Brainstorming via interactive whiteboards

Bob Dijck

promotor :
Prof. dr. Chris RAYMAEKERS

Eindverhandeling voorgedragen tot het bekomen van de graad
Master in de informatica human-computer interaction

Voorwoord

Voor u ligt een werkstuk dat de afsluiting vormt van een belangrijk hoofdstuk uit mijn leven. Deze thesis is geschreven als finaal onderdeel van de opleiding Informatica (optie Human-Computer Interaction) aan de transnationale universiteit Limburg. In deze thesis heb ik de interactie met digitale whiteboards onderzocht.

Voor deze thesis ben ik vele mensen dank verschuldigd. Mijn dankbaarheid gaat uit naar mijn thesispromotor prof. dr. Chris Raymaekers en naar mijn thesisbegeleiders Peter Vandoren en Maarten Cardinaels voor al hun hulp, suggesties en feedback. Als werkende student ben ik voor hen wellicht niet de makkelijkste student geweest om te begeleiden. Voor de geboden flexibiliteit ben ik hen dan ook zeer dankbaar. Hartelijk dank.

Voor de usability-test van de applicatie die ik in het kader van deze thesis ontwikkeld heb, kon ik op de hulp rekenen van enkele collega-studenten en medewerkers van het Expertisecentrum voor Digitale Media die ik hierbij nogmaals wil danken: Lode Vanacken, Tim Tutenel, Vicky Pagnaer en Erwin Cuppens.

Tijdens de voorbije jaren heb ik enorm veel steun gehad van Anneke, mijn vriendin die ondertussen mijn vrouw is geworden. Haar ben ik enorm dankbaar: niet alleen voor het nalezen van teksten, maar vooral voor de morele ondersteuning. Tot slot zijn ook mijn ouders me steeds door dik en dun blijven steunen en stonden ze altijd onvoorwaardelijk achter me. Ook aan hen: ongelooflijk bedankt.

Bob Dijck, 19 augustus 2007

Inhoudsopgave

1 Inleiding

1.1 Doel

Deze thesis heeft als doel de interactie via interactieve whiteboards te onderzoeken en een applicatie te ontwikkelen die gebruik maakt van de mogelijkheden van dergelijk whiteboard. Interactieve whiteboards bieden met hun aanraakgevoelige schermen namelijk mogelijkheden om op een andere manier met een whiteboard om te gaan dan het klassieke whiteboard.

1.2 Aanpak

De uitwerking van de thesis is in twee gedeelten verlopen: een deel literatuurstudie en een deel implementatie. De opbouw van deze thesistekst volgt deze tweedeling en bestaat dus uit twee grote onderdelen. Het eerste gedeelte, hoofdstukken 2 tot en met 5, wordt gevormd door een literatuurstudie over interactie met digitale whiteboards. Het tweede gedeelte beslaat hoofdstukken 6 en 7 en bestaat uit een implementatie: een applicatie waarin verschillende interactietechnieken aan bod komen.

Voor de literatuurstudie ben ik kunnen vertrekken van een aantal aangereikte papers op basis waarvan ik op zoek ben gegaan naar andere relevante literatuur over interactieve whiteboards en de interactie hiermee. Op basis van de gevonden literatuur heb ik vervolgens de studie uitgewerkt die in de vermelde hoofdstukken staat beschreven.

De implementatie is gebouwd in C# en maakt voor de grafische weergave gebruik van GDI+. Het is een applicatie om het interieur van een kamer in te richten en is bedoeld als een platform voor diverse technieken die de interactie met een interactief whiteboard ondersteunen.

1.3 Overzicht

In hoofdstuk 2 komt aan bod wat een interactief whiteboard precies is en hoe dergelijk whiteboard werkt: de manier waarop het contactpunt tussen de vinger van de gebruiker en het bord bepaald wordt en hoe het beeld geprojecteerd wordt. Ook in dit hoofdstuk wordt de rol beschreven van een interactief whiteboard binnen een grotere omgeving die bedoeld is om collaboratie te ondersteunen, en welke andere apparaten hierbij betrokken kunnen zijn. Verder komen ook de verschillende mogelijke vormen van collaboratie aan bod, alsook de verschillende soorten van toepassingen waarvoor een interactief whiteboard gebruikt kan worden.

Hoofdstuk 3 behandelt enkele bestaande interactieve omgevingen die ontwikkeld zijn door wetenschappers en waarin het interactief whiteboard een belangrijke rol speelt. De volgende omgevingen zullen in dit hoofdstuk de revue passeren: BlueBoard, Flatland, Tivoli, Dynamo, BrightBoard, iRoom, ContextWall en Kimura. Al deze

omgevingen maken gebruik van een interactief whiteboard en zijn interessante platformen waarin de interactie met digitale whiteboards onderzocht is.

Deze interactie vormt het onderwerp van hoofdstuk 4 waarin de verschillen tussen een interactief whiteboard en een klassiek desktopscherm aan bod komen. Verder wordt in dit hoofdstuk ook nog het probleem van *floor control* behandeld alsook de alternatieve menuvormen die voor interactieve whiteboards ontwikkeld zijn.

Dat een interactief whiteboard zijn eigen specifieke problemen kent, is het onderwerp van hoofdstuk 5. Ter vergelijking en voor een beter begrip is eerst een paragraaf opgenomen over de problemen van klassieke whiteboards. Daarna is het de beurt aan de problemen van interactieve whiteboards. Als belangrijkste problemen worden achtereenvolgens de schermgrootte, het invoeren van tekst, de ondersteuning voor meerdere gebruikers en de drempelvrees van gebruikers behandeld.

Hoofdstuk 6 vormt zoals aangehaald het eerste hoofdstuk van het tweede deel van de thesis, de implementatie. Hierin wordt in een eerste paragraaf de applicatie, een programma om het interieur van een kamer te ontwerpen, van naderbij bekeken. In de volgende paragrafen komen vervolgens de interactie met het FlowMenu aan bod en de verschillende interactietechnieken die voor de applicatie zijn uitgewerkt.

In hoofdstuk 7 komt de usability-test van de applicatie aan bod en ook welke resultaten deze test heeft opgeleverd.

Het slot wordt gevormd door hoofdstuk 8, waarin de algemene conclusies van deze thesis zijn opgenomen.

2 Interactieve whiteboards

In dit hoofdstuk ga ik in op wat interactieve whiteboards precies zijn en welke verschillende technologieën er bij interactieve whiteboards gebruikt kunnen worden. In de paragraaf over dit technologische aspect wordt een overzicht gegeven van de verschillende manieren waarop het contactpunt (dit is het punt waar de pen van de gebruiker het bord raakt) bepaald kan worden alsook de twee mogelijkheden voor de projectie van het beeld.

Verder zal het in dit hoofdstuk ook nog gaan over de verschillende andere apparaten -zoals bijvoorbeeld PDA's- die bij omgevingen van interactieve whiteboards een toegevoegde waarde kunnen betekenen. Vervolgens komt het begrip collaboratie aan bod: dit is immers een essentieel begrip wanneer we het hebben over interactie met meerdere personen aan een interactief whiteboard.

Daarna zal ik ook getoond worden dat de interactie met een groot oppervlak zich niet hoeft te beperken tot interactieve whiteboards die tegen de wand hangen, maar ook mogelijk is met interactieve tafels (die dus een horizontale oriëntatie hebben). Ook zal onderzoek aangevoerd worden dat gebeurd is betreffende de invloed van deze oriëntatie. Tot slot van dit hoofdstuk wordt een classificatie gemaakt van schermen met een grote omvang op basis van de toepassing.

2.1 Omschrijving

Een interactief whiteboard is een apparaat dat net als een klassiek computerscherm een tweedimensionaal beeld toont en waarop gebruikers kunnen tekenen en schrijven met een speciale stylus (pen). Een interactief whiteboard is bovendien een stuk groter dan een conventioneel computerscherm en wordt doorgaans niet op een bureau voor een gebruiker geplaatst, maar aan een wand gehangen om door meerdere gebruikers tegelijk gebruikt en/of gezien te kunnen worden.

Bovendien biedt een interactief whiteboard naast de beelduitvoer ook invoermogelijkheden: wanneer een gebruiker het scherm aanraakt met de pen of met de vinger (afhankelijk van de gebruikte technologie, zie paragraaf 2.2), geeft het whiteboard deze coördinaten door aan de applicatie die instaat voor de afhandeling van deze interactie.

Interactieve whiteboards zijn ondertussen gemeengoed geworden en worden onder andere gebruikt in bedrijven voor presentaties, vergaderingen en brainstormingsessies, en ook in het onderwijs als instructief en collaboratief instrument.

2.2 Technologie

2.2.1 Bepaling van het contactpunt met het bord

Het interactief whiteboard is verbonden met een computer door middel van een klassieke seriële verbinding of USB, maar ook een draadloze verbinding bestaat. Het whiteboard vertaalt het contact op het scherm doorgaans in muis*clicks*. De manier waarop dit contactpunt op het scherm bepaald wordt, kan op verschillende manieren gebeuren. In de volgende paragrafen worden de meest voorkomende technologieën beschreven om deze positie te bepalen: analoog-resistief, elektromagnetisch, laser-gebaseerd, of ultrasoon.

2.2.1.1 Analoog-resistief

Bij deze techniek liggen er twee geleidende folies over het scherm, met daartussen een zeer dun laagje lucht. Wanneer een gebruiker het scherm op een bepaalde plaats aanraakt, wordt het laagje lucht op dat contactpunt dun genoeg opdat de twee geleidende folies contact met elkaar kunnen maken. Hierdoor verandert de elektrische weerstand van de beide folies en kan op basis van deze verandering het contactpunt bepaald worden, wat door het systeem vertaald wordt in een X- en een Y-coördinaat.

Het aanraakoppervlak is bij deze technologie tamelijk zacht en kan men met een vinger, marker of stylus op het whiteboard schrijven of tekenen. Tijdens het schrijven drukt men immers de beide folies tegen elkaar aan waardoor de coördinaten bepaald kunnen worden. Een nadeel van deze techniek is dat de gebruiker zijn hand niet op het oppervlak kan laten rusten want dit wordt dan door het systeem geïnterpreteerd als een contact.

