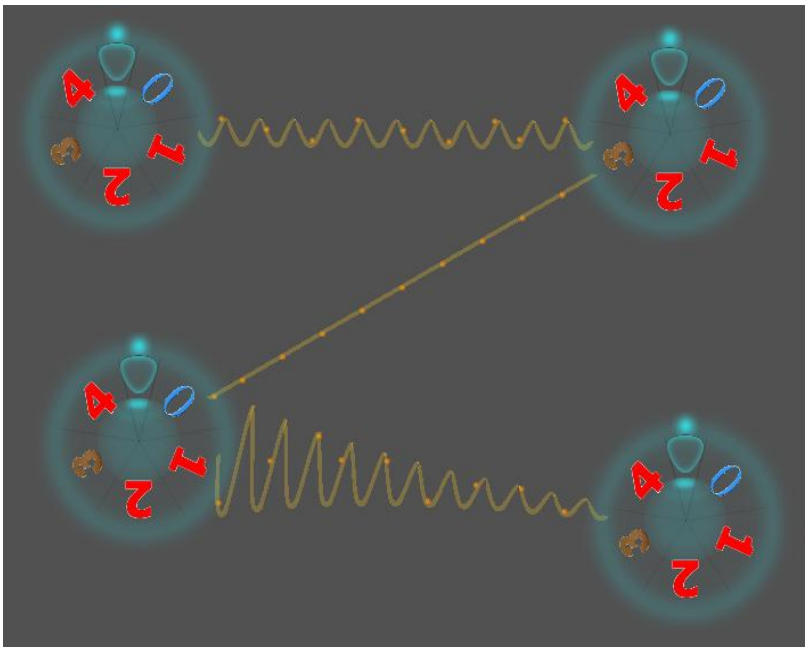


Figura 48. Diferents formes de la cadena variant les funcions que la defineixen.

Siguen O_1 i O_2 els punts centrals dels controls pare i fill, respectivament; siguen P_1 i P_2 els punts d'unió d'aquests (o, el que és el mateix, els punts d'inici i de fi, respectivament, de la cadena); i siguen R_1 i R_2 els radis d'ambdós, respectivament. El càlcul de P_1 i P_2 es realitza de la següent manera:

$$\vec{d} = \frac{O_2 - O_1}{|O_2 - O_1|}$$

$$P_1 = \begin{cases} O_1, & \text{si pare minimitzat} \\ O_1 + R_1 \cdot \vec{d}, & \text{si pare no minimitzat} \end{cases}$$
$$P_2 = \begin{cases} O_2, & \text{si fill minimitzat} \\ O_2 - R_2 \cdot \vec{d}, & \text{si fill no minimitzat} \end{cases}$$

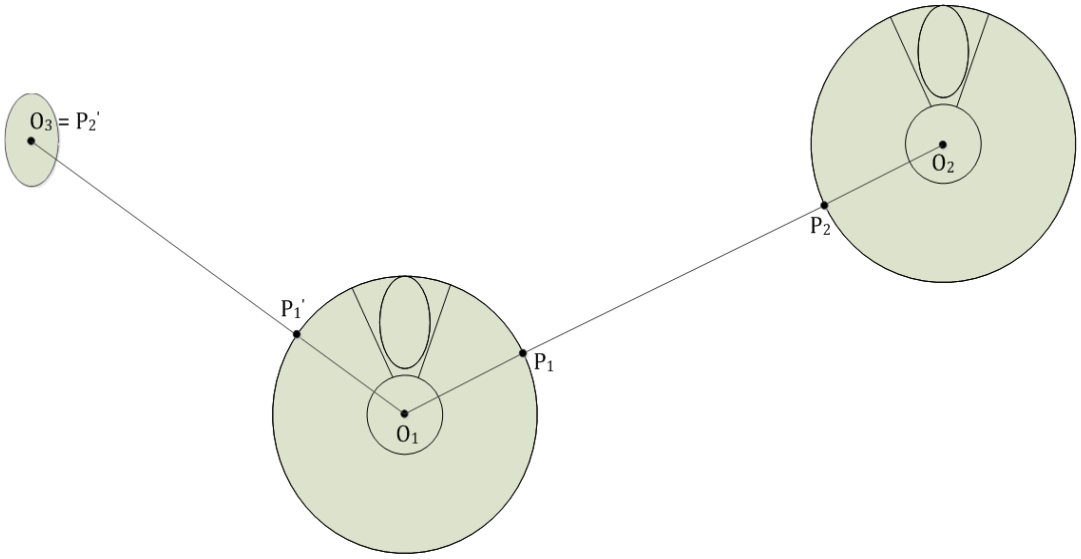


Figura 49. Càlcul dels punts d'inici P_1 i P_1' i de fi P_2 i P_2' de les cadenes que uneixen un menú pare (centre) amb dos menús fills. Un minimitzat (esquerra) i altre que no ho està (dreta).

Una vegada vist com es calculen els punts d'unió de la cadena amb els controls, només falta descriure de quina manera es genera la malla sòlida i el camí que segueixen els punts animats.

El mètode *GeneratePoints* de *ChainGen* (vore Algorisme 10) és l'encarregat de representar el "camí" de punts a partir de la funció escollida, que serà el que seguisquen els punts gràfics en una animació.

Mètode GeneratePoints
Variables auxiliars
- length: Longitud de la cadena.
- delta: Espai entre punts de la cadena. Quant més menut, més llises seran les corbes, però hi haurà més punts, fet que suposa un increment del cost computacional tant per a generar-los com per a dibuixar-los.
- pointList: Llista de punts que emmagatzema el resultat del processament d'aquest mètode.

Algorisme
```
for (x = 0; x <= length; x += delta) {
 y = chainFunction(x);
 point = (x, y, 0);
 pointList.Add(point);
}
```

Algorisme 10. Algorisme per generar els punts de la cadena en *ChainGen*.

Mètode GenerateMesh

Variables auxiliars

- MeshVisible: Propietat que indica si la malla serà visible i, per tant, si cal crear-la. Si no, només es generen els punts amb GeneratePoints.
- pointList: Llista amb els punts obtinguts en la crida al mètode GeneratePoints. Aquests serviran de base per a la creació dels triangles que compondran la malla.
- vertexBuffer: Llista que s'omplirà amb tots els vèrtexs que conformen els triangles de la malla. S'uniran dos punts de pointList amb un altre punt nou que es crearà amb una orientació de 45° respecte dels altres.
- indexBuffer: Llista de tuples de tres elements que representa, cadascuna, els tres vèrtexs que formen un triangle a ser dibuixat.
- meshWidth: Amplada que ha de tindre la malla.
- color: Color que s'assigna als vèrtexs en ser creats.

Funcions auxiliars

- vertexBuffer.Add: Afegeix un nou vèrtex a la llista. Els seus paràmetres són, per ordre, el punt a afegir, el seu color i les coordenades de la textura que se li apegarà. Aquestes coordenades són (0,0) per al cantó superior-esquerre; (0,1) per a l'inferior-esquerre; (1,0) per al superior-dret; i (1,1) per a l'inferior-dret.
- indexBuffer.Add: Afegeix un a sèrie d'índexs a la llista. Cada paràmetre és un índex a afegir.
- Normalize: Normalitza un vector obtenint el seu vector director.
- Rotate: Crea una rotació en l'eix Z (i.e. en 2D) sobre un punt.

Precondicions

- pointList ha de tindre, almenys, tres punts, que és el mínim per formar un triangle.

Algorisme

```
GeneratePoints();
if (MeshVisible) {
 // Primers tres vèrtexs
 vertexBuffer.Add(pointList[0], color, (0,0));
 vertexBuffer.Add(pointList[1], color, (0,1));
 vecDist = pointList[1] – pointList[0];
 dir = Normalize(Rotate(vecDist, π/4));
 vertexBuffer.Add(pointList[0] + dir * meshWidth, color, (1,0));

 // Resta de vèrtexs
 for (i = 1; i < pointList.Count – 1; i++) {
  vecDist = pointList[i + 1] – pointList[i];
  dir = Normalize(Rotate(vecDist, π/4));
  vertexBuffer.Add(pointList[i] + dir * meshWidth, color, (1,1));
  vertexBuffer.Add(pointList[i + 1], color, (0,0));
  if (i == pointList.Count – 1)
   vertexBuffer.Add(pointList[i + 1] + dir * meshWidth, color, (0,1));
 }

 // Index buffer
 index = 0;
```

```
 while (index < vertexBuffer.Count - 3) {
  if (index == 0) {
   indexBuffer.Add(index, index + 1, index + 3);
   indexBuffer.Add(index, index + 3, index + 2);
   index++;
  }
  else if (index == 1) {
   indexBuffer.Add(index, index + 3, index + 4);
   indexBuffer.Add(index, index + 4, index + 2);
   index += 3;
  }
  else {
   indexBuffer.Add(index, index + 2, index + 3);
   indexBuffer.Add(index, index + 3, index + 1);
   index += 2;
  }
 }
}
```

Algorisme 11. Algorisme per generar la malla de la cadena en *ChainGen*.

Així, quan el TangiWheel desitge generar la cadena, invocarà al mètode *GenerateMesh* de *ChainGen* que, al seu torn, invocarà al mètode descrit més amunt. Si, a més, la propietat *MeshVisible* està activa, es crearà la malla dibuixant triangles que uniran els punts ja creats, tal i com es pot observar a l'Algorisme 11. El procediment és, primer, omplir un *buffer* de vèrtexs amb tots els punts que compondran la malla i, a continuació, un *buffer* d'índexs, que indicarà, a l'hora de dibuixar, tuples de tres vèrtexs que compondran un triangle. A la Figura 50 es mostra un exemple de com es construeix la malla donada una seqüència de punts (en negre). Cadascun dels punts es converteix en un vèrtex (numerat segons l'ordre d'inserció en *vertexBuffer*), i en roig es mostren els vèrtexs addicionals que afegeix l'algorisme. Dins de cada triangle generat apareix un número que indica l'ordre de dibuixat segons les tuples de l'*indexBuffer*.

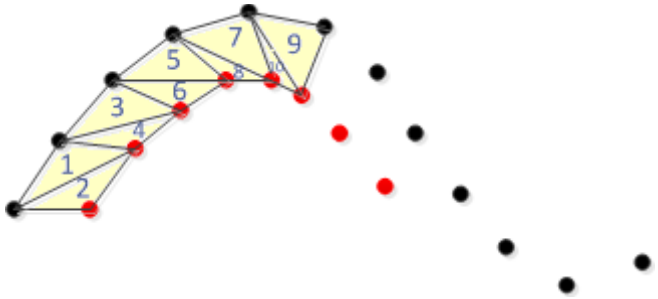


Figura 50. Generació de la malla de la cadena.

4 Mecanismes d'Interacció

En esta secció anem a descriure les solucions que s'han adoptat en TangiWheel per bregar amb la gestió de l'entrada quan es treballa amb col·leccions. S'han considerat dues característiques: modalitat i semblança.

TangiWheel suporta tres modes d'interacció per a dur a terme la seua funció principal de cerca d'ítems: Purament tangible, purament virtual o híbrid. En mode tangible, es realitza el seguiment d'un pom físic al llarg de la superfície, mantenint la rotació i la posició del menú consistents amb les de l'objecte si el control està en mode de rotació. Si està en mode d'exploració, la rotació del pom implica el canvi d'ítems visibles. Una organització en forma de pastís integra aquesta interacció més intuïtivament, car explorar realitzant una rotació pareix ser més natural en una distribució circular que en una lineal (en la que s'esperaria, més bé, un desplaçament). Si el pom abandona la taula, el control desapareix, tornant a aparèixer si el tangible torna a ser col·locat. Els ítems també poden ser seleccionats amb aquests elements físics. L'ús d'aquests per a la selecció es basa en la idea de tangibles com a *hypercards*, *phicons* i *phandlers* [37] per a contindre i transportar elements digitals.

En cas de tractar-se d'una interacció completament virtual (i.e. TangiWheels creats a partir de seleccions amb els dits que no tenen cap tangible associat), els menús responen solament davant de contactes detectats com a dits, tot i que pot tractar-se d'una entrada multitàctil en la que intervinguen diversos dits. En aquest cas, la rotació o exploració ve donada per un gest rotatori sobre la regió dels ítems visibles. Una alternativa a aquest enfocament orientat a l'ús d'un sol dit seria la proposada en [35], on s'empra una tècnica bimanual usant dos dits que descriuen trajectòries oposades per realitzar la rotació. No obstant, aquesta tècnica afectaria al nostre criteri de semblança, ja que la manipulació d'un sol tangible requereix només l'ús d'una sola mà. A més, amb la solució adoptada, on rotar significa explorar, és molt probable que, quan s'exploren grans col·leccions, es necessiten rotacions continues i constants fins aconseguir exploracions completes, que comportarien diverses rotacions de 360º. Per això, mentre que descriure cercles pot efectuar-se de forma continua sense alçar el dit de la superfície, la solució que involucra dues mans requeriria parar i tornar a començar la interacció repetidament (quan ambdós dits xocaren). Llavors, això faria aquesta alternativa menys efectiva a l'hora d'explorar col·leccions. Finalment, en aquesta modalitat, l'ocultació (i destrucció) del control es realitza polsant sobre la regió central i alçant el dit. Si, per contra, el dit no s'alça sinó que s'arrossega, es produeix la translació del menú.

La modalitat híbrida, com el seu nom indica, combina les propostes virtual i tangible de forma que l'usuari decideix en tot moment si interactuar amb els dits o bé emprar els poms per guardar algun element o altre TangiWheel a dintre. Gràcies a que eixes dues propostes són molt semblants, resulta fàcil la integració híbrida de les tècniques i, per tant, millora la flexibilitat que permet a l'usuari emprar el menú com millor s'ajuste a la seua forma d'actuar i a la tasca que ha de dur a terme.

5 Suport Multiusuari

TangiWheel està pensat per ser usat en escenaris on diversos usuaris puguen col·laborar creativament treballant conjuntament en una regió (la superfície interactiva) i compartint diverses col·leccions d'elements bàsics que, per mitjà de la seua combinació, permeten la creació d'entitats més complexes. Tots els membres de l'equip, per tant, han de tindre el mateix grau d'accés a les col·leccions i tindre una percepció similar d'aquestes, sense importar on estan localitzades en la taula. Així, el tindre un mecanisme d'abast i un control 360º [26] és una característica clau en aquest treball, a fi de fer un control reorientable i usable per diferents usuaris en un espai compartit. A més, per facilitar l'accés als usuaris des de diferents punts de la taula, es proveeixen tècniques per moure i posicionar controls en la superfície, a diferència d'altres propostes descrites anteriorment, en les quals les col·leccions i/o subcol·leccions estan ancorades a posicions predefinides.

La organització en forma de pastís evita orientar la informació cap a un sol punt fix. No obstant, si un usuari prefereix una orientació diferent del control respecte a la seua localització, pot canviar el mode d'exploració a rotació de forma que la rotació d'aquest comporte la reorientació del menú en la seua totalitat, provocant que els ítems visibles adopten una nova posició més adequada per a l'usuari en qüestió (sense perdre la seua localització relativa dins del TangiWheel). Es més, alguns dels mecanismes exposats anteriorment són importants característiques que faciliten l'accés en escenaris multiusuari. Aquests mecanismes són: posicionar el control en qualsevol punt de la superfície; associar tangibles a elements i col·leccions que poden ser esborrats i col·locats en la taula quan siga necessari; i suportar jerarquies niades i la manipulació paral·lela de col·leccions replicades.

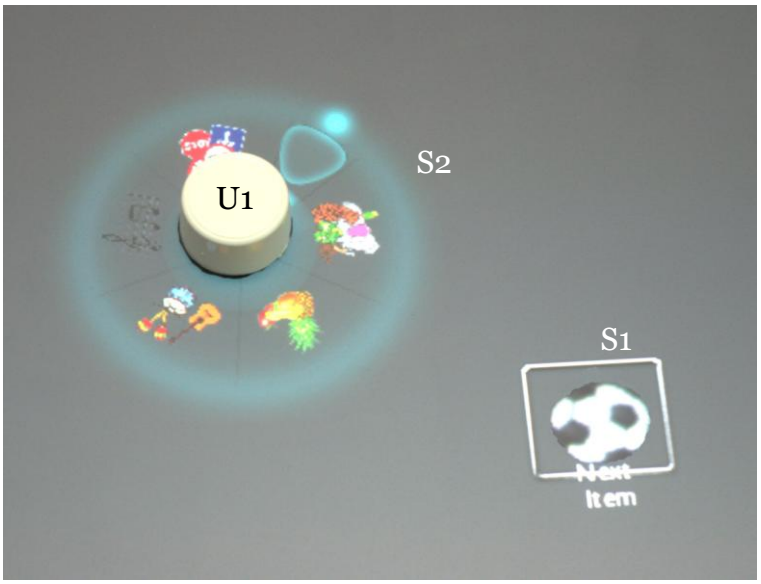


Figura 51. Interaccions 1, 2 i 3. Per a seleccionar l'ítem Pilota (*S1*), es col·loca un tangible sobre la taula (*U1*), mostrant-se un menú de categories (*S2*).

6 Exemple d'Interacció

Una vegada vistos els mecanismes d'interacció amb els controls TangiWheel (vore secció 4), la Figura 51, la Figura 52, la Figura 53 i la Figura 54 il·lustren les tècniques requerides per a seleccionar un element d'una col·lecció i dur-lo a una àrea específica en la taula. Les diferents interaccions de l'exemple estan numerades i explicades en termes d'accions d'usuari (*Ux*) i respostes del Sistema (*Sx*).

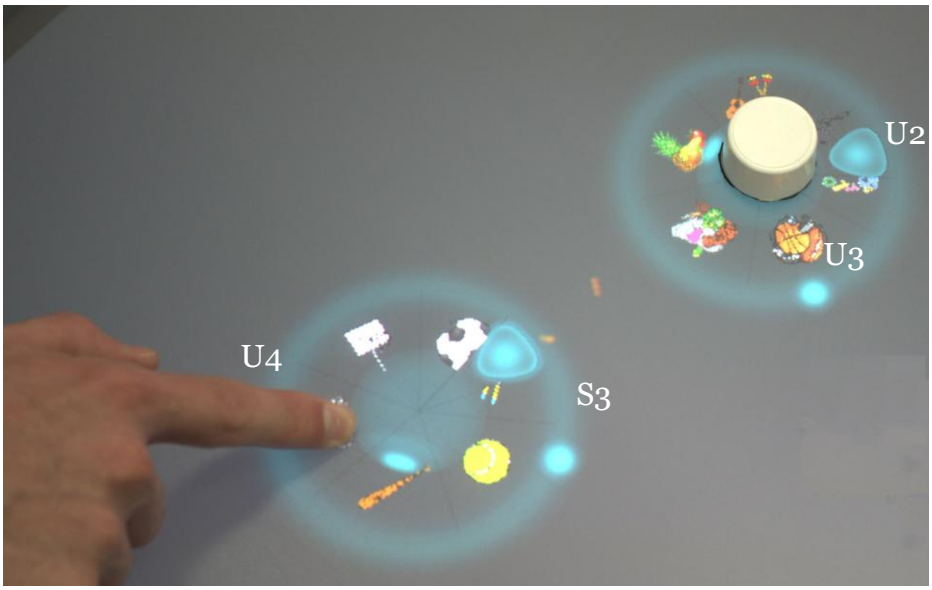


Figura 52. Interaccions 4, 5, 6 i 7. L'usuari activa el mode exploració (*U2*) i rota el tangible fins que la categoria Esports apareix. Llavors, la selecciona (*U3*). El sistema mostra el submenú d'esports (*S3*) i, fent un gest rotatiu amb els dits, s'explora en el mateix (*U4*).

1. S1: L'aplicació mostra un element objectiu a ser seleccionat representant una pilota de futbol.
2. U1: L'usuari col·loca un tangible buit sobre la superfície.
3. S2: El sistema crea un TangiWheel que conté una sèrie de categories (música, animals, etc. i entre elles la desitjada: esports) i l'associa al tangible. El control, a partir d'este moment, respon a les translacions i rotacions del pom com s'ha descrit anteriorment.
4. U2: La categoria d'esports no es mostra. Per tant, cal explorar els distints elements del menú. Per a fer-ho, l'usuari polsa sobre la bridge mark, canviant el mode de rotació (que és en el que estava) a exploració.
5. U3: Rotant el tangible es mostren els diferents ítems del control. Quan es troba el corresponent a la categoria d'esports, es toca la seua regió.
6. S3: El sistema respon a eixa acció desplegant un submenú amb els diferents elements de la categoria d'esports: una pilota de tenis, una bicicleta... El submenú va unit al seu menú pare mitjançant una cadena de punts animats que es mouen en la direcció pare-fill.
7. U4: Com s'han usat els dits per a la creació d'aquest submenú, el seu maneig es realitzarà amb aquests també: Per a la rotació/exploració, un gest circular sobre la regió dels sectors amb els elements; per al canvi de posició, col·locar

el dit sobre la regió central i arrossegar. En qualsevol moment se li podria associar un tangible tan sols posant-lo sobre la regió central. S'exploren els elements fins trobar el baló de futbol.

8. U5: Per a aquesta selecció s'elegeix emprar un pom. Es posa el mateix sobre la regió corresponent a l'element objectiu i immediatament el baló queda associat al tangible.
9. U6: Es porta el tangible amb el baló a la zona de la taula on es trobava l'objectiu a seleccionar (el pom pot alçar-se de la superfície si es desitja) i es diposita sobre la mateixa.

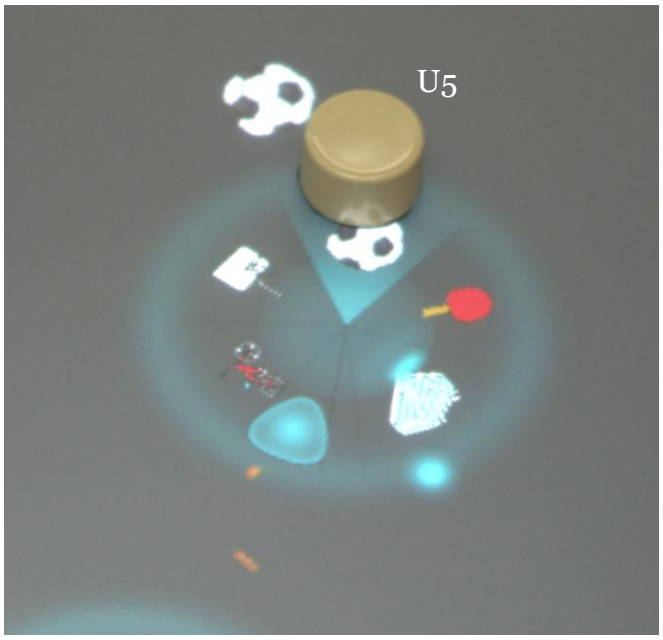


Figura 53. Interacció 8. Se selecciona l'ítem objectiu amb un tangible (*U5*).

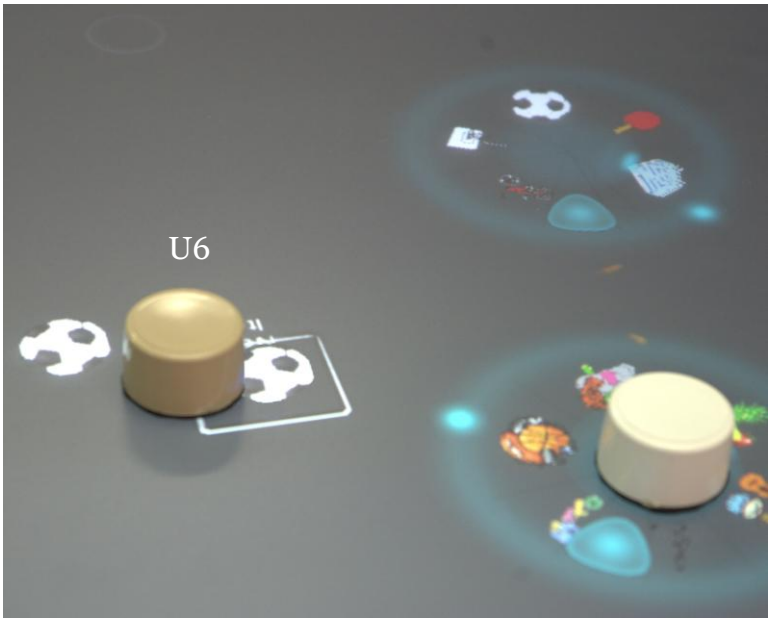


Figura 54. Interacció 9. Es porta l'element associat a la regió objectiu (*U6*).

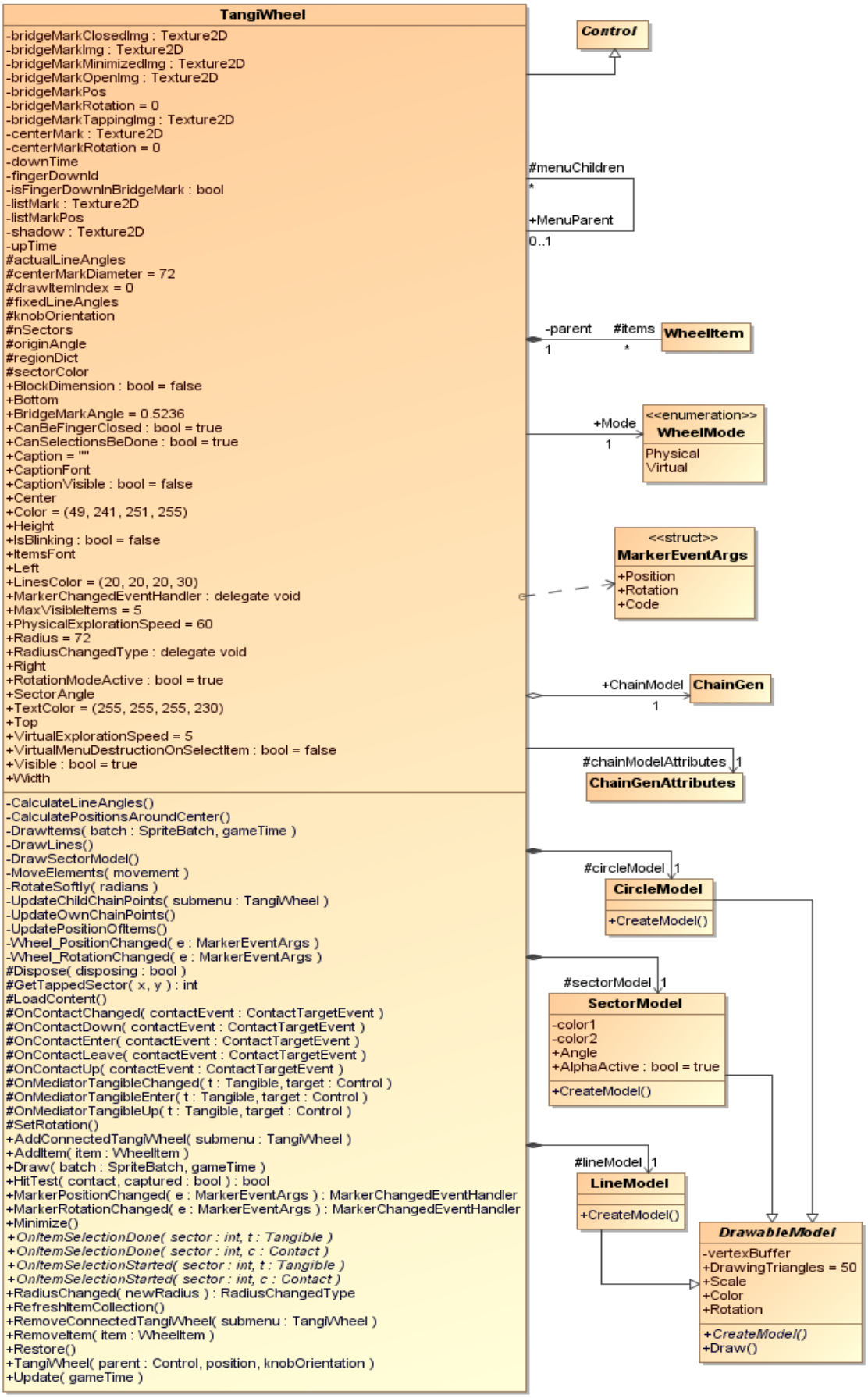


Figura 55. Diagrama de classes de *TangiWheel*.

7 Disseny de Classes

Al llarg d'aquest capítol s'ha explicat com és i com funciona TangiWheel. En les dues subseccions següents es descriurà el diagrama de classes del menú (vore Figura 55) i dels elements —anomenats WheelItems— (Figura 56), i es descriuran aquells atributs i mètodes més rellevants de forma tabulada.

7.1 Estructura del Menú

7.1.1 Atributs Privats

Nom	Descripció
bridgeMarkClosedImg	Imatge que representa la *bridge mark* en mode de rotació.
bridgeMarkImg	Variable que representa la imatge de la *bridge mark* que està sent dibuixada actualment en el menú.
bridgeMarkMinimizedImg	Imatge que representa la *bridge mark* quan el menú està minimitzat.
bridgeMarkOpenImg	Imatge que representa la *bridge mark* en mode d'exploració.
bridgeMarkPos	Posició de la *bridge mark*.
bridgeMarkRotation	Rotació de la *bridge mark*.
bridgeMarkTappingImg	Imatge que representa la *bridge mark* quan aquesta està sent polsada.
centerMark	Imatge que representa la regió central del menú.
centerMarkRotation	Rotació de la regió central.
downTime	Instant de temps en el què es realitza un *ContactDown* sobre el menú. Útil per a realitzar les animacions o per a distingir gestos com ara el de rotar o tancar el menú en mode virtual.
fingerDownId	Identificador del dit que ha fet un *ContactDown*.
isFingerDownInBridgeMark	Valor lògic que indica si un dit està sobre la *bridge mark*. Útil per poder distingir entre quan es vol canviar entre el mode de rotació i d'exploració i quan es vol minimitzar el control.
listMark	Imatge que representa la marca de llista.
listMarkPos	Posició de la marca de llista.
shadow	Imatge que representa la corona circular que envolta el TangiWheel.
upTime	Instant de temps en el què es realitza un *ContactUp* sobre el menú. Útil per realitzar les animacions o per distingir gestos com ara el de rotar o tancar el menú en mode virtual.