2.2.1.2 Elektromagnetisch

Achter het schrijfoppervlak van het interactief whiteboard bevindt zich een raster van elektrische draadjes die reageren op een spoel die in de stylus zit ingewerkt. Op deze manier bepaalt het systeem vervolgens de coördinaten. Er bestaan voor deze technologie twee soorten styli: een actieve variant die stroom nodig heeft van een interne batterij of via een kabel van het bord (wat uiteraard niet zo praktisch werkt wegens het snoer), en een passieve variant die zonder stroom werkt.

Bij deze technologie is het aanraakoppervlak harder dan bij analoog-resistieve systemen waardoor het robuuster is. Bovendien is dit soort van systeem aangenamer voor de gebruiker omdat deze zijn hand kan laten rusten op het bord (zonder dat dit door het systeem opgevat wordt als contact) en biedt deze technologie een grotere nauwkeurigheid. Tot slot is het scala van invoergegevens van deze technologie (in tegenstelling tot de analoog-resistieve) ruimer: ook *hover* (detectie wanneer de pen het aanraakoppervlak niet raakt, maar zich op een kleine afstand boven dit oppervlak bevindt) wordt ondersteund. Interactie middels de vingers is echter niet mogelijk.

2.2.1.3 Lasergebaseerd

In de linker- en rechterbovenhoek van het interactief whiteboard zijn bij deze technologie infrarode lasers gemonteerd. Deze lasers tasten met hun straal voortdurend het ganse oppervlak van het whiteboard af met behulp van draaiende spiegels. Op de stylus (of marker) zijn reflectoren geplaatst die deze stralen terugkaatsen en op basis daarvan kan het systeem de positie van de stylus of marker bepalen door middel van triangulatie.

Het aanraakoppervlak is bij deze technologie redelijk hard waardoor deze systemen een tamelijk lange levensduur hebben. Voor de input kan een passieve marker gebruikt worden, maar deze moet dan wel voorzien zijn van de nodige reflectoren om de laserstralen te kunnen weerkaatsen. Interactie middels de vingers is ook bij deze technologie niet mogelijk.

2.2.1.4 Ultrasoon

Net als bij de lasergebaseerde techniek, wordt ook bij deze techniek op basis van triangulatie van signalen de positie van een object (de stylus dus) bepaald. Bij de ultrasone techniek gaat het om auditieve signalen en is een actieve stylus nodig: de energie die de batterij levert, creëert een ultrasoon signaal. Dit ultrasone signaal wordt opgevangen door kleine microfoontjes die op het whiteboard geplaatst zijn.

Op basis van het verschil in tijdsduur dat beide signalen onderweg zijn naar de microfoontjes wordt vervolgens de precieze locatie van de stylus berekend. Deze technologie vertoont dus nogal wat overeenkomsten met de lasergebaseerde technologie (positiebepaling door gebruik te maken van signalen, zij het optische of auditieve), en kent ook dezelfde voor- en nadelen als die technologie.

2.2.2 Projectie van het beeld

Naast het verwerven en verwerken van invoer, levert een interactief whiteboard zoals al aangehaald ook uitvoer: een tweedimensionaal beeld. Dit beeld (bijvoorbeeld een presentatie) wordt aangeleverd door de video-uitgang van het computersysteem dat met het whiteboard is verbonden (veelal een gewone desktop computer), maar moet nog op het scherm kunnen komen. Dit kan op de volgende manieren: projectie van vòòr het scherm (*front projection*) of projectie van achter het scherm (*rear projection*).

Voor de volledigheid dient in dit verband nog vermeld te worden dat er ook interactieve whiteboards bestaan waarbij er geen projectiesysteem nodig is: het gaat dan om grote plasma- of LCD-schermen die voorzien zijn van een extra laag bovenop het beeldscherm die aanraakgevoelig is om de functionaliteit van aanraking (invoer) te kunnen bieden.

2.2.2.1 Front projection

Bij projectie die gebeurt vóór het scherm (*front projection*, zie Figuur 2.1) wordt het beeld op het whiteboard geprojecteerd door een video projector. Het nadeel bij *front projection* is dat de persoon die voor het bord staat een deel van het beeld doet verdwijnen door zijn schaduw. Dit kan -voor een gedeelte- verholpen worden door de arm wat meer te strekken, of een langer, dun *pointing device* te gebruiken, maar het blijft een groot nadeel. Een ander nadeel is dat iemand die aan het whiteboard een presentatie geeft, te kampen heeft met het projectielicht dat in zijn ogen schijnt. Bovendien is de projector een apart te monteren en transporteren apparaat. Een voordeel ten opzichte van de andere projectiemethodes is dat het whiteboard zelf tamelijk dun blijft.

Figuur 2.1 Front projection [36]

2.2.2.2 Rear projection

Bij *rear projection* wordt het weer te geven beeld van achteren op het whiteboard geprojecteerd. Op deze manier kunnen er geen schaduwen op het whiteboard vallen van personen die voor het whiteboard staan. Een ander voordeel van *rear projection* tijdens bijvoorbeeld presentaties is dat er geen licht in de ogen kan schijnen van de persoon aan het bord. Enkele nadelen van deze projectiemethode zijn dan weer dat het geheel van projector en whiteboard samen nogal groot en zwaar is, dat het systeem niet zo draagbaar is als een systeem met *front projection*, en dat de prijs doorgaans hoger ligt dan systemen met *front projection*.

Figuur 2.2 Rear projection [36]

2.3 Betrokken apparaten

Sommige werkomgevingen waarin interactieve whiteboards een rol spelen beperken zich uitsluitend tot het whiteboard zelf (bijvoorbeeld *BlueBoard*, zie paragraaf 3.1 en [2]), terwijl andere omgevingen ook vele andere apparaten bij de omgeving betrekken zoals laptops, desktop computers, PDA's, camera's, microfoons, enz.

Wanneer desktop computers bij de omgeving betrokken worden, verandert vaak de rol van het interactief whiteboard: het whiteboard is dan niet langer het primaire interactiemiddel, maar verschuift een beetje naar de achtergrond en genereert dan voornamelijk uitvoer, zoals bijvoorbeeld bij *Kimura* waar de interactieve whiteboards gebruikt worden voor de weergave van zogenaamde montages (zie paragraaf 3.8, [3] en [4]). Het blijft bij *Kimura* echter meestal ook mogelijk om rechtstreeks met het interactief whiteboard te interageren, bijvoorbeeld om de montages (zie paragraaf 3.8) van commentaar te voorzien en anders te organiseren.

Ook een draagbare computer is vaak betrokken bij interactieve whiteboard omgevingen. Een laptop kan de rol vervullen van informatiedrager: de gegevens worden dan van de laptop in het systeem gebracht of omgekeerd en het is de interactieve omgeving, niet de laptop, waarmee men vervolgens de gegevens kan manipuleren. Een laptop kan daarnaast ook gebruikt als een interactiemiddel en

aldus onderdeel worden van de interactietechnologie van de omgeving. Bij *iRoom* bijvoorbeeld kunnen vensters van en naar de laptop versleept worden en kunnen laptops fungeren als invoer- en controlemedium voor de ganse omgeving (zie paragraaf 3.6 en [5]).

Los van de situaties waarbij het interactief whiteboard steunt op optische of auditieve signalen om het contactpunt te bepalen (zie paragraaf 2.2.1), bestaan er omgevingen waarbij video en audio als extra medium gebruikt worden. Zo gebruikt het *BrightBoard* bijvoorbeeld het medium video als invoer bij het herkennen van bepaalde markers (zie paragraaf 3.5 en [7]): een camera registreert de speciale markeringen van de gebruiker op het whiteboard en *BrightBoard* zorgt voor de verdere analyse en afhandeling. Auditieve signalen worden ook gebruikt in interactieve omgevingen: gebruikers kunnen dan geluidsopnames maken zoals bijvoorbeeld een gesproken notitie.

2.4 Synchrone en asynchrone collaboratie

Bij asynchrone collaboratie gaat het voornamelijk om het uitwisselen van informatie waarbij de deelnemers de informatie op verschillende tijdstippen tot zich nemen. Voorbeelden hiervan zijn e-mail en nieuwsgroepen. Iemand verstuurt op een bepaald moment een boodschap, maar de geadresseerden lezen dit bericht veelal pas later, en in het geval van meerdere geadresseerden zullen zij het verstuurde bericht niet allemaal op hetzelfde tijdstip lezen, maar wanneer zij zelf kiezen dit te doen.

Asynchrone collaboratie heeft als belangrijkste voordelen vooral het gemak en de flexibiliteit: de deelnemers kiezen zelf wanneer zij de informatie willen en kunnen lezen. Maar net dit gegeven is ook een groot nadeel: het kan even duren vooraleer alle deelnemers het bericht gelezen hebben, en de gebruiker heeft weinig of geen garantie dat dit op een gegeven ogenblik gebeurd is, waardoor deze vorm van collaboratie vooral geschikt is voor het delen van minder belangrijke gegevens die geen onmiddellijke reactie vragen.

Voor taken die daarentegen wel een snelle reactie vragen is synchrone collaboratie meer aangewezen. Bij synchrone collaboratie delen twee of meerdere personen informatie op hetzelfde tijdstip. Voorbeelden hiervan zijn een *conference call* of een *face-to-face* vergadering. Zoals al vermeld heeft dit als voordeel dat er nagenoeg geen tijd verstrijkt tussen het verzenden en het ontvangen van de informatie, wat de interactiviteit in sterke mate bevordert: er kan onmiddellijk op verzonden informatie gereageerd worden.

Interactieve whiteboards kunnen zowel synchrone als asynchrone collaboratie ondersteunen, doch synchrone collaboratie komt veruit het vaakst voor omwille van de interactiviteitsgraad van typische activiteiten voor een whiteboard. Zo kan bijvoorbeeld het *BlueBoard* (zie paragraaf 3.1 en [2]) gebruikt worden om met meerdere personen een schets uit te werken. In deze tekst ligt de nadruk op de synchrone collaboratie.

Bij asynchrone applicaties voor interactieve whiteboards gaat het dus om systemen die als hoofddoel hebben asynchrone collaboratie mogelijk maken, d.i. samenwerking waarbij de deelnemers niet simultaan aanwezig hoeven te zijn. Dergelijke systemen lijken op digitale *bulletin board systems* en kiosken, waarbij echter niet alleen informatie ge*pushed* wordt, maar waar een gebruiker ook interactief informatie kan opvragen zoals bijvoorbeeld zijn agenda raadplegen of een e-mail sturen.

Een mooi voorbeeld van een dergelijk asynchroon systeem dat ook gebruik maakt van een interactief whiteboard is het prototype dat ontwikkeld is door D. Vogel [8]. Zij ontwikkelden een interactieve *public display* waarbij gebruikers via gebaren toegang kunnen krijgen tot zowel publieke als persoonlijke informatie.

2.5 Collaboratie op dezelfde plaats of vanop afstand

Wanneer we collaboratie typeren aan de hand van het locatieperspectief (ruimtelijke locatie van de collaboratie), dan stellen we vast dat collaboratie zowel gedistribueerd als *co-located* kan verlopen. Bij gedistribueerde collaboratie bevinden de verschillende deelnemers zich niet allemaal in dezelfde ruimte. Hierbij kunnen we bijvoorbeeld denken aan een *conference call* tussen verschillende personen: de deelnemers bevinden zich niet op dezelfde, maar op verschillende plaatsen.

Bij *co-located* collaboratie zijn de deelnemers wel samen op dezelfde plaats aanwezig. Hoewel er interactieve whiteboard omgevingen bestaan waar gedistribueerde collaboratie mogelijk is, zijn de meeste systemen bedoeld voor *co-located* collaboratie: de deelnemers bevinden zich allemaal voor het whiteboard.

Indien we de indeling in synchrone en asynchrone collaboratie uit de vorige paragraaf (zie paragraaf 2.4) combineren met het onderscheid tussen gedistribueerde en *co-located* collaboratie, dan krijgen we het volgende schema, waarbij de typische toepassingen voor interactieve whiteboards zich in het donker gemarkeerde gebied bevinden. In het schema zijn als voorbeelden een aantal verschillende vormen van collaboratie opgenomen.

	Synchroon	Asynchroon
Co-located	• vergaderingen in meeting rooms • klassieke whiteboard toepassingen • toepassingen voor interactieve whiteboards	• groepskalenders
Gedistribueerd	• instant messaging • video- / audio-conference	• e-mail • newsgroups

Tabel 1 Soorten collaboratie

2.6 Oriëntatie

Het overgrote deel van de commercieel beschikbare interactieve whiteboards zijn bedoeld voor verticaal gebruik (dus om tegen een wand te hangen). Toch gebeurt collaboratie traditioneel aan een tafel waarrond de deelnemers met elkaar discussiëren en waarop ze eventueel objecten kunnen leggen die relevant zijn voor de taak [9].

Yvonne Rogers en Siân Lindley hebben in [9] een onderzoek gevoerd naar het effect van de oriëntatie (verticaal of horizontaal) van interactieve schermen en kwamen daarbij tot de misschien verrassende vaststelling dat bij horizontale opstellingen de proefpersonen meer ideeën produceerden en beter over elkaars activiteiten op de hoogte waren. Bovendien voelden de proefpersonen zich meer op hun gemak bij horizontale dan bij verticale opstellingen. Ook Eden et al stelden in [10] reeds dat personen aan een verticaal whiteboard zelden echt interactief werken en zich onbeholpen voelden.

Een voorbeeld van het gebruik van digitale horizontale tafels is de iRoom (zie paragraaf 3.6 en [5]), die naast verticale whiteboards ook gebruik maakt van een digitale tafel in zijn interactieve werkplek.

2.7 Classificatie van grote schermen o.b.v. toepassing

De prijzen van beeldschermen zijn de laatste jaren sterk gedaald. Ook grote beeldschermen (groter dan 1 meter diagonaal) volgen deze trend en zijn steeds sterker vertegenwoordigd in de maatschappij. Maar grote schermen dienen niet allemaal hetzelfde doel. In onderstaand schema is weergegeven voor welke toepassingen grote schermen doorgaans gebruikt worden al naargelang de doelgroep [13]. Merk echter op dat dit slechts een van de mogelijke manieren is om een classificatie op te stellen.

Privaat gebruik	Gedeeld gebruik	Publiek gebruik
Privacy-preserving public displays	Interactieve whiteboards	Reclamedisplays Public displays

Tabel 2 Classificatie grote schermen

Privaat gebruik
: Bij privaat gebruik kunnen we denken aan schermen die in publieke omgevingen (bijvoorbeeld de vertrekhal op de luchthaven) geplaatst zijn en waarvan men gebruik kan maken om persoonlijke gegevens op te halen zonder dat dit de eigen privacy in het gedrang brengt (*privacy-preserving public displays*). Een voorbeeld hiervan zou een digitale kiosk kunnen zijn die in de receptieruimte van een bedrijf is geplaatst waarmee het mogelijk is om je agenda te raadplegen en e-mails te versturen.

Publiek gebruik

Grote schermen voor publiek gebruik zijn alomtegenwoordig. Het gaat hier om schermen die zich in publieke omgevingen bevinden en die bepaalde informatie tonen. Dit soort van grote schermen wordt vaak aangewend voor publicitaire doeleinden (een reclameboodschap die getoond wordt), maar ook voor informatieve doeleinden. Een voorbeeld van dit laatste zou kunnen zijn een groot scherm dat in een toeristisch centrum is geplaatst en dat informatie weergeeft over toeristische activiteiten die op het programma staan.

Ook kunnen dit soort van schermen deel uitmaken van een groter systeem dat bestaat uit een netwerk van public displays, en waarbij het systeem op basis van locatie- en contextinformatie over de gebruiker aangepaste gegevens laat zien. Een voorbeeld hiervan is een persoon die op zoek is naar een bepaalde ruimte in een gebouw (waar in de gangen schermen zijn geplaatst), en die op elk scherm aangepaste instructies kan lezen over hoe hij op zijn bestemming geraakt. Aldus vervult het netwerk van public displays voor de gebruiker de rol van gids.

Gedeeld gebruik

Interactieve whiteboards vinden we terug in de groep van de grote schermen voor gedeeld gebruik. Met gedeeld gebruik wordt hier bedoeld dat meerdere gebruikers samen en tegelijk van het scherm gebruik maken. Meestal gaat het hier ook om een gemeenschappelijke opdracht, m.a.w. de gebruikers werken samen aan een bepaalde taak. Dit laatste hoeft echter niet altijd het geval te zijn, zoals bijvoorbeeld bij het naamloze prototype dat ontwikkeld is door Vogel (zie paragraaf 2.4 en [8]).

3 Bestaande digitale whiteboard omgevingen

In de literatuur zijn vele voorbeelden te vinden van onderzoek naar de interactie met interactieve whiteboards. Vaak, maar niet altijd resulteert dergelijk onderzoek in (een prototype van) een omgeving waarin bepaalde interactievraagstukken kunnen getoetst worden. In dit hoofdstuk worden enkele van dergelijke omgevingen besproken.

Het spreekt voor zich dat dit geen exhaustieve lijst is, maar een selectie van enkele omgevingen die in de literatuur vaak genoemd worden en/of die ik voor deze thesis interessant vond. In de volgende paragrafen komen volgende omgevingen aan bod: BlueBoard, Flatland, Tivoli, Dynamo, BrightBoard, iRoom, ContextWall en Kimura.

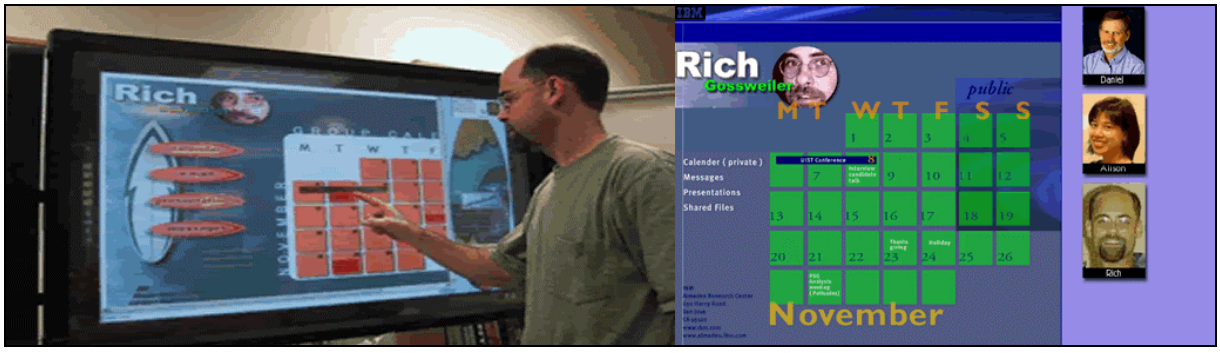
3.1 BlueBoard

Het BlueBoard-project werd ontwikkeld aan het IBM Almaden Research Center door Daniel M. Russel en Rich Gosswieler om de bruikbaarheid van computer interfaces in publieke ruimtes te onderzoeken. Het hoofddoel van het project was het ondersteunen van gemakkelijke, snelle toegang tot persoonlijke gegevens en van het delen van informatie door middel van *co-located* collaboratie [2].

3.1.1 Systeem

Het BlueBoard (zie Figuur 3.1) bestaat uit een groot plasmascherm met een diagonaal van 1,3 meter, een aanraakscherm en een kaartlezer (voor *badges*) en is bedoeld voor kort persoonlijk gebruik (bijvoorbeeld het snel nakijken van een agenda zoals rechts weergegeven in Figuur 3.1) en beperkt groepsgebruik (een klein aantal personen dat samen een ruwe schets van een bepaald idee uittekent). Interageren met het BlueBoard gebeurt niet met muis en toetsenbord zoals bij een traditionele desktopcomputer, maar via het aanraakscherm.

Hierdoor worden de mogelijkheden ietwat beperkt, maar om de doelstelling van snelle en gemakkelijke interactie niet in het gedrang te brengen, was het dan ook de bedoeling van de onderzoekers om BlueBoard eenvoudig te houden wat betreft de geboden functionaliteiten in tegenstelling tot de mogelijkheden die een computer met toetsenbord en muis kan bieden.



Figuur 3.1 BlueBoard [2]

Wanneer er geen gebruiker met het BlueBoard aan het werken is, toont het systeem herhaaldelijk (in een lus) een aantal pagina's met algemene informatie die relevant zijn voor de gemiddelde gebruiker, maar men kan ook persoonlijk gebruik maken van BlueBoard: door een magneetkaart door de kaartlezer te halen, meldt men zich aan bij het systeem en haalt het systeem de link naar de persoonlijke informatie van de gebruiker op uit de Badge Server Database. Met deze link haalt BlueBoard vervolgens de eigenlijke persoonlijke informatie op uit de Content Server Database. Na deze aanmeldingsprocedure verschijnt een icoontje van de gebruiker in kwestie in de rechterbovenhoek van het scherm, dit persoonlijk icoon wordt een *p-con* genoemd.