Taula 3. Atributs privats de *TangiWheel*.

7.1.2 Atributs Protegits

Nom	Descripció
actualLineAngles	Llista que conté, en tot moment, els angles reals i actuals de cadascuna de les línies que separen els sectors del menú. És la llista que es consulta per dibuixar les línies amb l'orientació correcta.
centerMarkDiameter	Diàmetre de la marca central.
chainModelAttributes	Informació sobre la cadena que uneix el menú actual amb el seu menú pare. Conté dades sobre el punt inicial i final de la cadena (que s'actualitza en moure els menús units) i sobre l'estat actual de l'animació de la mateixa.
circleModel	Cercle que serveix de base per al menú. Gairebé no es veu perquè, per costum, se li ha donat un color quasi transparent per qüestions d'estètica, però sí que es nota quan es tenen diverses instàncies del control amb diferents colors. A més, se li pot donar el color que es considere oportú.
drawItemIndex	Valor que indica el principi de la "finestra" d'elements visibles.
fixedLineAngles	Llista que conté els valors "estàtics" dels angles de cadascuna de les línies que delimiten els sectors. Mentre que els angles de *actualLineAngles* varien quan es realitza una exploració, aquests no canvien en cap moment mentre el nombre de sectors i l'angle de la regió de la *bridge mark* romanguen inalterats. Es prenen com a referència per a mantindre correctes els angles de *actualLineAngles*.
items	Llista de WheelItems que conté el menú, representant cadascun un element de la col·lecció emmagatzemada.
knobOrientation	Orientació del pom associat. Usat per poder mostrar el menú en la mateixa orientació que el pom quan aquest entra en la superfície.
lineModel	Línia que es dibuixa per delimitar els sectors. Només es té una, que es dibuixa diverses vegades aplicant la rotació que indique *actualLineAngles*.
menuChildren	Llista de menús que representen subcol·leccions de la pròpia col·lecció.
nSectors	Nombre de sectors del menú sense contar el de la *bridge mark*.
originAngle	Angle de la línia dreta de la *bridge mark*, que serveix de referència per calcular-ne la resta.
regionDict	Diccionari que associa cada contacte sobre el menú amb el sector sobre el que ha impactat.
sectorColor	Color del *sectorModel*, que és el qui apareix colorejant un sector determinat quan està sent impactat.
sectorModel	Sector colorejat que es col·loca sobre el sector del menú que està sent impactat per un contacte.

Taula 4. Atributs protegits de *TangiWheel*.

7.1.3 Propietats Públiques

Nom	NL	Descripció
BlockDimension		Per defecte, en minimitzar un menú les propietats de dimensió *Width* i *Height* així com les de posició *Left*, *Right*, etc. canvien per adaptar-se al nou tamany reduït. Activar aquesta propietat provoca que açò no es faça, considerant les dimensions com si el control no estiguera minimitzat.
Bottom	x	Posició sobre l'eix Y del límit inferior.
BridgeMarkAngle		Angle de la regió especial.
CanBeFingerClosed		Indica si el menú virtual pot ser eliminat fent un contacte sobre la marca central amb el dit.
CanSelectionsBeDone		Indica si es poden fer seleccions.
Caption		Text associat a un menú.
CaptionFont		Font del *Caption*.
CaptionVisible		Indica si *Caption* és visible. En cas de ser-ho, es col·loca a la part superior del menú just dalt de la regió especial i, en minimitzar-lo, se sitúa entre la *bridge mark* i el pom. En cas de no tindre pom associat, baix de la *bridge mark*.
Center		Posició central del menú.
ChainModel	x	Model de la cadena que uneix els TangiWheels. S'usa un fragment d'aquesta per a ser dibuixada, segons indique el corresponent *chainModelAttributes*.
Color		Color del control. Només hi ha un sol color per a tots els elements (menys per a les línies), i per a cadascun s'hi apliquen distintes tonalitats de transparència per donar un aspecte més atractiu.
Height	x	Altura del menú en píxels.
IsBlinking		Indica si la *shadow* ha de fer una animació parpadejant. Només té una semàntica visual.
ItemsFont		Font dels WheelItems associats que continguen text.
Left	x	Posició sobre l'eix X del límit esquerre.
LinesColor		Color de les línies que delimiten els sectors.

MaxVisibleItems		Nombre d'ítems màxim que es poden vore al mateix temps en el menú.
MenuParent		En cas de representar una subcol·lecció, indica quin menú és el qui té la col·lecció pare de la pròpia.
Mode		Indica si té un tangible associat (*Physical*) o no (*Virtual*). En el primer cas, les exploracions, les traslacions i les rotacions es realitzen amb el pom. En cas contrari, amb els dits. No s'ha de confondre amb la modalitat d'interacció (virtual o tàctil, tangible i híbrida) de la que es parla en capítols anteriors.
PhysicalExorationSpeed		Indica quantes vegades s'ha de realitzar una rotació (açò és, produir-se un esdeveniment *ContactChanged* amb un canvi de rotació) per a què es mostre per complet un ítem nou quan s'està explorant en mode físic.
Radius		Radi del menú.
Right	x	Posició sobre l'eix X del límit dret.
RotationModeActive		Indica si el menú està en mode rotació. Si val fals, està en mode d'exploració.
SectorAngle	x	Angle de cada sector que conté un ítem.
TextColor		Color del text dels ítems i de *Caption*.
Top	x	Posició sobre l'eix Y del límit superior.
VirtualExplorationSpeed		Indica quantes vegades s'ha de realitzar una rotació (açò és, produir-se un esdeveniment *ContactChanged* amb un canvi de rotació) per a què es mostre per complet un ítem nou quan s'està explorant en mode virtual.
VirtualMenuDestructionOnSelectItem		Si està actiu, indica que en seleccionar un element el menú es destruirà si és virtual.
Visible		Indica si el menú es dibuixa o no.
Width	x	Amplada del menú en píxels.

Taula 5. Propietats públiques de *TangiWheel*. La columna "NL" (Només Lectura) indica quines només poden ser llegides des d'una classe externa. Des de dins de *TangiWheel* sí que poden ser escrites.

7.1.4 Esdeveniments

Nom	Descripció
MarkerPositionChanged	Llançat quan el pom associat realitza una translació.
MarkerRotationChanged	Llançat quan el pom associat realitza una rotació.
RadiusChanged	Llançat quan el radi del menú canvia (per permetre la gestió apropiada per part dels WheelItems).

Taula 6. Esdeveniments de *TangiWheel*.

7.1.5 Mètodes Privats

Nom	Descripció
CalculateLineAngles	Calcula els angles de les línies dels sectors.
CalculatePositionsAroundCenter	Calcula les posicions de tots els elements gràfics del menú.
DrawItems	Dibuixa els diferents WheelItems en les posicions que hi pertoquen.
DrawLines	Dibuixa les línies dels sectors en el lloc on pertoquen.
DrawSectorModel	Dibuixa el sector colorejat damunt del sector del menú que pertoque.
MoveElements	Mou tots els elements gràfics del menú un increment de posició determinat.
RotateSoftly	Gestiona la rotació dels WheelItems i de les línies dels sectors quan es fa una exploració.
UpdateChildChainPoints	Quan es realitza un canvi de posició, actualitza el punt d'unió de la cadena amb un submenú determinat per aconseguir que la cadena tinga la mínima longitud possible.
UpdateOwnChainPoints	Quan es realitza un canvi de posició, una minimització o una restauració, actualitza el punt d'unió de la cadena amb el propi menú per aconseguir que la cadena tinga la mínima longitud possible.
UpdatePositionOfItems	Calcula les posicions dels WheelItems.
Wheel_PositionChanged	Gestiona un canvi de posició en el menú cridant als altres mètodes que actualitzen les posicions dels diferents elements dins del control.
Wheel_RotationChanged	Gestiona un canvi d'orientació en el menú cridant als altres mètodes que actualitzen les orientacions dels diferents elements dins del control (i, també, les posicions, ja que una rotació en el menú pot implicar un canvi de posició en alguns elements com, per exemple, la marca de llista).

Taula 7. Mètodes privats de *TangiWheel*.

7.1.6 Mètodes Protegits

Nom	Descripció
Dispose	Allibera recursos abans d'eliminar per complet la instància.
GetTappedSector	Calcula el sector sobre el que ha impactat un contacte.
LoadContent	En el moment de la inicialització, carrega els recursos gràfics (textures, efectes, fonts, etc.) necessaris.
OnContactChanged	Gestiona el comportament del menú davant un esdeveniment de canvi de posició o de rotació d'un contacte que està sobre el mateix.
OnContactDown	Gestiona el comportament del menú davant un esdeveniment de dipositació d'un contacte sobre el mateix.
OnContactEnter	Gestiona el comportament del menú davant un esdeveniment d'entrada d'un contacte en el mateix.
OnContactLeave	Gestiona el comportament del menú davant un esdeveniment d'eixida d'un contacte del mateix.
OnContactUp	Gestiona el comportament del menú davant l'alçament d'un contacte del mateix.
OnMediatorTangibleChanged	Gestiona el comportament del menú quan el seu tangible associat canvia la posició o és rotat.
OnMediatorTangibleEnter	Gestiona el comportament del menú quan el seu tangible associat entra en algun altre control de la superfície.
OnMediatorTangibleUp	Gestiona el comportament del menú quan el seu tangible associat és retirat de la superfície.
SetRotation	Una vegada s'ha establert la rotació del control, aquest mètode fa que tots els elements del menú s'orienten i es posicionen en la direcció i posició correcta.

Taula 8. Mètodes protegits de *TangiWheel*.

7.1.7 Mètodes Públics

Nom	Descripció
AddConnectedTangiWheel	Afegeix un submenú a la llista *menuChildren*.
AddItem	Afegeix un WheelItem a la llista *items*.
Draw	Mètode principal per realitzar el dibuixat del menú.
HitTest	Determina si un contacte està impactant dins de la regió delimitada pel menú.
Minimize	Minimitza el menú.
OnItemSelectionDone	Gestiona el comportament del menú quan s'alça un contacte d'un sector després de realitzar una selecció.
OnItemSelectionStarted	Gestiona el comportament del menú quan es posa un contacte sobre un sector.
RefreshItemCollection	Recarrega la llista de WheelItems.
RemoveConnectedTangiWheel	Elimina un submenú de *menuChildren*.
RemoveItem	Esborra un WheelItem d'*items*.
Restore	Restaura el menú que estava minimitzat.
Update	Mètode principal per realitzar les actualitzacions en el menú.

Taula 9. Mètodes públics de *TangiWheel*.

7.1.8 Classes d'Agregació

- **DrawableModel**: Classe abstracta que representa els elements gràfics del menú que no es dibuixen mitjançant imatges, sinó que es dibuixen amb primitives gràfiques (mitjançant triangles i *buffers* de vèrtexs).
- **CircleModel**: Subtipus de *DrawableModel* que representa un cercle. **SectorModel**: Subtipus de *DrawableModel* que representa un sector circular. Té dos colors (*color1* i *color2*) a banda del que hereten, i representen el mateix color que el de la classe pare però amb dos nivells de transparència distints. Així, s'aconsegueix que el sector tinga un color degradat.
- **LineModel**: Subtipus de *DrawableModel* que representa una línia.

7.2 Estructura dels Ítems Representats

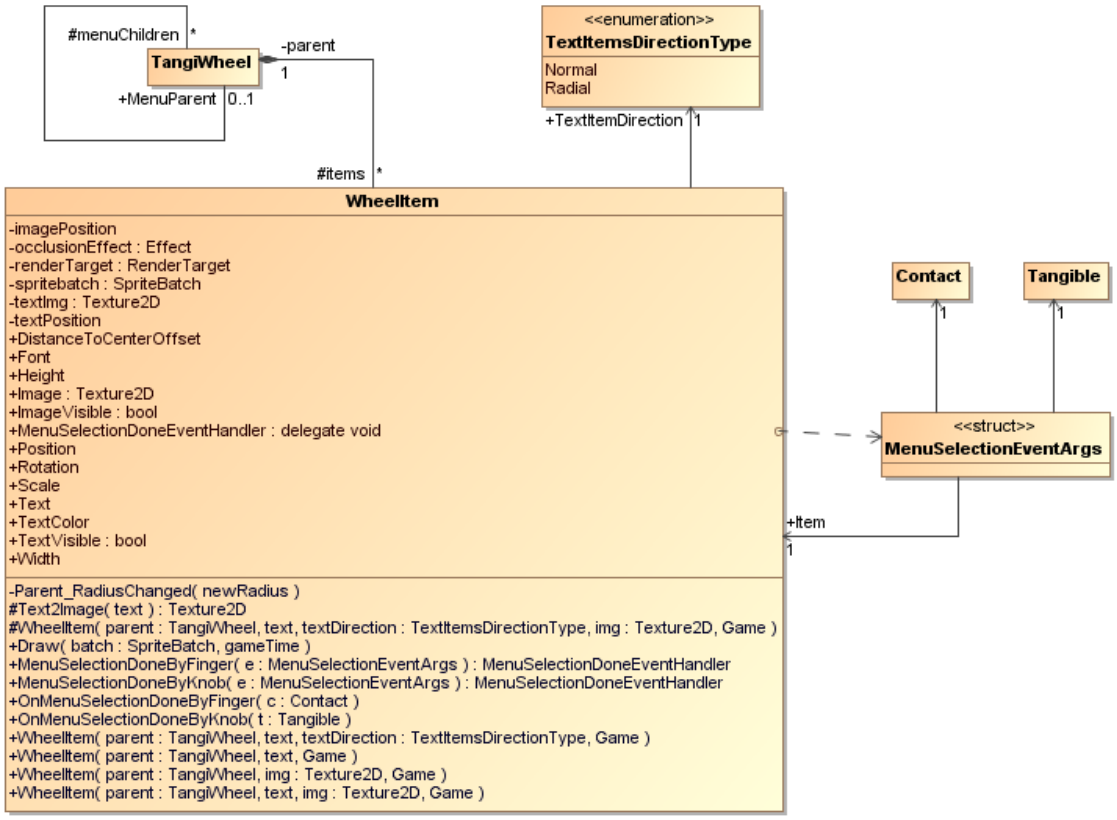


Figura 56. Diagrama de classes de *WheelItem*.