3.1.2 P-cons

P-con is een afkorting voor *personal icon*. Na het aanklikken van het *p-con* toont het systeem de startpagina van de gebruiker, zoals deze in de Content Server Database aanwezig is. Elke gebruiker moet zijn eigen startpagina zelf aanmaken en definiëren. BlueBoard zorgt dan voor het weergeven van de informatie.

Gebruikers kunnen hun *p-con* gebruiken om gegevens uit te wisselen: door het slepen van de informatie in kwestie (bijvoorbeeld een schets) naar het *p-con* van de persoon met wie men de informatie wil delen (deze persoon moet dan uiteraard wel aangemeld zijn). Wanneer een gebruiker zich vervolgens afmeldt door zijn magneetkaart opnieuw door de kaartlezer te halen, worden de gegevens via e-mail verstuurd naar de gebruiker. Op deze manier fungeert het *p-con* als een opslagbuffer om gegevens tijdelijk te bewaren tot de gebruiker zijn sessie aan het BlueBoard beëindigt.

3.2 Flatland

Flatland is een interactieve whiteboard interface met als doel het ondersteunen van informele activiteiten (zoals een takenlijst opstellen of het uittekenen van een routebeschrijving) in een kantooromgeving [6] [14].

3.2.1 Systeem

Flatland bestaat uit een SmartBoard dat gekoppeld is aan een videoprojector. Een SmartBoard is een whiteboard met aanraakscherm waar men met gewone whiteboardstiften op kan schrijven en met een stylus. Wanneer de gebruiker met zijn stylus op het bord tekent, wordt zijn tekening weergegeven op het SmartBoard.

3.2.2 Space management

Als een gebruiker zijn pen op het bord plaatst, maakt het systeem hiervoor automatisch een segment aan waarin de tekening terechtkomt. Een segment is een groeperingselement en wordt automatisch vergroot indien de gebruiker meerdere lijnen tekent, om alles wat bij elkaar hoort in het zelfde segment te doen vallen. Om te bepalen wat bij elkaar hoort, baseert Flatland zich onder andere op de tijd die verstreken is tussen het tekenen van de verschillende lijnen. Indien door dit automatisch groter worden van actieve segmenten (een actief segment is het

segment waar de gebruiker in aan het tekenen is), zullen de andere (passieve) segmenten automatisch verkleind worden om voldoende ruimte te maken.

3.2.3 Gedragingen

De gebruiker van Flatland kan verschillende soorten van gedrag toewijzen aan zijn tekening. Met deze gedragingen worden veel voorkomende taken aan het whiteboard ondersteund. Flatland ondersteunt volgende gedragingen: takenlijsten (*todo lists*), tweedimensionale tekeningen zoals bijvoorbeeld een schets van een woning (*2D drawing*), routebeschrijvingen (*maps*) en berekeningen (*calculator*). Door het toekennen van een bepaald gedrag aan een segment, gaat het systeem de lijnen die getekend worden, aanpassen op een manier die gekoppeld is aan dit bepaald gedrag. Een gebruiker die bijvoorbeeld een routebeschrijving aan het tekenen is, kan het *map*-gedrag toepassen op het segment waarin hij de wegbeschrijving tekent, waardoor het systeem een enkele lijn vervangt door een dubbele lijn om een weg weer te geven. In Figuur 3.2 zijn deze verschillende gedragingen weergegeven.

Figuur 3.2 Flatland [37]

3.3 Tivoli

Het Tivoli-project dateert reeds van 1991 en is een van de vroegste en bekendste ontwikkelingen op gebied van digitale whiteboards. Tivoli werd ontwikkeld aan het Xerox Palo Alto Research Center en werd gecommercialiseerd als kernapplicatie voor het digital whiteboard van Xerox, het Xerox *Liveboard*, een groot, stylus-gebaseerd, interactief videoscherm [12].

Met Tivoli wou men bij Xerox achterhalen in welke mate er voor deze -toen nog nieuwe- technologie andere technieken nodig waren voor wat betreft user interfaces, en onderzoeken welke functionaliteiten dergelijke interactieve systemen moesten bieden om eenvoudig bruikbaar te kunnen zijn voor vergaderingen van kleine teams (*informal workgroup meetings*).

3.3.1 Functies van Liveboard

Het onderzoeksteam van Tivoli stelt dat de meest primaire functie van een whiteboard het ondersteunen is van de interactie tussen mensen. Daarom zijn volgens hen de belangrijkste criteria voor een whiteboard dat de interactie onbewust en vloeiend moet gebeuren. Het criterium van onbewuste interactie moet voor de gebruikers waarborgen dat ze enkel op elkaar hun aandacht moeten vestigen in plaats van op het whiteboard, en het vloeiende karakter moet er voor zorgen dat het creëren en bespreken van ideeën zonder vervelende onderbrekingen en storingen gebeurt.

3.3.2 Vergaderingen van kleine teams

De doelgroep van Tivoli strekt zich niet uit over alle mogelijke vormen van samenwerking tussen personen, maar beperkt zich tot dat soort van vergaderingen waarbij enkele deelnemers nauw samenwerken rond een bepaald idee of probleem (*small workgroup meetings*). Dergelijke vergaderingen bestaan meestal uit ongeveer acht personen (voor een groter aantal zou het *Liveboard* met zijn afmetingen van ongeveer 120 cm bij 80 cm te klein zijn) en er is doorgaans geen notulist noch iemand die formeel de vergadering leidt.

De onderzoekers van Tivoli ontwikkelden vooraf enkele concrete scenario's voor het gebruik van Tivoli die van elkaar verschilden in inhoud (bijvoorbeeld het maken van een ontwerp of een administratieve taak) en stijl (bijvoorbeeld informeel of gestructureerd). Uit al deze scenario's werd vervolgens een selectie gemaakt. Tot de kernfunctionaliteit van Tivoli moest zeker het volgende behoren: eenvoudige peninvoer, het ondersteunen van meerdere pagina's, laden en bewaren van pagina’s, afdrukken en het importeren van afbeeldingen. Samenwerking van op afstand (met deelnemers die zich op een andere locatie bevinden), integratie met andere apparaten, en hulpmiddelen voor het beheren van vergaderingen waren functionaliteiten die niet in eerste instantie zouden worden onderzocht.

3.3.3 Kenmerken

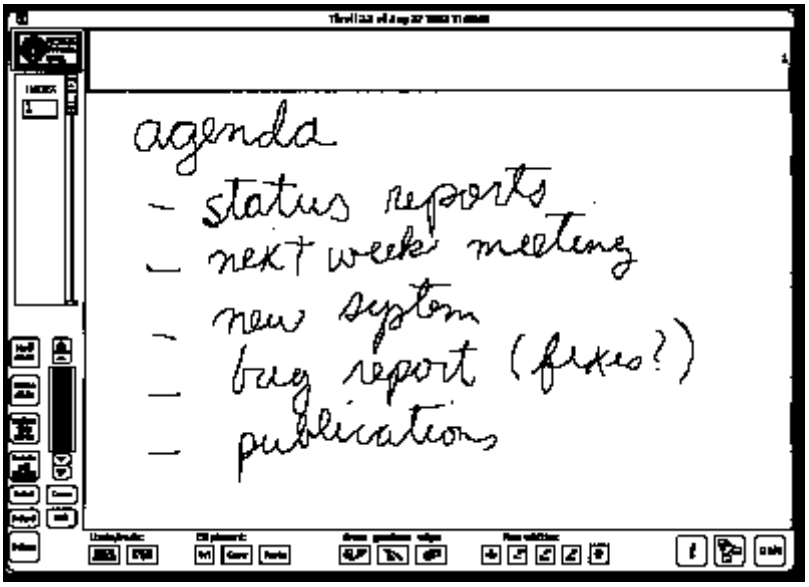
De onderzoekers van Tivoli wezen als een van de eersten op het feit dat klassieke grafische user interfaces voor de gewone desktopcomputer uit die tijd gebonden waren aan een aantal veronderstellingen over de gebruikte technologie en de gebruikssituatie, en bijgevolg niet zonder meer overgedragen kunnen worden naar interactieve whiteboards. De muis van een klassieke desktop computer is in hoofdzaak bedoeld als aanwijzer, wat niet helemaal hetzelfde is als wat we gewend zijn te doen met een pen: met een pen kunnen we weliswaar ook aanwijzen, maar kunnen we ook gemakkelijk tekeningen maken en woorden schrijven. Bovendien is de relatie tussen de beweging van de muis en de beweging van de cursor op het scherm indirect, terwijl dit voor een pen op een whiteboard een duidelijk directe relatie is, en er eigenlijk zelfs geen cursor nodig is (zie paragraaf 4.1.1).

Niet alleen wat de invoer betreft, zijn er verschillen in gebruikte technologie. Ook het scherm vertoont een duidelijk verschil: de grootte. Een normaal scherm van een desktop computer is niet te groot om als gebruiker het overzicht te verliezen, wat echter bij een *Liveboard* wel kan gebeuren: wanneer een gebruiker een *pop-up* te zien krijgt die aan de andere kant van het whiteboard verschijnt, bestaat de kans dat

dit niet of minder makkelijk opgemerkt zal worden. De klassieke opstelling van een menubalk aan de rand van het scherm (doorgaans bovenaan) is eveneens niet langer adequaat: de gebruiker moet applicatiecontrole kunnen uitoefenen op de positie waar hij zich bevindt, namelijk bij de pen, bijvoorbeeld door het gebruik van *gestures* (zie paragraaf 4.2.2).

Tot slot is er nog de gebruikssituatie die kan verschillen: een *Liveboard* ondersteunt drie pennen, terwijl voor een normale desktopcomputer slechts één muis wordt ondersteund. Ook hiervoor zijn aanpassingen nodig: bepaalde tools op het scherm kunnen door meerdere gebruikers gedeeld worden (bijvoorbeeld tools om objecten te tekenen).

Zoals op Figuur 3.3 te zien is, heeft Tivoli een tamelijk informeel uiterlijk. Het lijkt een beetje op een zeer eenvoudig tekenprogramma, maar onderscheidt zich van deze laatste soort programma's doordat tekeningen als objecten (bestaande uit lijnstukken) worden opgeslagen en niet als pixel maps. Dit heeft het voordeel dat getekende objecten gemakkelijk te selecteren en verplaatsen zijn zonder delen van andere objecten mee te moeten selecteren en dat eigenschappen zoals bijvoorbeeld de kleur van een object eenvoudig te wijzigen zijn.



Figuur 3.3 Tivoli [38]

Tivoli maakt gebruik van zogenaamde *gestures*. Het begrip *gestures* kent in de literatuur meerdere invullingen, maar voor Tivoli zijn dit kleine *meta-tekeningen* die niet geïnterpreteerd worden als af te beelden grafische elementen, maar als een opdracht (commando). Het probleem met gestures is dat het systeem duidelijk gemaakt moet worden welke tekeningen *gestures* zijn en welke gewoon grafische elementen. De onderzoekers van Tivoli hebben gekozen om dit op te lossen met een *post-fix* operator: voor het maken van een *gesture* volstaat het om eerst de *gesture* te tekenen (op dat punt is het gewoon een grafisch element en wordt deze ook als dusdanig weergegeven) en om dan vervolgens met een druk op de knop van de pen

aan het systeem duidelijk te maken dat het getekende object geïnterpreteerd moet worden als een *gesture*.

3.4 Dynamo

Dynamo is een interactief systeem met grote schermen dat bedoeld is voor een grote groep gebruikers. Het gaat hier niet alleen om groepen van een tiental personen, maar ook om grotere groepen, zoals bijvoorbeeld de schoolgemeenschap in Engeland waar Dynamo reeds op de proef is gesteld. Dynamo is een platform waarmee de leden van een gemeenschap allerlei soorten digitale media zoals video (leuke filmpjes), audio (liedjes), afbeeldingen en dergelijke kunnen delen, uitwisselen en aan elkaar tonen [15].

3.4.1 Systeem

De ondersteuning voor meerdere gebruikers gebeurt bij Dynamo door de zogenaamde interactiepunten (*interaction points*). Dit zijn locaties waar invoerapparaten (normaal een draadloze muis en toetsenbord of een draagbare computer) geplaatst zijn, waarmee men kan interageren met het systeem. Het delen en uitwisselen van informatie gebeurt door middel van USB-hubs die verbonden zijn met Dynamo. Op deze manier kunnen gebruikers hun digitale media transporteren van hun persoonlijke apparaten (bijvoorbeeld een USB-stick of digitale camera) naar Dynamo.

Elk interactiepunt in Dynamo wordt op het scherm weergegeven door een *telepointer*, wat niet meer is dan een muiscursor in een bepaalde kleur, waardoor het mogelijk is om de muiscursor van elke gebruiker van een interactiepunt in een andere kleur weer te geven. Dit is te zien in Figuur 3.4: er zijn drie gebruikers (zie de icoontjes onderaan) die zich bevinden aan verschillende interactiepunten en die elk hun eigen muiscursor hebben: een blauwe, een rode en een groene.

Figuur 3.4 Dynamo [39]

Het toevoegen aan en ophalen van informatie aan Dynamo gebeurt via *palettes*. Een *palette* is een groep iconen die informatiebronnen en informatiebestemmingen voorstellen (in Figuur 3.4 bovenaan het scherm). Een informatiebestemming kan bijvoorbeeld een printer of een USB-stick zijn, een informatiebron een filmpje of een Powerpoint-presentatie. Dynamo bestaat uit twee soorten palettes: publieke palettes en individuele palettes.

Een publiek palet kan door iedereen gebruikt worden: op deze manier heeft iedereen toegang tot apparaten die als publiek gedefinieerd zijn (zoals bijvoorbeeld een printer). Door het zich toe-eigenen van een gedeelte van het schermoppervlak (dit wordt in Dynamo *carving* genoemd) kan een gebruiker een deel van het scherm voor zich opeisen, maar hiervoor moeten gebruikers zich eerst registreren. Dit deel van het scherm is dan privaat en niet toegankelijk voor anderen, maar de eigenaar van een privaat gedeelte kan de toegang uitbreiden door anderen uit te nodigen.

Tot slot zijn er nog twee functies die het delen van informatie ondersteunen: pakketjes (*parcels*) en notities (*notes*). Met pakketjes ondersteunt Dynamo het asynchrone delen van informatie: media kunnen geplaatst worden op het scherm zodat anderen ze op een later tijdstip kunnen ophalen. Gebruikers kunnen notities aan items op het scherm koppelen en op deze manier een asynchrone discussie voeren over deze items.

3.4.2 Implicaties voor het ontwerp van interactieve systemen

Op basis van een onderzoek dat men voerde bij een schoolgemeenschap, kwamen een aantal vaststellingen naar voor met betrekking tot het ontwerp van een interactief systeem dat gebruikt wordt door grote groepen.

Een eerste vaststelling was dat een interactief systeem zoals Dynamo moet passen in de omgeving en zich moet kunnen integreren met de andere apparaten die reeds in de ruimte aanwezig zijn. Ook bleek uit het onderzoek dat een gemeenschappelijk gebruik van een interactief systeem in sterke mate configureerbaar moet zijn en zo veel mogelijk soorten van digitale media moet ondersteunen.

Andere vaststellingen met betrekking tot het ontwerp waren dat de applicaties die draaien op een dergelijk interactief systeem niet te sterk gestructureerd mochten zijn en dat de interactie met het systeem vlot en op een aangename manier te leren moet zijn, zonder dat gebruikers de hulp van anderen moeten inroepen.

3.5 BrightBoard

BrightBoard is op het eerste gezicht precies hetzelfde als een klassiek (niet digitaal) whiteboard. Maar BrightBoard is toch meer dan dat: het biedt functionaliteiten die niet beschikbaar zijn bij een klassiek whiteboard zoals het bewaren, printen, faxen en verzenden van documenten door gebruik te maken van bepaalde markeringen die door de gebruiker op het whiteboard aangebracht worden [7].

3.5.1 Computer Augmented Environment

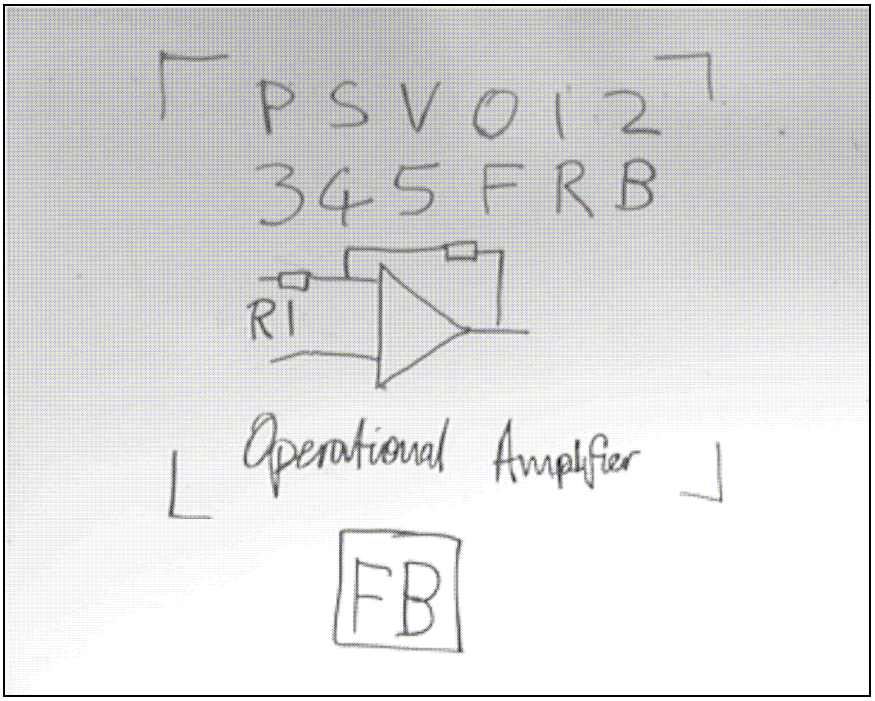
BrightBoard noemt zichzelf een voorbeeld van een *Computer Augmented Environment* (CAE). De filosofie achter het creëren van een CAE is dat computers meer de rol van assistent moeten spelen in plaats van de rol van blinde slaaf: in plaats van blindelings opdrachten uit te voeren (bijvoorbeeld het afdrukken van een budgetrapport), zou de afdruk in een CAE klaar liggen wanneer het nodig is zonder dat dit commando gegeven is: het systeem, de CAE, wéét dat de afdruk nodig is (bijvoorbeeld omdat het in de agenda ziet dat de gebruiker vandaag een budget-review vergadering heeft gepland).

Wanneer computers de wereld van de gebruikers op deze manier moeten helpen verbeteren, hebben ze uiteraard veel informatie nodig. Dit kan dan zoals in het voorbeeld gaan over gegevens uit iemands digitale agenda, maar ook andere soorten informatie zoals bijvoorbeeld de lichaamshouding van de gebruiker. Hoe meer informatie het systeem heeft, hoe beter het systeem de rol van assistent kan vervullen maar ook hoe meer informatie het systeem tot zich moet nemen, en hoe meer sensoren hiervoor nodig zijn. Bij BrightBoard heeft men zich beperkt tot één sensor voor visuele informatie, vandaar dat BrightBoard ook wel een *Video Augmented Environment* (VAE) genoemd wordt.

3.5.2 Systeem

BrightBoard bestaat uit een klassiek whiteboard en een videocamera. Het systeem geeft als dit nodig is feedback aan de gebruiker via auditieve signalen. Er zijn geen styli nodig om input te geven aan het systeem. Zoals op Figuur 3.5 te zien is, nemen markeringen een centrale plaats in. Er zijn twee soorten markeringen: commando- en selectiemarkeringen. Commandomarkeringen zijn vierhoekjes met daarin een of meerdere letters die een commando voorstellen. Selectiemarkeringen zijn twee korte lijnen die aan de uiteinden haaks op elkaar zijn geplaatst (winkelhaakjes).

In de figuur is bijvoorbeeld te zien hoe door de gebruiker het ‘FB’ commando gegeven wordt (dit zou bijvoorbeeld kunnen staan voor ‘Fax to Bart’. Wat er precies gefaxt wordt, de selectie, is aangegeven met de selectiemarkeringen in de vier hoeken van de afbeelding.



Figuur 3.5 BrightBoard [7]

Het systeem is sterk gebaseerd op het herkennen van de markeringen. Wanneer op het whiteboard getekend wordt, gaat het systeem het beeld van de videocamera analyseren en gaat het op zoek naar markeringen die het kent (commando- en selectiemarkeringen). Wanneer het dergelijke markering gevonden heeft, voert het systeem het bijhorende commando uit op de aangegeven selectie.