7.2.1 Atributs Privats i Protegits

Nom	Descripció
imagePosition	Posició de la imatge.
occlusionEffect	Efecte del *Pixel shader* que realitza l'oclusió dels ítems sota la regió de la *bridge mark* en el menú quan s'explora.
parent	WheelMenu al qual pertany.
renderTarget	Quan es vol transformar el text de l'ítem en imatge, es dibuixa sobre aquest objecte i després es guarda el resultat com una imatge.
spritebatch	S'encarrega d'emmagatzemar tots els objectes a ser dibuixats i els passa tots alhora a la targeta gràfica, de forma que s'augmenta l'eficiència per no estar enviant un per un els objectes (açò es traduiria en moltes interrupcions).
textImg	Imatge del text després d'haver sigut dibuixat sobre l'objecte *renderTarget*.
textPosition	Posició del text.

Taula 10. Atributs privats i protegits de *WheelItem*.

7.2.2 Propietats Públiques

Nom	NL	Descripció
DistanceToCenterOffset		El menú col·loca els ítems a una distància fixa del punt central, centrant-los en la regió disponible del sector. Amb aquesta propietat es controla un desplaçament afegit en la posició de l'ítem (positiu allunyant-lo i negatiu acostant-lo al centre del TangiWheel).
Font		Font del text.
Height	x	Altura de la imatge més el text.
Image		Imatge de l'ítem.
ImageVisible		Indica si la imatge ha de dibuixar-se.
Position		Posició en coordenades del món en la que es dibuixarà l'ítem.
Rotation		Rotació amb la que es vorà l'ítem.
Scale	x	Escala de l'ítem. Aquesta només es pot llegir, ja que es calcula automàticament donades les dimensions del menú. La forma de calcular-la és: sabent una mida adequada per a un radi concret del menú, en canviar aquest radi es pot saber quina és la nova mida realitzant una regla de tres.
Text		Text de l'ítem.
TextColor		Color del text.
TextItemDirection		Direcció (normal o radial) de l'ítem amb només text.
TextVisible		Indica si el text és visible o no.
Width	x	Amplada de la imatge més el text.

Taula 11. Propietats públiques de *WheelItem*. La columna "NL" (Només Lectura) indica quines només poden ser llegides des d'una classe externa. Des de dins de *WheelItem* sí que poden ser escrites.

7.2.3 Esdeveniments

Nom	Descripció
MenuSelectionDoneByFinger	Llançat quan l'ítem és seleccionat amb un dit.
MenuSelectionDoneByKnob	Llançat quan l'ítem és seleccionat amb un pom.

Taula 12. Esdeveniments de *WheelItem*.

7.2.4 Mètodes Privats i Protegits

Nom	Descripció
Parent_RadiusChanged	Quan el menú canvia el seu radi, aquest mètode recalcula l'escala de l'ítem.
Text2Image	Passa un text a imatge dibuixant-lo sobre l'objecte *renderTarget*.

Taula 13. Mètodes privats i protegits de *WheelItem*.

7.2.5 Mètodes Públics

Nom	Descripció
Draw	Realitza el dibuixat de la imatge i del text, si escau.
OnMenuSelectionDoneByFinger	Quan es detecta des del menú que un dit impacta sobre un element, s'invoca a aquest mètode, que només llança l'esdeveniment *MenuSelectionDoneByFinger*.
OnMenuSelectionDoneByKnob	Quan es detecta des del menú que un pom impacta sobre un element, s'invoca a aquest mètode, que només llança l'esdeveniment *MenuSelectionDoneByKnob*.

Taula 14. Mètodes públics de *WheelItem*.

Capítol 4. Experiments

Dels diferents aspectes considerats en el disseny de TangiWheel (organització dels elements, mecanismes d'interacció i suport multiusuari), s'ha realitzat una avaluació experimental dels tres modes d'interacció: tangible, virtual (també anomenat tàctil o, en anglès, *touch*) i híbrid. La resta d'aspectes seran considerats més endavant en futures experimentacions. En el present treball s'han estudiat les següents característiques:

- Adquisició i manipulacions bàsiques realitzades per establir la posició i l'orientació dels menús.
- Selecció d'una seqüència d'elements d'entre un conjunt de categories.
- Composició de sèries d'elements, que requereix no només explorar les col·leccions sinó, a més, inserir-hi o esborrar-hi ítems.

L'objectiu d'aquest disseny experimental era estudiar l'efecte de l'entrada de l'usuari en les interaccions (i.e. dits o poms) sobre el rendiment a l'hora de manipular col·leccions, mesurant el temps per acomplir la tasca i el nombre de manipulacions requerides. La complexitat dels experiments s'incrementava gradualment per poder avaluar l'ús del control sota diferents circumstàncies. TangiWheel és el primer control per manipular col·leccions en taules interactives que suporta un mode híbrid d'interacció amb alts valors de semblança entre interaccions tàctils i tangibles. Per això, és important obtindre informació sobre quines situacions es beneficien més d'aquest nou tipus d'interacció que d'unes propostes purament virtuals o purament tangibles.

1 Participants

Vint-i-tres voluntaris participaren en aquest estudi, 16 homes i 7 dones, la majoria estudiants o doctorands de la Universitat Politècnica de València. Un d'ells era esquerrà i dos, ambidextres. Les seues edats variaven des de 21 fins als 40 (M=27.5, SD=5.23). Divuit participants confessaren utilitzar ordinadors cada dia; tres, gairebé tots els dies i, dos, una o dues vegades per setmana. Tretze, a més, declararen ser usuaris regulars de dispositius tàctils, mentre que deu tenien poca o cap experiència amb aquests. Vuit participants no tenien cap experiència prèvia amb superfícies interactives, i la resta en tenien poca (provinent, sobretot, d'exhibicions i demostracions).

2 Equipament

Els experiments han sigut realitzats sobre dues unitats de Microsoft Surface. El *framework* d'interacció per a TangiWheel està implementat en C# usant la Microsoft Surface SDK v1.0 amb les llibreries del nucli específiques per al *framework* Microsoft XNA. També s'han utilitzat els tangibles (poms) que tenen a la base unes etiquetes amb un codi imprès reconegut pel sistema de visió de la Surface.

3 Procediment

Les sessions d'avaluació van ser organitzades en funció de la disponibilitat dels participants. Per tal d'evitar l'aprenentatge d'aquests a partir de l'observació als seus companys, només dues persones estaven al laboratori al mateix temps, però cadascun a una unitat Surface distinta, separats per una pantalla plegable.

Els participants reberen una xerrada introductòria i una demostració en directe de les interaccions típiques seguides d'una sessió d'entrenament lliure (tot i que supervisat), on els participants podien familiaritzar-se amb el control i adquirir cert confort previ amb les interaccions, abans de les sessions d'avaluació pròpiament dites. Aquest entrenament durava, aproximadament, uns 15-20 minuts.

L'estudi començà amb cada participant, tot sol, realitzant els experiments d'adquisició i manipulacions bàsiques. Aquests eren seguits pels de selecció d'elements i els d'edició de sèries. Per evitar que els participants interactuaren amb les diferents alternatives de disseny en el mateix ordre, el mode d'interacció va ser establert seguint el disseny dels quadrats llatins [2, 3]. A més, els participants van ser animats a acomplir les tasques el més ràpidament possible amb un premi per al més ràpid.

El programari per a l'execució dels experiments guardava, automàticament, totes les interaccions dels usuaris en una sèrie de fitxers de diari. Aquest conjunt de fitxers eren postprocessats per obtindre les dades per a l'anàlisi estadístic quantitatiu. La informació principal extreta estava relacionada amb els temps de compleció de les tasques i el nombre corresponent d'accions requerides. Diverses parts de les sessions van ser gravades en vídeo per realitzar una anàlisi més profunda (bàsicament, estudiar patrons de comportament en els usuaris). A més, els voluntaris emplenaren diversos qüestionaris relatius a la facilitat i comoditat d'ús així com a l'efectivitat del control.

4 Experiment 1: Adquisició i Manipulacions Bàsiques

L'ús d'un element d'interfície involucra, almenys, dues fases d'interacció [11]: típicament, adquisició i manipulació. Al control TangiWheel, les tasques d'adquisició i manipulació bàsiques són necessàries per establir la posició i l'orientació del menú abans d'explorar i seleccionar elements. L'objectiu d'aquest experiment era, per tant, explorar l'efectivitat dels modes d'interacció virtual i tangible en executar les interaccions típiques preliminars comentades.

4.1 Tasca

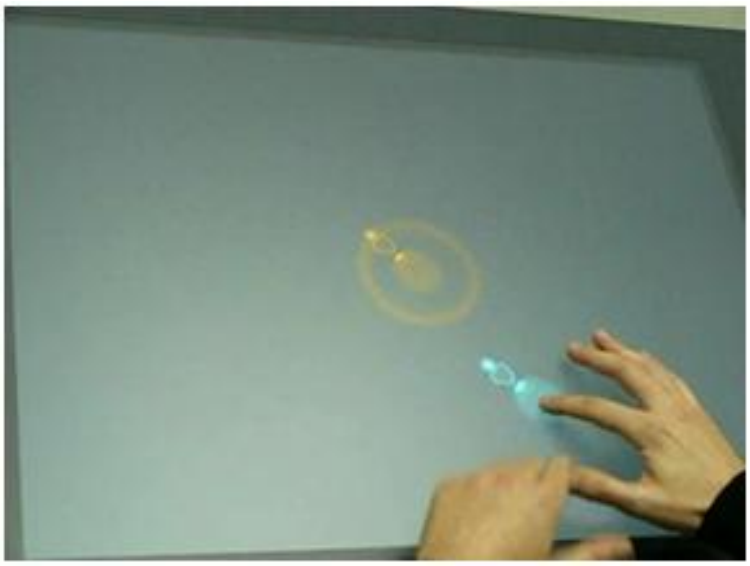


Figura 57. Experiment 1: Adquisició i manipulacions bàsiques. Mode virtual.

Figura 58. Experiment 1: Adquisició i manipulacions bàsiques. Mode tangible.

Se sol·licita als participants que establisquen la posició i l'orientació de vint-i-cinc instàncies TangiWheel. Un objectiu taronja amb l'orientació desitjada es mostra al centre de la superfície, i es mostra també un altre control blau en una posició pseudo-aleatòria i amb una rotació estàndard (vore Figura 57 i Figura 58). L'usuari ha d'adquirir el control i efectuar-hi les manipulacions corresponents per casar la posició i orientació del menú blau amb les de l'objectiu taronja. Quan ambdós se superposen (amb un lleu marge d'error), desapareixen i es mostren altres dos controls per repetir la tasca, fins que s'arriba a un total de vint-i-cinc repeticions. Les posicions i les orientacions estan predefinides (tot i que els usuaris no les coneixen), però l'ordre d'aparició es calcula aleatòriament per a cada execució de la tasca (així, n'és un diferent per a cada participant).

4.2 Procediment

Aquest experiment va ser efectuat per cada participant dues vegades: una amb el mode d'interacció virtual (només usant els dits) i altre amb mode d'interacció tangible (només usant poms) de TangiWheel. Aquest mode s'assignà a cada participant seguint un disseny de quadrat llatí amb la finalitat d'evitar efectes d'ordre (i.e. que no tots els usuaris realitzaren primer la tasca amb un mode i després amb l'altre, sinó que aquest ordre variara).

Amb el mode tangible, el voluntari havia de prendre el pom amb una mà, adquirir el control posant el tangible en la seua regió central i, llavors, dur-lo cap a la localització objectiu, realitzant els ajustaments de rotació necessaris. La comprovació de si el menú casava amb l'objectiu es realitzava quan el tangible romania quiet durant 250ms.

En l'enfocament virtual, el participant havia d'aconseguir el posicionament arrossegant el control amb el dit des de la seua marca central i, després, ajustar l'orientació descrivint gestes circulars en l'àrea dels ítems. La comprovació d'acoblament es realitzava quan el dit abandonava el menú.

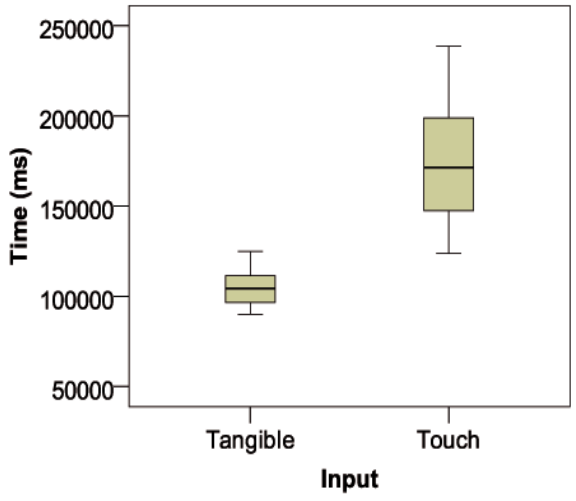


Figura 59. Experiment 1: Adquisició i manipulacions bàsiques. Temps de compleció de la tasca.

Aquestes operacions havien de ser realitzades amb la major rapidesa possible, d'acord amb les instruccions.

4.3 Resultats

Per a cada mode d'interacció s'han mesurat tres variables, sobre les quals s'aplica el superíndex *Virtual* o *Tangible* depenent del mode en el que funcione TangiWheel:

- T_{total}: Temps mitjà per completar l'experiment (vint-i-cinc repeticions).
- T_{acob}: Temps mitjà per aconseguir un sol acoblament.
- A_{acob}: Nombre mitjà d'accions requerides per realitzar un sol acoblament.

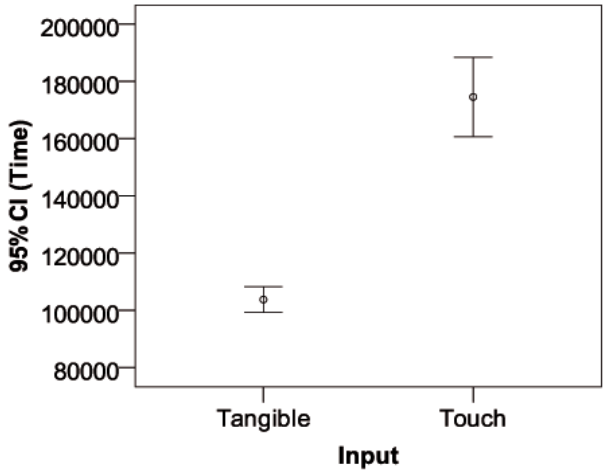


Figura 60. Experiment 1: Adquisició i manipulacions bàsiques. Interval de confiança per al temps de compleció de la tasca.

Com es pot observar a la Figura 59 i a la Figura 60, als participants els costa més completar la tasca en mode virtual ($T_{total}^{Virtual}$: M=174.56s, SD=31.210s) que en mode tangible ($T_{total}^{Tangible}$: M=103.799s, SD=9.714s).

L'anàlisi de la variància (ANOVA) efectuat sobre el temps necessari per completar l'experiment demostra que el mode d'interacció té un efecte sobre l'adquisició preliminar i les manipulacions bàsiques (F=94.327, p-valor=0.000), i que el mode tangible supera al virtual. Aquests resultats coincideixen amb altres estudis empírics sobre posicionament de diferents elements d'interfície en ambdós enfocaments, multitàctil i tangible [35] [24], els quals mostren que les interfícies tangibles són efectives en tasques de distribució espacial. Els nostres experiments proveeixen proves empíriques addicionals en aquest aspecte per a les nostres tècniques de posicionament i orientació de TangiWheel.