3.6 iRoom

Een team van onderzoekers aan de Stanford University onderzocht met het *interactive workspaces* project de interactie tussen mens en computer bij grote beeldschermen [5]. Het prototype van deze interactive workspace noemden ze iRoom en de softwareinfrastructuur die ze voor deze omgeving ontwikkelden, doopten ze iROS, wat een afkorting is voor *interactive Room Operating System*.

3.6.1 Systeem

De iRoom bestaat uit drie aanraakgevoelige whiteboards en een *interactive mural*, wat letterlijk interactieve muurschildering betekent. De *interactive mural* is een zeer groot beeldscherm met een diameter van meer dan 270 cm en een resolutie van 9 megapixels. Verder bestaat iRoom ook nog uit een digitale tafel van ongeveer 90 bij 120 centimeter (zie Figuur 3.6). De iRoom is tot slot ook uitgerust met camera's, een draadloos netwerk, microfoons en draadloze schakelaars.

De iRoom is ontworpen om drie typische kenmerken van activiteiten te ondersteunen die kenmerkend zijn voor een interactieve werkplek. Ten eerste is dit het uitwisselen van gegevens tussen de verschillende gegevensdragers. Een tweede kenmerk is dat het controleren van de verschillende apparaten op een flexibele manier mogelijk moet zijn: elke gebruiker moet in principe vanop elke plaats elk apparaat of applicatie

kunnen controleren. Tot slot moest iRoom in staat zijn de activiteiten van verschillende applicaties te coördineren, zo zou bijvoorbeeld de financiële impact van een wijziging in het ontwerp van een CAD-tekening ook meteen zichtbaar moeten zijn in een spreadsheetapplicatie die op een ander scherm het financiële plaatje toont.

Figuur 3.6 iRoom [5]

3.6.2 Interactie

Voor de *interactive mural* bleken de gangbare invoermethodes zoals stylus of aanraking niet toereikend omwille van de grootte van het scherm. Daarom kozen de onderzoekers voor ultrasone en lasergebaseerde invoer (zie paragraaf 2.2). Voor de drie andere interactieve whiteboards die niet zo groot zijn, konden de gangbare methoden wel volstaan, namelijk stylus en aanraking met de vingers.

Een belangrijke doelstelling van de iRoom was om de interactie met het systeem zeer vlot te laten verlopen opdat het de aandacht van de gebruikers niet zou afleiden van hun interpersoonlijke interactie. Hiertoe werden een drietal technieken gebruikt: *FlowMenu*, *Zoomscape* en *typed drag-and-drop*.

FlowMenu (zie paragraaf 4.2.1.3) is een contextmenu zoals Windowsgebruikers dit te zien krijgen wanneer ze op de rechter muisknop drukken: het menu verschijnt pas na het klikken (in plaats van altijd zichtbaar te zijn, bijvoorbeeld bovenaan het scherm in een balk) en verschijnt precies op de plaats waar de muiscursor zich bevond bij het klikken. Verder is het *FlowMenu* radiaal in plaats van lineair (zoals bijvoorbeeld bij Windows XP) waardoor elke menuoptie even ver van de muiscursor verwijderd is en verschillende menuacties in één beweging kunnen gebeuren, waardoor de gebruikers bepaalde menusequenties kunnen aanleren wat minder mentale aandacht vraagt van de gebruiker.

Bij *ZoomScape* worden de objecten op het scherm vergroot of verkleind afhankelijk van de positie die ze op het scherm innemen. Het scherm is daarbij verdeeld in verschillende gebieden: in de gebieden die het verst van het centrum liggen, worden objecten kleiner weergegeven omdat ze niet in het middelpunt (letterlijk en figuurlijk) van de belangstelling staan, en dus gerust wat kleiner geschaald kunnen worden om meer ruimte vrij te maken voor het centrale gebied op het scherm.

Typed drag-and-drop ten slotte is een techniek om handgeschreven tekst te herkennen en te interpreteren. In tegenstelling tot sommige andere systemen wordt hierbij de tekst zoals die met de hand op het whiteboard is geschreven niet vervangen door getypte symbolen, maar geannoteerd met de geïnterpreteerde tekst. De geschreven tekst blijft dus staan.

3.6.3 Applicatiecontrole

Bij interactieve whiteboards stelt zich het probleem van applicatiecontrole (zie paragraaf 4.2). En bij omgevingen zoals de iRoom die bestaan uit meerdere interactieve schermen, stelt dit probleem zich nog scherper: gebruikers moeten kunnen interageren met meerdere schermen en willen meerdere schermen tegelijk gebruiken terwijl ze natuurlijk niet voor elk scherm tegelijk kunnen staan.

Om dit probleem het hoofd te bieden, zonder een natuurlijk en vlot verloop van een activiteit abrupt te verstoren, ontwikkelden de onderzoekers van de iRoom een mechanisme dat ze *PointRight* noemden. Met *PointRight* kan de muiscursor (of een andere pointer zoals de pen van een PDA) van eender welk apparaat in een soort van super- muiscursor (*superpointer*) veranderen op vraag van de gebruiker.

Deze superpointer kan dan gebruikt worden om alle interactieve whiteboards te controleren: wanneer een superpointer de rand van het scherm van een interactief whiteboard overschrijdt, gaat deze verder op het volgend interactief whiteboard alsof alle beeldschermen in de iRoom één grote virtuele desktop vormen.

3.7 ContextWall

ContextWall is een ontwikkeling van Patrick Baudisch die gebaseerd is op het principe van *focus-plus-context screens*. Hierbij is het beeldscherm opgebouwd uit twee gebieden: een context- en een focusgebied. Voor een systeem voor twee gebruikers wil dit zeggen dat elke gebruiker beschikt over zijn eigen focusscherm, en dat het contextscherm gedeeld wordt door de beide gebruikers. De filosofie achter deze twee verschillende soorten schermen is dat de gebruikers het gemeenschappelijke gebied kunnen gebruiken voor de activiteiten die ze gezamenlijk doen, en het eigen focusscherm voor hun private activiteiten [16].

3.7.1 Systeem

ContextWall noemt zich een *Single Display Groupware (SDG) system* en biedt aan zijn gebruikers dus zowel een persoonlijk als een gedeeld werkvlak aan. Zoals op Figuur 3.7 te zien is, bestaat het grote scherm uit twee focusschermen met hoge resolutie die ingebed zitten in een contextscherm met lagere resolutie. Elke gebruiker beschikt over een toetsenbord en muis en kan zich hiermee vrij bewegen over zijn

eigen focusscherm, het ganse contextscherm én het focusscherm van de andere gebruiker.

Figuur 3.7 ContextWall [16]

Het contextscherm wordt op een wand geprojecteerd, terwijl elk focusscherm aangestuurd wordt door een aparte computer, die verbonden is met een Linux-server waarop een VNC-server draait.

3.7.2 Collaboratiestijlen

Tijdens een typisch gebruikscenario (bijvoorbeeld wanneer twee gebruikers samen een website ontwerpen) hanteren de gebruikers onbewust verschillende collaboratie-stijlen tijdens het ontwerpproces.

Bij de *individuele* stijl werken beide gebruikers onafhankelijk van elkaar. Ze gebruiken vooral de focusschermen, terwijl de contextschermen mogelijkheden bieden om met grote afbeeldingen te werken, of een e-mailprogramma in de gaten te houden. Bij de *peer-to-peer* stijl wordt er onderling informatie uitgewisseld: de ene gebruiker sleept bijvoorbeeld een document naar het focusscherm van de andere gebruiker omdat die daarom gevraagd heeft. Bij de *groepsstijl* ten slotte zijn er meer dan twee gebruikers betrokken, en zitten de gebruikers verder van het scherm vandaan om het zicht niet te blokkeren.

3.8 Kimura

Het doel van de ontwikkelaars bij Kimura is het verbeteren van traditionele kantoor-omgevingen, en wel die kantooromgevingen waarin computers een belangrijke rol spelen. Om dit voor elkaar te krijgen, tracht Kimura gebruik te maken van de context van een activiteit. Deze context wordt aan de gebruiker getoond op een apart scherm en moet hem in staat stellen gemakkelijker en efficiënter met meerdere taken bezig te zijn [3] [4].

3.8.1 Systeem

Kimura bestaat uit interactieve whiteboards en een gewone desktopcomputer. Vanuit het standpunt van de hardware, voegt Kimura dus interactieve whiteboards toe aan de werkplek van de gebruiker. Kimura verdeelt de werkplek van een gebruiker in

twee delen: een *focal display* (een desktop beeldscherm, waarop de gebruiker zijn activiteiten uitvoert zoals in een traditionele werkplek) en *peripheral displays* (die extra contextinformatie weergeven en die geprojecteerd worden op de wand) zoals te zien is op Figuur 3.8.

Figuur 3.8 Kimura [4]

3.8.2 Physical context en virtual context

Voor Kimura is een werkactiviteit (zoals bijvoorbeeld het beheren van een project) voor te stellen als een cluster van documenten en signalen. Signalen modelleren interacties met andere personen en objecten die voor de taak relevant zijn. Het Kimura systeem noemt dergelijke cluster een *working context*. Een *working context* kan bestaan uit documenten als tekstbestanden en webpagina's, maar ook uit aanwijzingen over activiteiten zoals het afdrukken van documenten of e-mails waarop nog geen reactie is gekomen. Kimura onderscheidt een fysische en virtuele context (*physical context* en *virtual context*).

De *virtual context* wordt opgebouwd uit allerlei bronnen van informatie. Kimura legt zo veel mogelijk gebruikersactiviteit vast door gebruik te maken van *hooks* via de Win32 API. Op deze manier houdt Windows Kimura op de hoogte van elke relevante actie van de gebruiker zoals het openen van een venster, een toets indrukken op het toetsenbord en een muisknop indrukken. Kimura volgt ook het e-mailverkeer van de gebruiker op en onthoudt met welke randapparatuur de gebruiker bezig is. Daarnaast neemt Kimura regelmatig een screenshot van de desktop. Al deze informatie wordt geïntegreerd in zogenaamde *montages* die aldus visuele representaties vormen van een gebruikersactiviteit.

Naast informatie over de virtuele context, houdt Kimura ook gegevens bij over de fysieke context van een activiteit (*physical context*). De gegevens die hiervoor van belang zijn, zijn onder andere de beschikbaarheid en fysieke locatie van collega's. Om deze gegevens te verzamelen beschikt Kimura over sensoren die de exacte locatie van personen detecteert en over informatie over hun activiteiten (door het in de gaten houden van hun desktop via onder andere *keyloggers*).

4 Interactie

In dit hoofdstuk komen enkele onderwerpen aan bod die extra aandacht verdienen in het kader van de interactie met interactieve whiteboards. In een eerste paragraaf zal dieper ingegaan worden op de verschillen tussen interactieve whiteboards en de conventionele desktopcomputer, en gewezen worden op de impact van deze verschillen op het ontwerp van user interfaces voor interactieve whiteboards.

Verder zullen enkele specifieke vormen van interactie behandeld worden die gebruikt kunnen worden voor interactieve whiteboards. Ook het probleem van *floor control* komt aan bod: whiteboards worden immers door meerdere gebruikers tegelijk gebruikt en de meeste interactieve whiteboards kunnen echter maar van één gebruiker tegelijk invoer verwerken. Tot slot worden in een laatste paragraaf nog enkele typische taken weergegeven die op een interactief whiteboard uitgevoerd kunnen worden.

4.1 Verschillen met klassieke desktop

4.1.1 Invoer

4.1.1.1 Mogelijkheden voor invoer

Alvorens in te gaan op de verschillen tussen interactieve whiteboards en de klassieke desktopcomputers (waarbij de invoer gebeurt door middel van muis en toetsenbord), is het nuttig om even een blik te werpen op de verschillende mogelijkheden waarop de invoer van de gebruiker kan doorgegeven worden aan een interactief systeem.

In Tabel 3 is een mogelijke indeling van die verschillende mogelijkheden weergegeven. Daarin is enerzijds het onderscheid gemaakt op basis van de relatie tussen de beweging van het invoerapparaat en de beweging van de cursor op het scherm, en anderzijds op basis van de relatie tussen de fysieke locatie van het invoerapparaat en het scherm.

		relatie m.b.t. beweging	
		relatief	absoluut
relatie m.b.t. fysieke locatie	direct	vb. HybridPointing	stylus op whiteboard
	indirect	muis	stylus op tablet

Tabel 3 Invoermogelijkheden

Bij directe invoer gebeurt de manipulatie van object op dezelfde positie als waar de pen of vinger zich bevindt. De gebruiker kan als het ware zelf de objecten rechtstreeks op het scherm aanraken. Bij indirecte invoer is dit niet het geval, zo is er bij het werken met een muis een ruimtelijke scheiding (afstand) tussen de muis en

het scherm waarop de manipuleerbare objecten worden weergeven, van het aanraken van het scherm met het invoerapparaat is hier geen sprake [17].

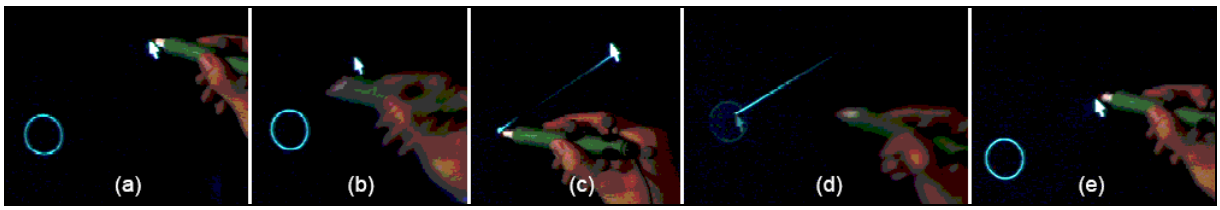
Bij een absoluut invoermechanisme is er een één-op-één relatie tussen de posities van het invoerapparaat en de cursor: wanneer het invoerapparaat zich bevindt op bepaalde coördinaten, dan zal ook de cursor zich op die coördinaten bevinden. Bij relatieve invoer is deze relatie tussen de posities niet absoluut, maar worden de bewegingen van de cursor vermenigvuldigd en samengeteld met variabelen.

Absolute, directe invoer (zoals gebruikelijk is voor digitale whiteboards) is de meest natuurlijke vorm, maar er kleven toch ook enkele nadelen aan vast. Vooreerst is de locatie waar de manipulatie gebeurt (de *locus of attention*) ook meteen de minst zichtbare plaats omdat handen, vingers en styli het zicht kunnen belemmeren. Bovendien wordt deze locatie -althans bij *front-projection* (zie paragraaf 2.2.2.1)- geplaagd door schaduwen van de genoemde lichaamsdelen en stylus. Daarnaast is er het probleem van bereik: op grote schermen moet een gebruiker vaak ver reiken, of zich zelfs een paar stappen verplaatsen, om bepaalde schermelementen te manipuleren [18].

Bij relatieve directe invoer stellen de laatstgenoemde problemen zich niet, al voelt het niet zo natuurlijk aan als absolute invoer. Door het invoeren van een *offset*, bevindt de *locus of attention* zich niet langer op de plaats van de invoer maar op een bepaalde afstand van de invoerlocatie vandaan (onder, boven, links, rechts). Het probleem van het bereik kan opgelost worden door kleine bewegingen van het invoerapparaat te vertalen naar grote bewegingen van de cursor. Omgekeerd kan het trouwens ook: een grote beweging van de pen leidt tot een kleine beweging van de cursor, wat belangrijk kan zijn voor taken waar een grote nauwkeurigheid voor vereist is.

Een laatste niet te onderschatten voordeel van relatieve, directe invoer is dat voor applicaties waarbij meerdere gebruikers tegelijk actief zijn (en voor interactieve whiteboards is dit al snel het geval), het in principe mogelijk is om op dezelfde schermlocatie te werken, en toch fysiek elkaar niet voor de voeten te moeten lopen.

In dit verband is de techniek van *HyperPointing* het vermelden waard [17]. Bij HyperPointing is het mogelijk te alterneren tussen relatieve en absolute invoer met een direct invoerapparaat (zie Figuur 4.1).



Figuur 4.1 HyperPointing [17]

Toelichting bij Figuur 4.1: In (a) is het systeem in de absolute invoermodus: de cursor en de pen bevinden zich op dezelfde plaats. Door een speciaal schermelement aan te raken (b), schakelt het systeem over naar de relatieve invoermodus: er is nu

steeds een vaste afstand (weergeven door een lijnstuk) tussen pen en cursor (c). Door de pen op te heffen van het scherm (d), keert het systeem terug naar de absolute invoermodus (e).

4.1.1.2 Verschillen met desktopcomputers

Bij een klassiek desktopsysteem gebeurt de invoer door de gebruiker met de muis. Bij interactieve whiteboards is dit niet het geval: als invoerapparaat gebruikt men een stylus (pen), of kan men zelfs gewoon de vingers van de hand gebruiken (afhankelijk van de gebruikte technologie, zie paragraaf 2.2.1).

Een klassieke muis noemt men een *pointing device*, vandaar dat men ook wel eens spreekt van een muisaanwijzer, terwijl een stylus naast een aanwijzerfunctie ook op een natuurlijke wijze gebruikt kan worden om te schrijven en tekenen. Functies zoals het schrijven van tekst en tekenen zijn veel vanzelfsprekender uit te voeren met een stylus dan met een muis ([12]). Een muis is -zoals uiteengezet in paragraaf 4.1.1.1- een indirect invoerapparaat in die zin dat door middel van extra berekeningen de beweging van het invoerapparaat vertaald wordt in bewegingen van de cursor op het scherm.

Bij een stylus daarentegen wijst men precies naar het scherm en hoeft er eigenlijk zelfs geen cursor te bestaan [19]: deze cursor zou immers toch maar in de weg staan omdat de positie van die cursor dezelfde is als de positie van de pen (hierbij ga ik gemakkelijkheidshalve even voorbij aan eventuele fouten in de calibratie van het invoerapparaat). Het feit dat we te doen hebben met een direct invoerapparaat maakt een directe, natuurlijke interactie mogelijk met objecten op het whiteboard.

4.1.2 Schermgrootte

Zelfs bij een desktopcomputer met een groot scherm (tegenwoordig is een beeldschermdiagonaal van 19 of 21 duim niet uitzonderlijk meer) is de gebruiker nog redelijk goed in staat om een algemeen overzicht van de desktop te krijgen vanop een normale kijkafstand. Bij grotere schermen zoals interactieve whiteboards is dit echter niet zo vanzelfsprekend meer: de gebruiker staat vaak te dicht bij het scherm om een goed overzicht te hebben van alles wat er op het whiteboard staat.

4.1.3 Ontwerpimplicaties en GUI beschouwingen

Door het verschil in grootte zijn er een aantal gevolgen voor het ontwerp van gebruiksinterfaces waar men rekening mee moet houden. Zo heeft men bij de ontwikkeling van Tivoli [12] terdege rekening gehouden met de grootte en de plaatsing van *pop-up* berichten opdat de gebruikers ze altijd zouden opmerken. Maar niet alleen *pop-up* berichten vragen aandacht, ook de plaatsing en grootte van menu's en commandoknoppen moet aangepast worden vanwege het probleem van het manipulatiebereik van de gebruiker, waarover ik in paragraaf 5.2.2 meer zal vertellen.

In het algemeen is het raadzaam om bij het ontwerp van grafische user interfaces (GUI) de controle die de gebruiker uitoefent zo veel mogelijk te lokaliseren bij de stylus, en zeker voor activiteiten die de gebruiker vaak uitvoert en voor activiteiten die deel uitmaken van het schrijven op een interactief whiteboard. Dit soort

activiteiten kan dus bijvoorbeeld ondersteund worden door middel van *gestures* (zie paragraaf 4.2.2), zoals dit gedaan is bij Tivoli [12].