Els resultats obtinguts poden ser deguts a que les mans estan acostumades a rotar i posicionar (agafar) elements en nombroses accions diàries, a diferència dels dits, la funció principal dels quals és tocar o seleccionar. Agafar objectes és,

per tant, una interacció més natural per realitzar accions de moviment que arrossegar els objectes per la superfície amb els dits. Es més, el mode virtual es veu afectat per una qüestió d'interacció observada en [35] i anomenada "error d'eixida" (de l'anglès, *exit error*). Açò es refereix a la dificultat d'amollar l'objecte virtual sense causar algun tipus de moviment extra inintencionat. Aquest és un problema inherent en mode tàctil sobre superfícies interactives que pot afectar als usuaris en major o menor manera. En concret, s'observà que afectava al correcte posicionament dels controls i, al final, contribuïa a temps significativament superiors en el mode virtual.

S'ha realitzat un estudi sobre el temps necessari per portar un control al seu lloc objectiu i el nombre d'accions requerides. Aquest acoblament costava més temps en mode virtual ($T_{acob}^{Virtual}$: M=5.861s, SD=2.024s) que en tangible ($T_{acob}^{Tangible}$: M=3.772s, SD=0.930s). No obstant, eren requerides menys accions en l'enfocament virtual ($A_{acob}^{Virtual}$: M=3.45 accions, SD=2.53 accions) que en el tangible ($A_{acob}^{Tangible}$: M=7.17 accions, SD=2.43 accions). Açò pot ser explicat pel fet de què una rotació completa en el menú pot ser efectuada de forma contínua amb un dit descrivint circumferències en la superfície, i això compta com una sola acció. No obstant, la mateixa operació efectuada amb el pom requereix més accions, ja que la rotació continua no pot aconseguir-se (la mà no pot rotar 360º sobre ella mateixa), i cada moviment de la mà és comptabilitzat com una acció addicional.

Per realitzar la comparació de mitjanes no s'ha pogut aplicar un test paramètric com el t-Student, ja que un test Kolmogorov-Smirnoff mostrà que la condició de normalitat no s'acomplia. Per tant, s'ha hagut de realitzar mitjançant tests Mann-Whitney. Així, s'han trobat diferències significatives en el temps necessari per realitzar un sol acoblament ($H_0: T_{acob}^{Tangible} = T_{acob}^{Virtual}$; z=-19.06, p-valor=0.00) i també en el nombre d'accions requerides ($H_0: A_{acob}^{Tangible} = A_{acob}^{Virtual}$; z=-22.46, p-valor=0.00).

A la vista d'aquests resultats, malgrat que es podria pensar que el mecanisme tàctil (virtual) no és un bon mode d'interacció per a TangiWheel, aquesta conclusió serà posada a prova en el següent experiment (secció 5).

5 Experiment 2: Selecció d'Elements

Seleccionar una sèrie d'elements objectiu és una tècnica clàssica en estudis cognitius [25], en la qual es produeix un estímul visual que es manté sobre la pantalla mentre l'usuari realitza una selecció.

L'objectiu d'aquest experiment és comparar l'efectivitat dels diferents modes d'interacció de TangiWheel (tangible, virtual i híbrid) a l'hora de buscar i seleccionar elements en diferents col·leccions en resposta a un estímul visual.

Figura 61. Experiment 2: Selecció d'elements. Mode virtual.

Figura 62. Experiment 2: Selecció d'elements. Mode tangible.

5.1 Tasca

Els estímuls visuals són imatges pertanyents a les següents categories: números, instruments musicals, animals, fruites, notes musicals i senyals de tràfic. En promig, cada col·lecció conté onze elements. Els participants han de buscar i seleccionar, seqüencialment i el més ràpidament possible, un total de vint elements objectiu que pertanyen a diferents categories. Una vegada se selecciona un element correctament, es mostra el següent.

5.2 Procediment

Cada participant va executar la tasca 3 vegades variant el mode d'interacció de TangiWheel. L'ordre de les execucions s'ordenà seguint un disseny de quadrat llatí per evitar efectes d'ordre. Mentre que el mode híbrid permetia la combinació lliure de poms i dits, aconseguint la funcionalitat màxima del control, els enfocaments tangible i virtual només responien a poms i a dits, respectivament. Açò significa que, en mode tangible, els poms s'usaven tant per obrir col·leccions com per a seleccionar elements; mentre que, en mode virtual, aquestes operacions es realitzaven només amb els dits (vore Figura 61 i Figura 62).

5.3 Resultats

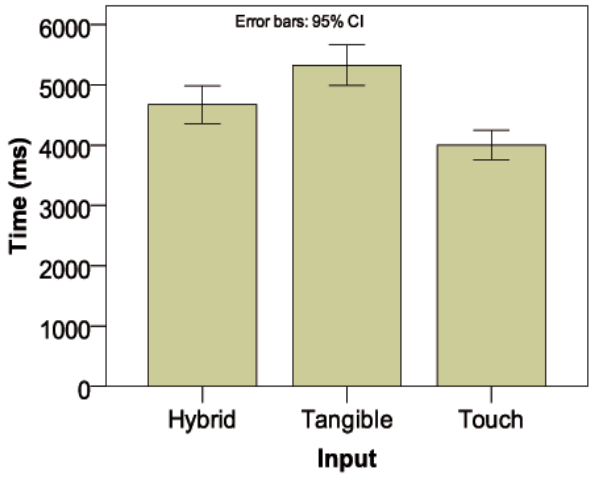


Figura 63. Experiment 2: Selecció d'elements. Temps de selecció.

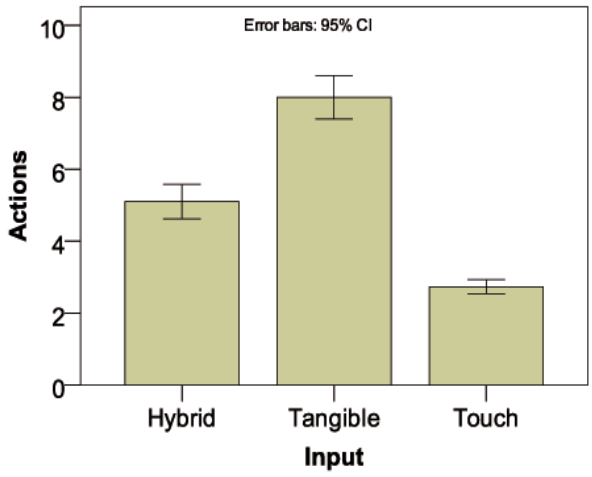


Figura 64. Experiment 2: Selecció d'elements. Nombre d'accions.

En aquest experiment s'han mesurat tres variables per a cada modalitat, sobre les quals s'aplica el superíndex *Virtual* o *Tangible* depenent del mode en el que funcione TangiWheel:

- Temps mitjà de compleció de l'experiment (vint seleccions).
- T_{sel}: Temps mitjà per buscar i seleccionar un element.
- A_{sel}: Nombre mitjà d'accions requerides per realitzar la cerca i la selecció d'un element.

L'aplicació de l'ANOVA als temps obtinguts en completar l'experiment en la seua totalitat ha demostrat que estan influenciats significativament pel mode d'interacció (F=15.512, p-valor=0.000). Posteriorment, aplicant comparacions per parelles usant tests t (corregits pel mètode de Bonferroni), s'ha obtingut que totes tres modalitats eren significativament diferents (vore Taula 15). Els resultats mostren que el mode més ràpid és el virtual, seguit per l'híbrid i, finalment, pel tangible.

Comparació	Diferències mitjanes (ms)	p-valor
Híbrida – Tangible	-19662.57	0.013
Híbrida – Virtual	16804.16	0.041
Tangible – Virtual	36466.73	0.000

Taula 15. Comparació per parelles dels modes d'interacció per al temps requerit en completar l'experiment 2: Selecció d'elements.

La Figura 63 mostra els temps mitjans de selecció d'un sol element per a les tres modalitats d'interacció. L'ús exclusiu dels dits dóna millors resultats que els altres dos modes, a les quals se'ls permet l'ús de tangibles. La Figura 64 mostra que la proposta virtual és la que també requereix menor nombre d'accions, seguida per l'híbrida i la tangible.

Com la condició de normalitat no se satisfà, es duu a terme una comparació per parelles del temps que requereix completar una selecció (T_{sel}) i les accions necessàries (A_{sel}). Es realitza usant tests Mann-Whitney per determinar si les diferències són significatives. Tal i com es pot vore a la Taula 16, els tests mostren que hi ha diferències significatives entre totes tres modalitats d'interacció per a les dues variables mesurades.

H_0	z	p-valor
$T_{sel}^{Híbrid} = T_{sel}^{Tangible}$	-3.089	0.002
$A_{sel}^{Híbrid} = A_{sel}^{Tangible}$	-9.488	0.000
$T_{sel}^{Híbrid} = T_{sel}^{Virtual}$	-2.388	0.017
$A_{sel}^{Híbrid} = A_{sel}^{Virtual}$	-5.932	0.000
$T_{sel}^{Tangible} = T_{sel}^{Virtual}$	-5.588	0.000
$A_{sel}^{Tangible} = A_{sel}^{Virtual}$	-16.882	0.000

Taula 16. Comparació de T_{sel} i A_{sel} per a les diverses modalitats d'interacció.

Els resultats indiquen que el nombre d'accions i el temps mitjans necessaris per cercar i seleccionar un element són també significativament menors per a la proposta virtual i, per tant, suggereixen que els dits són més efectius en escena-

ris on es requereixen seleccions en una col·lecció que està situada en una posició fixa i que, per tant, no requereix de manipulacions bàsiques com són translacions i rotacions sobre la superfície. Esta tècnica, llavors, seria apropiada per a usuaris treballant individualment sobre interfícies amb col·leccions representades en posicions fixes predefinides i amb una orientació 2D predeterminada.

No obstant, en aplicacions on diferents usuaris han d'adquirir i moure col·leccions amb pocs elements al seu espai personal, la proposta tangible seria més adient que la virtual, perquè el nombre d'accions d'adquisició seria major que el nombre d'exploracions. Finalment, en escenaris amb un nombre relativament gran d'adquisicions i també amb col·leccions grans, una modalitat híbrida seria la millor, car integraria els avantatges d'ambdós enfocaments anteriors. TangiWheel suporta els tres modes d'interacció, per tant, pot ser considerat un control flexible que pot ser usat amb efectivitat en un ampli conjunt d'escenaris.

5.3.1 Rendiment de la Modalitat Híbrida

Quant al rendiment de la proposta híbrida, la Taula 17 resumeix l'ús de dits i de poms per a diverses operacions importants en termes del nombre mitjà d'operacions per usuari i el percentatge d'aquests. Els resultats mostren que els subjectes prefereixen, generalment, manipular el control utilitzant dits (72.5%) a tangibles (27.5%) quan tenen l'oportunitat. Els dits eren utilitzats, majorment, per a seleccionar ítems dels menús i establir l'objectiu (82.75% i 97.23%, respectivament). Mitjançant les gravacions de vídeo es va poder observar més tard que els usuaris tenien dificultats a l'hora d'usar els poms per a seleccionar un element.

No obstant, els resultats també mostren que els usuaris feien major ús dels poms per crear altres instàncies TangiWheel (61.25%), que eren després explorats amb el tangible. Açò suggereix que o bé trobaven més senzilla la manipulació amb el pom o bé l'exploració dels elements (o ambdues). En promig, cada participant usava quasi 6 poms diferents (M=5.7) per acomplir la tasca, suggerint que feien un ús extensiu d'ells i que no limitaven el nombre de tangibles usats simultàniament.

	Dits		Poms	
Operació	Mitjana	%	Mitjana	%
Manipulació	41.60	72.5	15.78	27.5
Instanciació menú	4.04	38.75	6.39	61.25
Selecció	25.86	82.75	5.39	17.24
Establiment objectiu	21.30	97.23	0.61	2.77

Taula 17. Experiment 2: Selecció d'elements. Ús de dits i de poms (operacions per usuari) en mode híbrid.

6 Experiment 3: Edició de Sèries

Els experiments 1 (secció 4) i 2 (secció 5) avaluaven el rendiment en activitats bàsiques de manipulació sota diferents modalitats d'interacció i la localització i selecció d'elements en col·leccions estructurades per a què coincidiren amb mostres individuals extretes d'una sèrie. En altres paraules, hi havia una relació un a un entre l'estímul visual i l'element objectiu sent manipulat quan s'executava una acció simple (rotació, translació o selecció). Tot i això, per explorar completament la possible influència en el rendiment dels diferents modes d'interacció de TangiWheel, s'ha dissenyat un experiment en el que hi ha un nombre major de mostres i d'accions possibles.

L'objectiu principal d'aquest experiment és incrementar la càrrega cognitiva quan es manipula una col·lecció sota diferents modalitats i involucrar a l'usuari en una tasca complexa que requereix la combinació d'operacions de rotació, translació, cerca, selecció, inserció i esborrat d'elements. Açò ens permet simular un escenari més realista augmentant la complexitat cognitiva, però mantenint un entorn experimental controlat en el qual es pot efectuar un estudi comparatiu de les diferents modalitats d'interacció.

6.1 Tasca

Es va sol·licitar als participants que, o bé crearen unes sèries d'elements seleccionant-los d'un conjunt de col·leccions o bé que modificaren una sèrie ja existent (vore Figura 65 i Figura 66). Per a fer-ho, havien d'explorar les col·leccions i afegir o esborrar elements a les sèries.

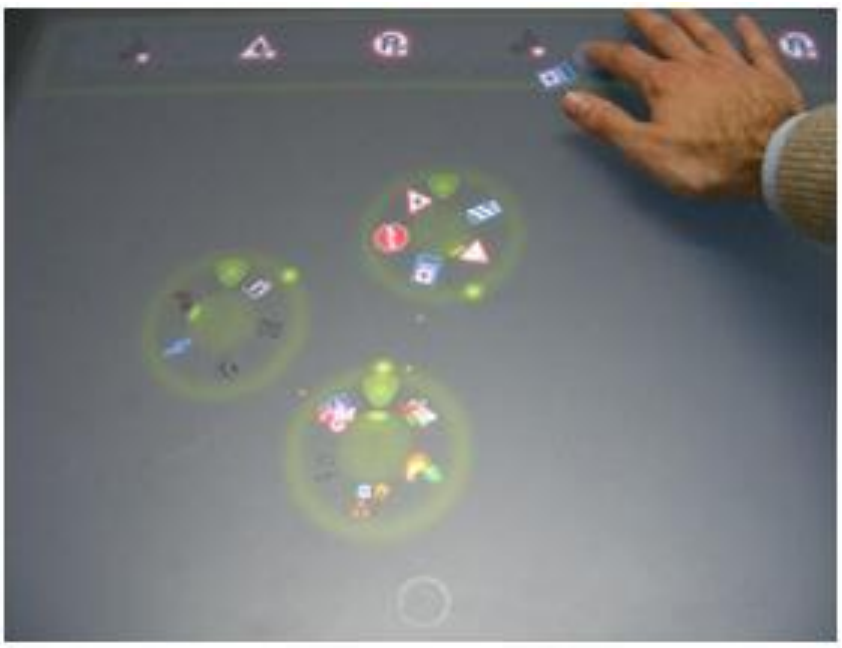


Figura 65. Experiment 3: Edició de sèries. Mode virtual.

Figura 66. Experiment 3: Edició de sèries. Mode tangible.

6.2 Procediment

Es realitzaren tres execucions de l'experiment, una per a cada mode d'interacció de TangiWheel. Es va fer servir un disseny de quadrat llatí per establir l'ordre de les execucions per a cada usuari i així evitar efectes d'ordre.

Als participants se'ls va donar una fulla d'instruccions on se'ls demanava crear dos series noves i modificar-ne una existent. Les sèries proposades per a cada estil d'interacció eren de dificultat similar en termes de longitud i variabilitat dels elements inclosos.

En la modalitat virtual, els subjectes havien de cercar i seleccionar ítems en diferents col·leccions utilitzant els seus dits sobre controls virtuals, que podien ser rotats o moguts per la superfície, com en els experiments previs. Per inserir un element en una posició específica de la sèrie de referència, havien d'arrossegar l'ítem des de la seua col·lecció fins a la posició que desitjaren de la sèrie, sense alçar el dit de la taula. Per esborrar un element de la llista, només havien de polsar un botó per a tal efecte que es mostrava al costat de cada ítem.

D'altra banda, per afegir un element en mode tangible, els subjectes havien d'usar poms per cercar i seleccionar els elements. Com aquests estaven organitzats en diverses categories niades, podien associar un pom a una col·lecció en qualsevol moment per accedir-hi directament i usar el tangible per rotar o moure el menú a qualsevol àrea de la superfície. Els elements eren seleccionats col·locant-los un pom al damunt i eren inserits en la llista objectiu col·locant el pom amb l'ítem associat en la posició desitjada de la sèrie. L'esborrat dels elements es feia mitjançant un pom especial que servia només per a aquesta tasca, i s'aconseguia tan sols posant el tangible sobre l'element a eliminar.

6.3 Resultats

Per a cada mode d'interacció s'han mesurat tres variables:

- T_{total}: Temps mitjà per completar l'experiment (dos creacions i una modificació).
- T_{edi}: Temps mitjà per aconseguir l'edició d'una sola sèrie.
- A_{edi}: Nombre mitjà d'accions requerides per realitzar una sola edició.

La Figura 67 mostra la distribució del temps requerit per completar l'experiment en cada modalitat d'interacció i, la Figura 68, el temps necessari per completar l'edició d'una sola sèrie. Aplicant una ANOVA als temps requerits per a completar l'experiment (T_{total}) es demostra que aquest està afectat per la modalitat en qüestió (F=4.413, p-valor=0.016).

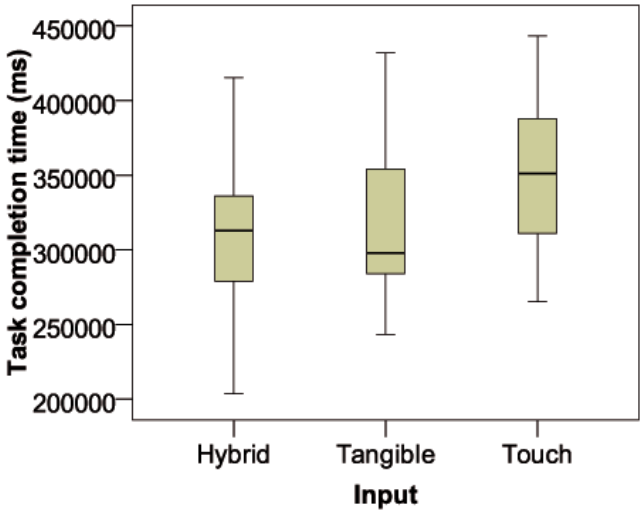


Figura 67. Experiment 3: Edició de sèries. Temps de compleció de la tasca.

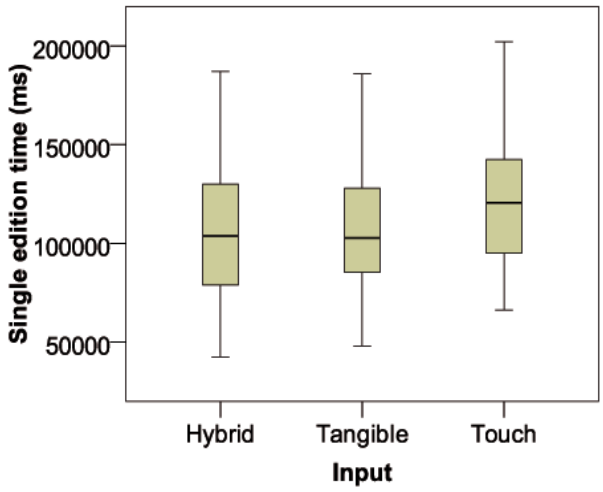


Figura 68. Experiment 3: Edició de sèries. Temps d'una sola edició.

Posteriorment, realitzant comparacions per parelles amb test t (corregits pel mètode de Bonferroni), s'observa que l'enfocament virtual és significativament més lent que l'híbrid, però no es troben diferències significatives entre tangible i virtual o entre tangible i híbrid (vore Taula 18). No obstant, el temps mitjà requerit per completar la tasca és major per a les propostes virtual i la tangible que per a l'híbrida; i, per a la virtual, major que per a la tangible.

Comparació	Diferències mitjanes (ms)	p-valor
Híbrid – Tangible	-11892.487	1.000
Híbrid – Virtual	-44728.876	0.018
Tangible – Virtual	-32836.389	0.103

Taula 18. Experiment 3: Edició de sèries. Comparació de T_{total}.

També s'ha analitzat el temps necessari per completar l'edició d'una sola sèrie (T_{edi}) i el nombre d'accions requerides (A_{edi}) per fer-ho. La Figura 69 i la Figura 70 mostren les corresponents representacions dels valors mitjans. Aplicant una ANOVA a T_{edi} s'ha demostrat que està afectat pel mode d'interacció (F=4.503, p-valor=0.012).

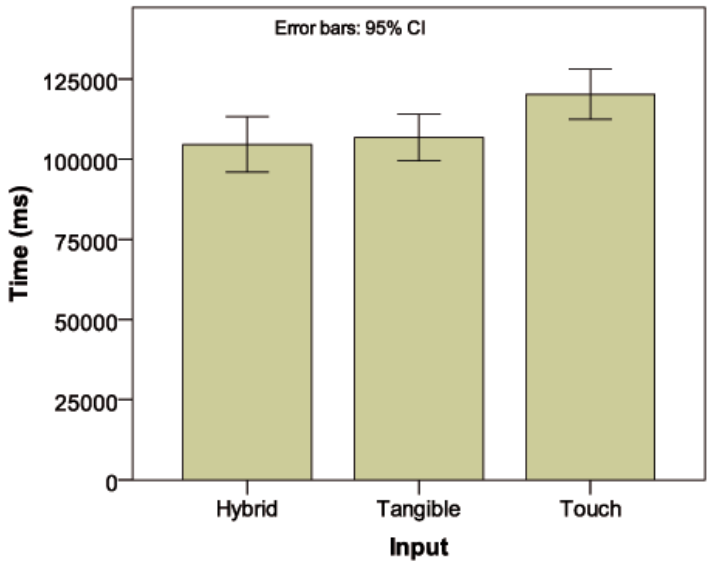


Figura 69. Experiment 3: Edició de sèries. Temps d'edició d'una sola sèrie.

Realitzant comparacions per parelles utilitzant tests t, amb correcció de Bonferroni (vore Taula 19), trobem que la modalitat virtual és significativament més lenta que l'híbrida. Encara que no arriba a ser significatiu, per poc, pareix que la virtual també presenta temps majors que la tangible (p-valor=0.053 amb un nivell de confiança del 95%). On no s'hi troben diferències estadísticament significatives és en les propostes híbrida i tangible.

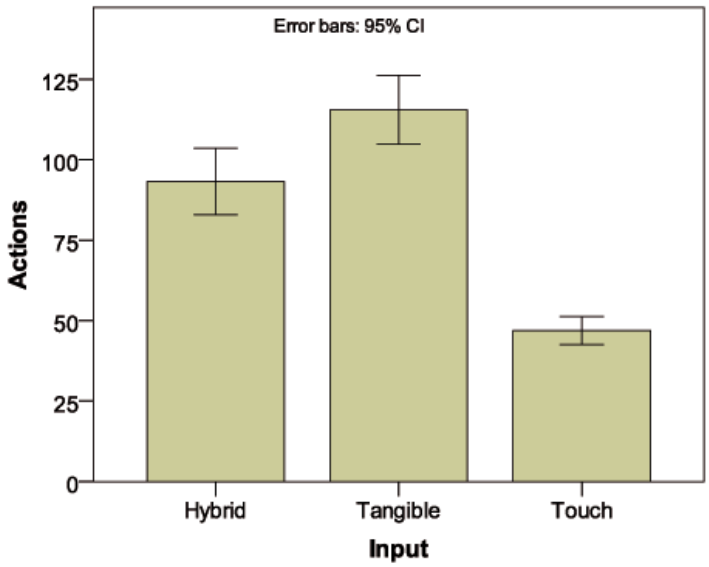


Figura 70. Experiment 3: Edició de sèries. Accions requerides per editar una sola sèrie.

Comparació	Diferències mitjanes (ms)	p-valor
Híbrid – Tangible	-2153.072	1.000
Híbrid – Virtual	-15611.345	0.018
Tangible – Virtual	-13458.273	0.053

Taula 19. Experiment 3: Edició de sèries. Comparació de T_{edi}.

Aplicant un ANOVA a A_{edi}, es demostra que el mode d'interacció de Tangi-Wheel és un factor que afecta a aquest paràmetre (F=60.269, p-valor=0.000). Comparant dos a dos les modalitats mitjançant tests t (amb la correcció de Bonferroni) s'obté que les tres presenten resultats significativament diferents (vore Taula 20). En mode virtual es requereix un nombre menor d'accions que en els altres dos, i en tangible es requereixen més accions que en híbrid.

Comparació	Diferències mitjanes (ms)	p-valor
Híbrid – Tangible	-22.253	0.002
Híbrid – Virtual	46.350	0.000
Tangible – Virtual	68.603	0.000

Taula 20. Experiment 3: Edició de sèries. Comparació de A_{edi}.

Els resultats, llavors, mostren que treballant amb el mode virtual es requereixen menys accions que en híbrid però, en canvi, aquest últim presenta uns temps menors. No hi ha diferència significativa entre el temps de la proposta híbrida i de la tangible, però, en canvi, sí que hi ha evidències de que la segona requereix més accions. Per tant, es podria concloure que la modalitat híbrida és la més efectiva per a escenaris més complexos que el descrit en l'experiment de selecció d'elements (secció 5). Açò es deu a que l'edició de sèries requereix una combinació de les accions estudiades en els dos experiments anteriors. Davant d'aquesta situació, el pobre rendiment de la interacció de tipus virtual per a tasques de manipulacions bàsiques (arrossegar i rotar) fa aquesta modalitat menys

efectiva (malgrat que mostra un bon comportament davant de cerques i seleccions d'ítems en col·leccions). De fet, l'anàlisi dels vídeos gravats durant les sessions revela dues situacions importants respecte a açò:

Primer, en modes híbrid i tangible, els subjectes es beneficiaven dels poms com a elements contenidors i usaven un sol d'aquests per realitzar múltiples insercions d'elements que apareixien repetidament a les sèries. D'altra banda, quan usaven el mode virtual, havien d'arrossegar els elements per la superfície varies vegades per inserir els elements repetits en diferents parts de la llista. Això penalitzava el temps de compleció de la tasca perquè, tal i com es mostra en el primer experiment (vore secció 4), aquesta modalitat d'interacció és menys efectiva per realitzar operacions de manipulacions bàsiques.

Segon, també es podia observar que, donat que el nombre d'elements en les sèries era gran, el nombre d'instàncies TangiWheel mostrades en la superfície creixia amb el temps, forçant, amb el mode virtual, a recol·locar-les per a què no interferiren amb la tasca. De nou, això també penalitzava el temps.

Llavors, es pot concloure que els dissenys purament tàctils o tangibles presenten pitjors resultats que el mode híbrid, el qual permet usar els dos anteriors a la vegada. Així, el comportament en aquesta última modalitat és millor per a situacions que requerisquen moviments de col·leccions sobre superfícies que tinguen un gran nombre d'elements representats a sobre. També, en aquelles on siga necessari reutilitzar elements de les col·leccions en diferents àrees de la superfície; i en les que requerisquen la reorientació dels contenidors dels elements. Aquests requeriments estan presents quan diversos participants comparteixen l'espai de treball o en situacions on es precisen distribucions flexibles de col·leccions. El fet de tindre un millor comportament de l'estil híbrid d'interacció en aquestes situacions és important, ja que les superfícies estan dissenyades, principalment, per suportar interaccions tàctils.

6.3.1 Rendiment de la Modalitat Híbrida

	Dits		Poms	
Acció	Mitjana	%	Mitjana	%
Manipulació	78.26	51.18	74.65	48.82
Instanciació de menús	8.52	61.44	5.35	38.56
Selecció	9.87	30.80	22.17	69.20
Inserció d'elements	5.22	13.82	32.52	86.18

Taula 21. Ús de dits/poms en la tasca d'edició de sèries amb el mode híbrid.

Com indica la Taula 21, els resultats de la modalitat d'interacció híbrida mostren que, al contrari que en la tasca de selecció d'elements (secció 5.3.1), eren majoritàries les seleccions amb poms (69.2%) i la instanciació de menús amb els dits (61.44%). A més a més, els ítems eren inserits en les sèries utilitzant, sobretot, tangibles. En promig, els usuaris feien servir 6 o 7 poms diferents per completar la tasca (M=6.48).

Tots aquests resultats confirmen l'evidència obtinguda dels vídeos: Les tècniques tangibles es comporten millor amb una dificultat de la tasca i càrrega elevades. Com que la superfície de treball esdevé abarrotada fàcilment i els elements prèviament seleccionats i inserits són candidats molt probables a se reutilitzats diverses vegades, els poms resulten avantatjosos, car són més fàcils de manipular en aquestes situacions. Els subjectes pareixien cansar-se a mesura que la tasca avançava quan havien d'arrossegar els ítems repetidament amb els dits.

7 Resultats dels Qüestionaris

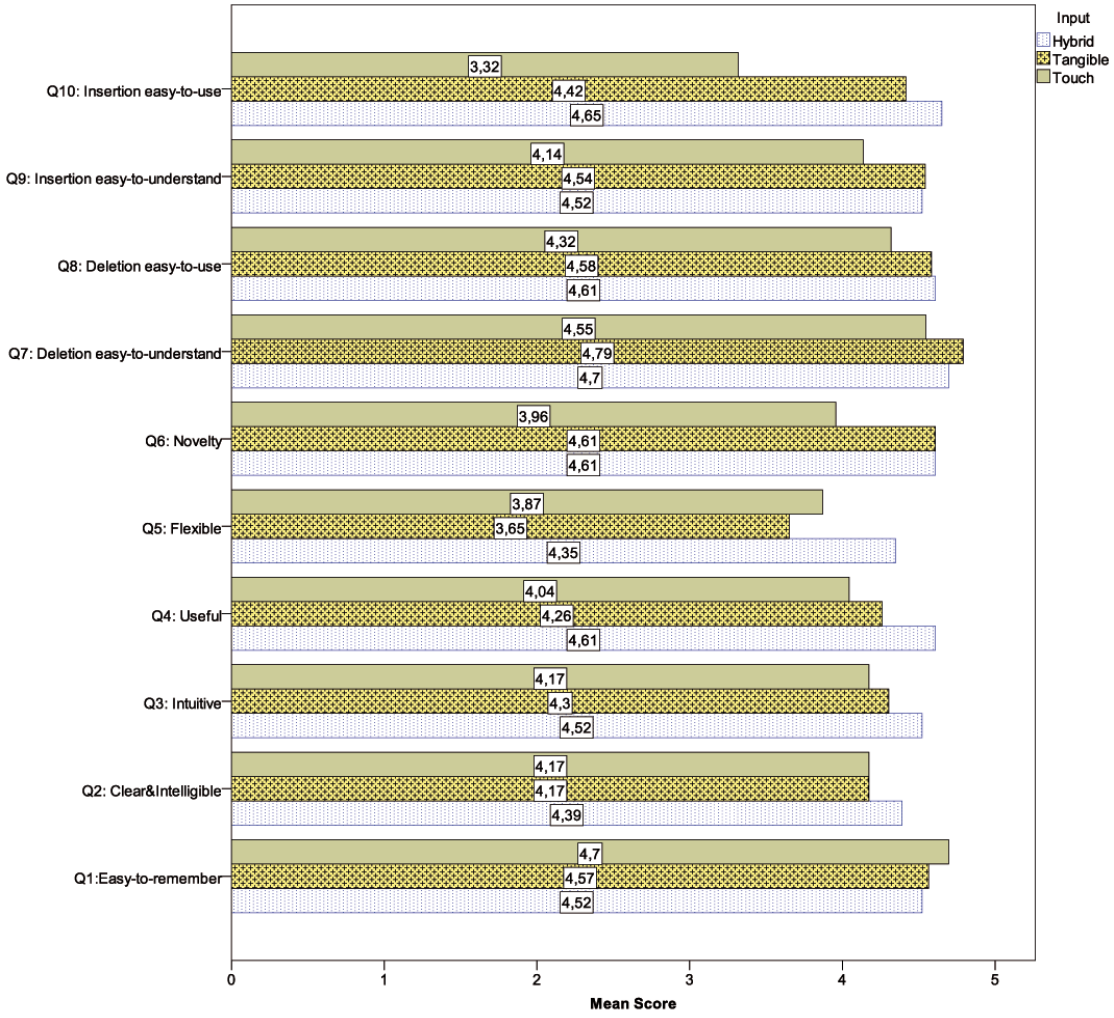


Figura 71. Resultats dels qüestionaris (escala *Likert* de cinc punts).

Al finalitzar cada tasca, es va demanar als participants que emplenaren uns qüestionaris sobre la facilitat d'ús i la utilitat del control que acabaven d'utilitzar. Les respostes eren contestades sobre una escala *Likert* de cinc punts, i versaven sobre les tasques de selecció i d'edició de sèries. Amb aquestes preguntes es pretenia comparar les tres modalitats d'interacció de TangiWheel, les quals van ser avaluades mitjançant tests per parelles Mann-Whitney. La Taula

22 conté les preguntes que van ser puntuades pels voluntaris en el qüestionari repartit, mentre que la Figura 71 mostra els resultats mitjans de les respostes.

Q	Qüestions: Considere que...
Q1	els mecanismes per a explorar les col·leccions són senzills de recordar.
Q2	la interacció és clara i intel·ligible.
Q3	el control usat és intuïtiu a l'hora de manipular-lo.
Q4	el mecanisme per a interactuar amb les col·leccions és útil.
Q5	el mecanisme per a interactuar amb les col·leccions és flexible.
Q6	la manera d'interactuar amb les col·leccions és original.
Q7	la metàfora per a l'esborrat d'elements és fàcil d'entendre.
Q8	la metàfora per a l'esborrat d'elements és fàcil d'usar.
Q9	la metàfora per a la inserció d'elements és fàcil d'entendre.
Q10	la metàfora per a la inserció d'elements és fàcil d'usar.

Taula 22. Preguntes puntuades pels participants als qüestionaris.

En general, els tres modes van ser puntuats positivament i no es detectaren problemes de disseny a partir de les respostes. Per exemple, els subjectes trobaven que les tècniques d'exploració de col·leccions eren fàcils de recordar en tots tres mètodes (Q1), sense diferències significatives entre ells. També consideraven que les tècniques d'interacció amb el control eren clares i intel·ligibles (Q2). A més, trobaren que la modalitat híbrida era la més intuïtiva (Q3). Aquestes respostes mostren que els usuaris anticiparen el comportament dels tangibles per realitzar accions naturals tals com l'adquisició, la rotació i la translació, mentre que amb la proposta purament virtual eren necessàries accions més complexes que, per tant, requerien un major esforç cognitiu. Els participants consideraren que el disseny híbrid els permetia combinar les accions més intuïtives de les altres dues modalitats.

La interacció amb les col·leccions va ser trobada útil en totes tres modalitats (Q4). L'híbrida, però, obtingué una significativa major puntuació (z=-2.896, p-valor=0.004). Aquest resultat suggereix que els usuaris perceben més útil la possibilitat de poder elegir si usar els dits o els tangibles, depenent del tipus de tasca que es porten entre mans, que el fet de que se'ls impose una o altra.

Va haver discrepàncies sobre la flexibilitat. El mode híbrid va ser considerat més flexible, mentre que el tangible, més enutjós (Q5). Les diferències entre ambdós resulten significativament altes (z=-2.625, p-valor=0.009), mentre que les diferències entre híbrid i virtual queden a prop de la rellevància (z=-1.839, p-valor=0.066). Aquests resultats suggereixen que la combinació dels estils tangible i virtual és un avanç quant a la flexibilitat.

També consideraren que tant les propostes híbrida com tangible eren igualment originals (Q6). La virtual, no obstant, va ser considerada com a menys original o nova (z=-2.410, p-valor=0.016), el que vol dir que la component d'emprar poms feia augmentar la sensació d'originalitat.

Les qüestions de la Q7 a la Q10 en la Figura 71 mostren els resultats mitjans per a preguntes relacionades amb les metàfores d'inserció i esborrat d'elements. Mentre que les tècniques involucrades en eliminacions eren considerades fàcils d'entendre i d'usar en totes les modalitats d'interacció (Q7 i Q8), es van trobar diferències significatives entre les associades a les insercions. La metàfora d'inserció en mode virtual és significativament més difícil de comprendre que la del mode tangible, segons el test sobre els resultats de Q9 (z=-2.034, p-valor=0.042). A més, l'operació d'inserció en mode virtual resulta significativament més difícil de dur a terme (Q10) que en mode híbrid (z=-4.333, p-valor=0.000) o en tangible (z=-3.675, p-valor=0.000). La metàfora associada a obtindre, alçar i dipositar un contenidor tangible per inserir un element en una secció específica dins d'una col·lecció resultà millor entesa que haver d'arrossegar ítems sobre la superfície amb els dits.

Altres preguntes addicionals específiques a la selecció en mode híbrid (H) es van dur a terme a fi d'avaluar preferències i percepcions. La Taula 23 mostra les afirmacions avaluades i indica quant d'acord estaven els participants amb les mateixes. Els resultats, en general, són positius per a la modalitat en qüestió. Primer, la capacitat dels tangibles per contindre elements ha sigut puntuada positivament (H1). Segon, els subjectes estaven lleugerament més a favor d'usar tangibles que els dits quan fóra possible (H2 i H3). Tercer, sentien que usar el control en la seua modalitat híbrida millora clarament tant l'efectivitat (H4) com la productivitat (H5) en la selecció d'ítems.

H	Qüestions	Totalment en desacord	En desacord	Neutral	D'acord	Totalment d'acord
H1	És avantatjós que els tangibles siguen capaços de contindre elements.	0 (0%)	2 (8.7%)	3 (13%)	8 (34.8%)	10 (43.5%)
H2	Preferisc usar components virtuals sempre que siga possible.	0 (0%)	5 (21.7%)	13 (56.5%)	5 (21.7%)	0 (0%)
H3	Preferisc usar tangibles sempre que siga possible.	0 (0%)	6 (26.1%)	8 (34.8%)	8 (34.8%)	1 (4.3%)
H4	TangiWheel en mode híbrid millora la meua efectivitat per a seleccionar ítems.	0 (0%)	1 (4.3%)	2 (8.7%)	6 (26.1%)	14 (60.9%)
H5	TangiWheel en mode híbrid em permet seleccionar ítems més ràpidament.	0 (0%)	1 (4.3%)	0 (0%)	8 (34.8%)	14 (60.9%)

Taula 23. Respostes a les preguntes sobre selecció en mode híbrid. Per a cada pregunta *H* i per a cada possible nivell de concordança s'inclou el nombre de participants que donaren eixa resposta i el percentatge que representen.

Al final de les sessions, es va passar un qüestionari final als participants dels experiments. Aquest contenia preguntes referents a quina de les modalitats d'interacció de TangiWheel trobaven més adequada per realitzar certes operacions, on només podien contestar amb una sola resposta. La Taula 24 mostra els resultats d'aquestes qüestions finals (F).

F	Qüestions: En general, considere que...	Híbrid	Tangible	Virtual	No importa
F1	Adquirir un TangiWheel és més fàcil en...	8 (34.8%)	14 (60.9%)	1 (4.3%)	0 (0%)
F2	Moure un TangiWheel a una posició objectiu és més fàcil en...	2 (8.7%)	16 (69.6%)	2 (8.7%)	3 (13%)
F3	Rotar un TangiWheel a una orientació objectiu és més fàcil en...	1 (4.3%)	18 (78.3%)	4 (17.4%)	0 (0%)
F4	Explorar en col·leccions és més fàcil en...	8 (34.8%)	8 (34.8%)	5 (21.7%)	2 (8.7%)
F5	Seleccionar ítems en col·leccions és més fàcil en...	7 (30.4%)	5 (21.7%)	11 (47.8%)	0 (0%)
F6	Moure un element d'una col·lecció a una localització és més fàcil en...	2 (8.7%)	19 (82.6%)	2 (8.7%)	0 (0%)
F7	Els esborrats són més fàcils en...	1 (4.3%)	6 (26.1%)	11 (47.8%)	5 (21.7%)
F8	Editar sèries és més fàcil en...	11 (47.8%)	10 (43.5%)	0 (0%)	2 (8.7%)
F9	Jo explore elements de forma més precisa en...	7 (30.4%)	10 (43.5%)	5 (21.7%)	1 (4.3%)
F10	Jo realitze el procés de selecció més ràpidament en...	16 (69.6%)	4 (17.4%)	3 (13%)	0 (0%)
F11	Editar sèries és més ràpid en...	10 (43.5%)	12 (52.2%)	0 (0%)	1 (4.3%)
F12	Manipulacions repetitives sobre un ítem és més efectiu en...	1 (4.3%)	22 (95.7%)	0 (0%)	0 (0%)

Taula 24. Respostes al qüestionari final. Per a cada pregunta *F* i per a cada possible resposta s'inclou el nombre de participants que la respongueren i el percentatge que representen.

Com mostren les respostes F1, F2 i F3 en la Taula 24, les manipulacions bàsiques eren clarament considerades com més fàcils en la proposta tangible. Els usuaris trobaven més fàcil realitzar exploracions mitjançant l'ús de poms (F4). Açò també inclou al mode híbrid, car moltes operacions d'instanciació de col·leccions en l'experiment 2 (secció 5) involucraven l'ús de tangibles.

La interacció tangible va trobar-se més senzilla que la virtual per moure un element d'una col·lecció a una posició objectiu (F6), acció necessària en operacions d'inserció. Per esborrar ítems, el mode virtual dóna millors resultats (F7). Els participants consideraven que agafar un pom només per realitzar un esborrat interrompia la seqüència d'operacions. No obstant, el procés d'edició en general va ser trobat més fàcil en la modalitat híbrida, seguida de prop de la tangible (F8). Cap usuari va elegir l'enfocament virtual, el que suggereix que, a l'hora

de dissenyar ITUs, s'ha d'evitar arrossegar elements davant de situacions on hi puga haver una alta concentració d'elements sobre la superfície. Aquesta conclusió suporta també la idea de què escenaris més generals i complexos (amb un rang d'accions diferents com ara explorar, seleccionar, arrossegar...) proveeixen millors resultats amb interaccions híbrides.

Els voluntaris sentien que el mode híbrid els permetia efectuar seleccions de forma més ràpida (F10), probablement com a conseqüència de la combinació d'explorar amb poms i seleccionar amb dits, tal i com ha estat comentant-se durant tot aquest estudi. L'híbrid va ser considerat també efectiu per editar sèries, encara que no tan ràpid com el tangible (F11), al contrari del que podria esperar-se. Finalment, la memòria dels tangibles va ser considerada una característica positiva, ja que manipulacions repetitives eren clarament considerades més efectives en mode tangible (F12).

Capítol 5.
Conclusions i Treball Futur

En aquest treball s'ha presentat TangiWheel, un control per a l'exploració d'elements sobre superfícies interactives. En concret, es presenta com un control idoni per a un tipus d'aplicació en la que la navegació per col·leccions siga una característica essencial: les aplicacions constructives, on la tasca fonamental consisteix en muntar un objecte complex a partir de certs components més menuts organitzats en col·leccions. D'ací s'extreu un objectiu principal d'aquest projecte: dissenyar un menú que suporte de forma eficient operacions d'exploració i de selecció d'elements.

A més, aquestes aplicacions per a realitzar construccions són idònies per a entorns col·laboratius, on diverses persones al voltant d'una superfície treballen conjuntament per realitzar un muntatge. Açò porta a un altre objectiu: optimitzar el disseny de TangiWheel per a entorns multiusuari. Concretament, proporcionar equitat i simultaneïtat a tots els usuaris, permetre la compartició dels controls i proporcionar una orientació que no limite la visibilitat depenent de la localització de l'usuari.

Tot i motivar aquesta proposta amb un tipus d'aplicació concret, TangiWheel s'ha dissenyat de forma genèrica aplicable a nombrosos àmbits i que, a més, és altament configurable des del punt de vista del desenvolupador. De cara a l'usuari, també permet una certa personalització, ja que el control pot ser manipulat tant amb els dits com emprant uns objectes tangibles als que hem anomenat "poms".

La plataforma tecnològica escollida per implantar TangiWheel ha sigut la Microsoft Surface 1.0 (coneguda com Microsoft PixelSense des de juny de 2012), que proporciona informació sobre diversos contactes realitzats sobre la aquesta al mateix temps i dóna suport per a l'ús d'objectes físics en les interaccions. Sobre la Surface es pot treballar amb certs controls predefinits (botons, panels, llistes...) en WPF, però el disseny d'un nou control requeria treballar a més baix nivell i un esforç important en la part de la gestió de la informació d'entrada. Per a això s'ha creat una capa d'abstracció que manipula aquesta informació i la presenta de forma més còmoda de cara a treballar amb controls.

Per demostrar les bondats de TangiWheel comentades més amunt, s'han realitzat tres experiments enfocats en l'estudi de l'adquisició del menú, les manipulacions bàsiques per realitzar-hi translacions i rotacions, l'exploració de les col·leccions i la selecció d'elements. S'han comparat els tres modes d'interacció

possibles amb el menú per manipular-lo: purament amb els dits, emprant només poms i una combinació lliure d'ambdós.

L'objectiu del primer experiment era posicionar i rotar una instància de TangiWheel en una posició i orientació predefinides. S'ha conclòs que les manipulacions d'aquest estil realitzades amb els tangibles són més ràpides que les tàctils, deguts, segurament, a què l'ús d'objectes per a aquest tipus d'operacions resulta més natural i semblant a les accions diàries.

L'experiment 2 consistia en realitzar una sèrie de seleccions d'elements, resultant necessària una exploració prèvia. Els resultats obtinguts mostren que el gest circular amb els dits per realitzar exploracions és més efectiu que l'ús de tangibles, sobretot degut a què no es produeixen interrupcions, permetent una interacció contínua. Per aquesta mateixa raó es podria pensar que un gest lineal per explorar col·leccions grans seria menys efectiu també, però seria necessari realitzar experiments addicionals per comprovar-ho. No obstant, com s'apunta en el tercer experiment, si es duen a terme seleccions recurrents i és freqüent moure elements de les col·leccions a altres llocs de la superfície, així com si es té un nombre elevat de controls sobre la mateixa, llavors els elements físics són més avantatjosos en aquest aspecte que l'ús dels dits.

Per tant, els experiments mostren que no hi ha una modalitat d'interacció preferible per a tots els casos, ja que depèn de la tasca que s'estiga realitzant en cada moment. Llavors, una modalitat híbrida d'interacció com la que es presenta en aquesta dissertació ofereix un avantatge respecte a altres controls d'exploració existents.

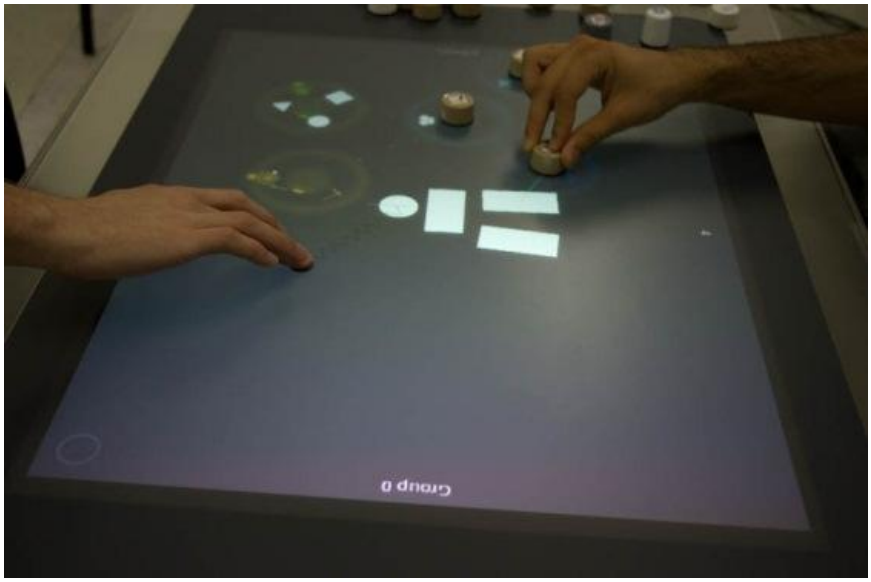


Figura 72. Exemple d'aplicació constructiva fent servir TangiWheels.

El projecte de TangiWheel, juntament amb la capa intermèdia de gestió de contactes, s'ofereix com un *toolkit* d'interacció en el context d'un projecte

d'investigació en el grup ISSI[4] de la Universitat Politècnica de València. Altres membres d'aquest grup l'han emprat satisfactòriament per construir interfícies d'usuari on l'objectiu és l'edició d'estructures, d'entitats i de regles de comportament (vore Figura 72). Aquestes interfícies s'estan usant actualment en una plataforma de simulació de jocs sobre una Microsoft Surface.

Quant al treball futur, es pretén realitzar més experiments sobre els avantatges que presenta TangiWheel front a altres controls d'exploració existents i sobre la seua idoneïtat per a entorns multiusuari. També s'estudiarà la possibilitat d'extendre el comportament i funcionalitat d'aquest menú per donar suport a característiques que el facen més genèric i potent, com s'ha comentat al llarg del document. Per exemple, afegir la possibilitat de previsualitzar un submenú abans d'obrir-lo per explorar-ne el contingut; mostrar d'alguna manera els elements accedits més recentment (que, segons l'aplicació, poden ser candidats a ser tornats a seleccionar), o bé ordenar dinàmicament la col·lecció per a què aquests apareguen al principi de la llista; i solucionar el problema de l'oclusió de la regió visible amb la mà, entre altres.

Donat que s'ha creat una capa de gestió de contactes per a la Surface que permet la creació senzilla de controls, en un futur es dissenyaran i implementaran més controls que completen el *toolkit* per a la creació d'interfícies tangibles d'usuari sobre superfícies interactives.

Altra línia de treball futur és afegir als diferents controls una "sensibilitat" al context, de forma que la seua aparença i comportament s'adapten dinàmicament a l'entorn. Per exemple, una gestió més intel·ligent de la distribució espacial dels controls, basant-se en l'agrupació d'aquestos entorn a un usuari concret.

A continuació s'exposa una llista de publicacions científiques que s'han derivat del present treball [7, 8, 6, 15]:

- Fernando García, Alejandro Catalá, Javier Jaén, Jose Antonio Mocholí. Una interfaz tangible para la navegación genérica de estructuras de colección. *Actas de las Jornadas de Ingeniería del Software y Bases de Datos, "JISBD 2011"*, pp. 1031-1044, 2011 (ISBN: 978-84-9749-486-1)

- Alejandro Catalá, Fernando García, Javier Jaén, Jose Antonio Mocholí. TangiWheel: A widget for manipulating collections on tabletop displays supporting hybrid input modality. *Journal of Computer Science and Technology*. 27(4):1 (JCR, IF: 0.656).

- Alejandro Catalá, Fernando García, Jose Azorín, Javier Jaén, Jose Antonio Mocholí. Exploring direct communication and manipulation on interactive surfaces to foster novelty in a creative learning environment.

[4] http://issi.dsic.upv.es/

Proceedings of the International Conference on Virtual Learning (ICVL'11). Distinguised with the *Excellence Award*.

- Alejandro Catalá, Fernando García, Jose Azorín, Javier Jaén, Jose Antonio Mocholí. Exploring direct communication and manipulation on interactive surfaces to foster novelty in a creative learning environment. *International Journal of Computer Science Research and Application* 2012, Vol. 02, Issue. 01 (Special Issue. Extended Selected Papers ICVL2011), pp. 15-24. ISSN 2012-9572 (20% acceptance rate).

Bibliografia

[1] Johnny Accot and Shumin Zhai. Beyond fitts' law: models for trajectory-based hci tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '97, pages 295–302, New York, NY, USA, 1997. ACM.

[2] Rosemary A. Bailey. *Design of Comparative Experiments*, chapter 6. Row-Column designs. Cambridge Series in Statistical and Probabilistic Mathematics (No. 25). Cambridge University Press, April 2008.

[3] Rosemary A. Bailey. *Design of Comparative Experiments*, chapter 9. More about Latin squares. Cambridge Series in Statistical and Probabilistic Mathematics (No. 25). Cambridge University Press, April 2008.

[4] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. Wave menus: Improving the novice mode of hierarchical marking menus. In *INTERACT 2007*, pages 475–488, 2007.

[5] Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '88, pages 95–100, New York, NY, USA, 1988. ACM.

[6] Alejandro Catala, Fernando Garcia-Sanjuan, Jose Azorin, Javier Jaen, and Jose A. Mocholi. Exploring direct communication and manipulation on interactive surfaces to foster novelty in a creative learning environment. In *Proceedings of the International Conference on Virtual Learning*, ICVL'11, 2011. Distinguished with the *Excellence Award*.

[7] Alejandro Catala, Fernando Garcia-Sanjuan, Jose Azorin, Javier Jaen, and Jose A. Mocholi. Exploring direct communication and manipulation on interactive surfaces to foster novelty in a creative learning environment. *International Journal of Computer Science Research and Application*, Vol. 02, Issue. 01:15–24, 2012. Special Issue. Extended Selected Papers ICVL2011 (20% acceptance rate).

[8] Alejandro Catala, Fernando Garcia-Sanjuan, Javier Jaen, and Jose A. Mocholi. Tangiwheel: A widget for manipulating collections on tabletop displays supporting hybrid input modality. *Journal of Computer Science and Technology*. (to appear).

[9] Microsoft Corporation. Microsoft pixelsense. WWW Page, June 2012. http://www.microsoft.com/en-us/pixelsense/default.aspx.

[10] Kenneth P. Fishkin. A taxonomy for and analysis of tangible interfaces. *Personal Ubiquitous Computing*, 8(5):347–358, September 2004.

[11] George W. Fitzmaurice and William Buxton. An empirical evaluation of graspable user interfaces: towards specialized, space-multiplexed input. In *CHI '97*, pages 43–50, 1997.

[12] George W. Fitzmaurice, Hiroshi Ishii, and William Buxton. Bricks: Laying the foundations for graspable user interfaces. In *CHI '95*, pages 442–449, 1995.

[13] Herbert Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers*, EC-10, Issue: 2:260–268, June 1961.

[14] Daniel Gallardo and Sergi Jordà. Tangible jukebox: back to palpable music. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, TEI '10, pages 199–202, New York, NY, USA, 2010. ACM.

[15] Fernando Garcia-Sanjuan, Alejandro Catala, Javier Jaen, and Jose A. Mocholi. Una interfaz tangible para la navegación genérica de estructuras de colección. In *Actas de las Jornadas de Ingeniería del Software y Bases de Datos*, JISBD 2011, pages 1031–1044, 2011.

[16] Mark Hancock, Otmar Hilliges, Christopher Collins, Dominikus Baur, and Sheelagh Carpendale. Exploring tangible and direct touch interfaces for manipulating 2d and 3d information on a digital table. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 77–84, New York, NY, USA, 2009. ACM.

[17] Tobias Hesselmann, Stefan Flöring, and Marwin Schmitt. Stacked half-pie menus: navigating nested menus on interactive tabletops. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 173–180, New York, NY, USA, 2009. ACM.

[18] Otmar Hilliges, Dominikus Baur, and Andreas Butz. Photohelix: Browsing, sorting and sharing digital photo collections. In *Horizontal Interactive Human-Computer Systems (TABLETOP '07)*, pages 87–94, 2007.

[19] Don Hopkins. Directional selection is easy as pie menus! *The Usenix Association Newsletter, 12, 5*, 1987.

[20] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *CHI '97*, pages 234–241, 1997.

[21] John I. Kiger. The depth/breadth trade-off in the design of menu-driven user interfaces. *International Journal of Man-Machine Studies*, 20(2):201–213, March 1984.

[22] Gordon Kurtenbach and William Buxton. The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 482–487, New York, NY, USA, 1993. ACM.

[23] G. Julian Lepinski, Tovi Grossman, and George Fitzmaurice. The design and evaluation of multitouch marking menus. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 2233–2242, New York, NY, USA, 2010. ACM.

[24] Aurélien Lucchi, Patrick Jermann, Guillaume Zufferey, and Pierre Dillenbourg. An empirical evaluation of touch and tangible interfaces for tabletop displays. In *TEI 2010*, pages 177–184, 2010.

[25] Michael Maydak, Robert Stromer, Harry A. Mackay, and Lawrence T. Stoddard. Stimulus classes in matching to sample and sequence production: The emergence of numeric relations. *Research in Developmental Disabilities*, 16-3:179–204, May-June 1995.

[26] MSDN. Microsoft surface user experience guidelines. WWW Page, May 2011. http://msdn.microsoft.com/en-us/library/ff318692.aspx.

[27] MSDN. Hlsl. WWW Page, June 2012. http://msdn.microsoft.com/en-us/library/bb509561%28v=vs.85%29.

[28] MSDN. Windows presentation foundation. WWW Page, June 2012. http://msdn.microsoft.com/en-en/library/ms754130.aspx.

[29] MSDN. Xna developer center. WWW Page, June 2012. http://msdn.microsoft.com/en-us/centrum-xna.aspx.

[30] Kent L. Norman. *The Psychology of Menu selection: Designing Cognitive Control at the Human/Computer Interface*. Ablex Publishing Corporation, 1991.

[31] James Patten, Ben Recht, and Hiroshi Ishii. Interaction techniques for musical performance with tabletop tangible interfaces. In *ACE '06*, 2006.

[32] Chia Shen, Mark S. Hancock, Clifton Forlines, and Frédéric D. Vernier. Cor2ds: Context-rooted rotatable draggables for tabletop interaction. In *CHI 2005*, pages 1781–1784, 2005.

[33] Hyunjoo Song, Bohyoung Kim, Bongshin Lee, and Jinwook Seo. A comparative evaluation on tree visualization methods for hierarchical structures with large fan-outs. In *Proceedings of the 28th international conference on*

Human factors in computing systems, CHI '10, pages 223–232, New York, NY, USA, 2010. ACM.

[34] Microsoft TechNet. Physical features of a microsoft surface unit. WWW Page, June 2012. technet.microsoft.com/en-us/library/ee692114(v=surface.10).aspx.

[35] Philip Tuddenham, David Kirk, and Shahram Izadi. Graspables revisited: Multi-touch vs. tangible input for tabletop displays in acquisition and manipulation tasks. In *CHI '10*, pages 2223–2232, 2010.

[36] Brygg Ullmer and Hiroshi Ishii. The metadesk: Models and prototypes for tangible user interfaces. In *UIST'97*, pages 223–232, 1997.

[37] Brygg Ullmer, Hiroshi Ishii, and Dylan Glas. mediablocks: Physical containers, transports, and controls for online media. In *SIGGRAPH '98*, pages 379–386, 1998.

[38] Malte Weiss, Julie Wagner, Yvonne Jansen, and Roger Jennings. Slap widgets: Bridging the gap between virtual and physical controls on tabletops. In *CHI 2009*, pages 481–490, 2009.

[39] NUI Group Community Wiki. Diffused illumination (di). WWW Page, June 2012. http://wiki.nuigroup.com/Diffused_Illumination.

[40] Shengdong Zhao, Maneesh Agrawala, and Ken Hinckley. Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 1077–1086, New York, NY, USA, 2006. ACM.