4.2 Applicatiecontrole

4.2.1 Menu's

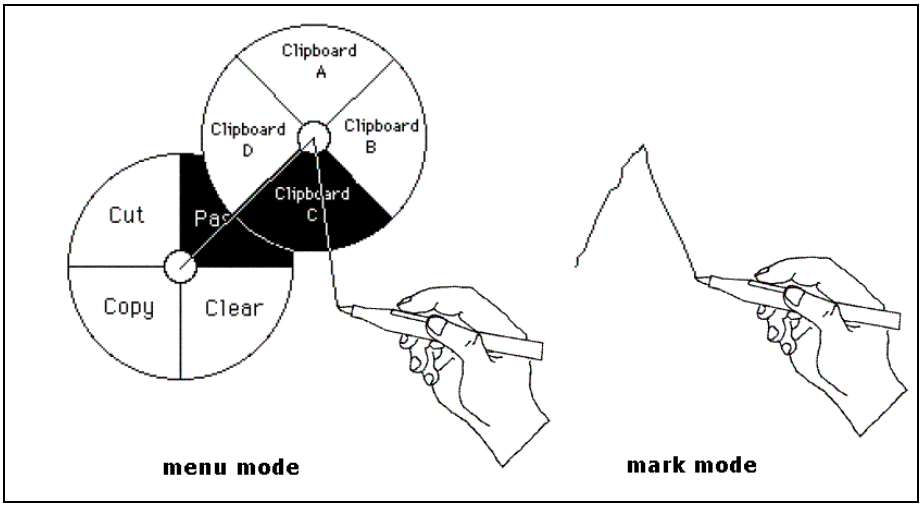
4.2.1.1 Marking menu

In "*The design and evaluation of marking menus"* beschrijft G.P. Kurtenbach een *marking menu* als een interactietechniek die de gebruiker in staat stelt om een selectie te maken uit een menu van verschillende items. Deze selectie kan op twee verschillende manieren gebeuren (zie Figuur 4.2):

menu mode
: In deze modus gebeurt de selectie door een keuze te maken uit een menu dat getoond wordt door het systeem. Deze modus wordt geactiveerd doordat de gebruiker de stylus tegen het scherm drukt en zo houdt gedurende ongeveer één derde van een seconde. Deze actie noemt Kurtenbach de *press-and-wait*. Na de *press-and-wait* toont het systeem een radiaal menu op de locatie waar de stylus het scherm raakt.

  Bij een radiaal menu zijn de menu-items niet onder elkaar weergegeven (zoals bij de traditionele menu's van de meest courante applicaties het geval is) maar in een cirkel rond de cursor (merk op dat deze cursor bij interactieve whiteboards niet altijd zichtbaar is). Wanneer het radiaal menu getoond wordt, kan de gebruiker zijn stylus bewegen naar de menu-optie van zijn keuze. De gekozen menu-optie wordt vervolgens visueel aangepast weergegeven (bijvoorbeeld door de kleur van het item te veranderen) en de gebruiker kan tot slot zijn keuze bevestigen door de stylus weg van het scherm te bewegen (zie de linkerkant van Figuur 4.2).

mark mode
: De selectie gebeurt in deze modus door het maken van een *mark* (een of meer lijnstukken). Deze modus wordt ook geactiveerd doordat de gebruiker de stylus tegen het scherm duwt maar nu -in tegenstelling tot de menu mode- beweegt de gebruiker vervolgens onmiddellijk (dus zonder de *press-and-wait)* de stylus in de richting van de gewenste menu-optie. In plaats van het tonen van het menu, wordt enkel het 'inktspoor' getoond dat de gebruiker heeft getekend (zie rechterkant van Figuur 4.2). De keuze van de gebruiker wordt bevestigd doordat hij de stylus weg van het scherm beweegt.



Figuur 4.2 Marking menu [20]

Het belangrijkste kenmerk van een *marking menu* als interactietechniek is dat de beweging (de *mark*) die met de stylus uitgevoerd moet worden om een optie te kiezen in *mark mode*, precies dezelfde is als de beweging die de gebruiker met de stylus moet maken om dezelfde menu-optie te kiezen in *menu mode*.

Een *marking* menu kan bestaan uit meerdere niveaus (hiërarchisch). Als een menu-item uit een submenu bestaat, en de gebruiker beweegt de stylus naar dat menu-item dan verschijnt in *menu mode* na een korte wachttijd (met de stylus tegen het scherm gedrukt) het submenu. Dit submenu wordt ook weergegeven als een radiaal menu. Selectie in het submenu gebeurt net als hogerop beschreven voor een niet-hiërarchisch menu. Het werken met submenu's kan ook in *mark mode*: de gebruiker hoeft enkel de *marks* te tekenen (zonder wachttijd) die overeenkomen met het pad dat gevolgd zou zijn in *menu mode*. De menu's in Figuur 4.2 zijn trouwens hiërarchische menu's.

Marking menu's in *mark* modus doen in sterke mate een beroep op het geheugen van de gebruiker: hij krijgt bij het maken van de *marks* volgens de beschreven methode immers geen feedback over de opties die hij kiest. Hierom is de techniek uitgebreid met de zogenaamde *mark confirmation*: wanneer de gebruiker bevestiging wil van de keuze die hij op het punt staat te maken (dus voor de stylus van het scherm weg bewogen wordt), worden na een korte wachttijd (even lang als de periode bij *press-and-wait*) de radiale menu's getoond alsof de gebruiker de menu-keuzes gemaakt heeft in *menu* mode.

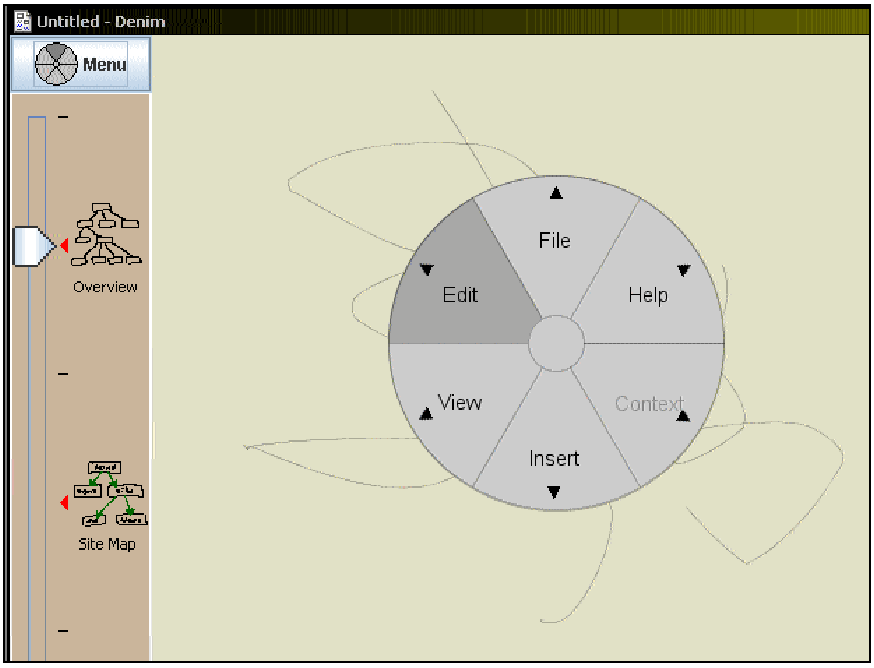
4.2.1.2 Pie menu

Een *pie* menu is een soort van radiaal menu, met andere woorden: de menu-items worden in een cirkel rond de cursor geplaatst. De oorsprong van *pie* menu's ligt in een zeer ver verleden: reeds in 1969 stelden Wiseman, Lemke en Hiles het radiale menu voor als interactietechniek in [21]. Meer in het recentere verleden heeft Don Hopkins veel werk verricht rond *pie* menu's en kan men *pie* menu's niet enkel

terugvinden in de wetenschappelijke literatuur, maar ook in computerspellen als 'The Sims' en de open source web browser Mozilla [22].

De menu-items hebben de vorm van de stukken van een taart, vandaar de naam, met de cursor als middelpunt van de 'taart' [22]. Het oproepen van een *pie* menu gebeurt door het aanraken van het scherm met de stylus. Het centrum van het *pie* menu is niet actief zodat nogmaals klikken op deze locatie niet leidt tot een selectie van een menu-item, maar het menu doet verdwijnen.

De hoeveelheid beweging die nodig is om een menu-item te selecteren is wegens het cirkelvormige karakter van het *pie* menu minimaal en bovendien is elk menu-item even ver verwijderd van de muiscursor, dit in tegenstelling tot traditionele, lineaire menu's. Hoe verder de muis of stylus bewogen wordt weg van het centrum, hoe makkelijker het voor de gebruiker is om de juiste menu-optie te kiezen. Op Figuur 4.3 is een voorbeeld van een *pie* menu in actie getoond in de DENIM-applicatie, een applicatie om softwareprototypes mee te maken [23].



Figuur 4.3 Pie menu [23]

Net als bij *marking* menu's bestaan er ook implementaties van *pie* menu's [22] waarbij er niet altijd visuele feedback wordt gegeven bij de menu selectie. Hiermee bedoel ik dat dan het menu enkel getoond wordt indien men na een bepaalde tijdsperiode (bij *marking* menu *press-and-wait*) nog geen beweging heeft gemaakt en dat men ook selecties kan maken voordat het menu wordt getoond (in [22] wordt dit *marking ahead* genoemd).

Pie menu's kennen een aantal varianten. Een eerste variant is het *pie* menu dat niet uit een volledige 'taart' bestaat, maar uit een halve cirkel. Deze halve cirkel kan dan tegen de zijkant van het scherm (of van een venster) geplaatst worden. Bij een andere variant kunnen twee parameters bepaald worden op basis van de richting en

de afstand van het centrum, wat interessant is wanneer er twee parameters zijn en het maakt daarbij niet uit of de parameters discreet of continu zijn. Zo zou bijvoorbeeld bij een tekstverwerkingsprogramma de keuze van het lettertype en de grootte ervan samen kunnen gebeuren door gebruikt te maken van een *pie* menu: de afstand tot het centrum bepaalt dan bijvoorbeeld de grootte van het lettertype en de richting die de gebruiker kiest, bepaalt het soort lettertype.

De nadelen van *pie* menu's zijn onder andere dat ze vaak nogal veel ruimte innemen op het scherm, zeker wanneer er een groot aantal menu-items moet getoond worden en deze menu-items ook nog eens voorzien moeten worden van een passende tekst [22].

Een ander pijnlijk probleem van *pie* menu's is ook inherent aan het feit dat de menu-items in een cirkel rondom de muiscursor geplaatst worden: de ontwerper van de GUI moet er rekening mee houden dat –met name bij een absoluut invoerapparaat, zie paragraaf 4.1.1.1- ook dicht tegen de zijkant van het scherm of venster een *pie* menu getoond moet kunnen worden. Wanneer immers het middelpunt van de cirkel zich op een afstand van de zijkant bevindt die kleiner is dan de straal van de cirkel, zal een gedeelte van de cirkel buiten het scherm vallen en dus niet zichtbaar zijn. Bij hiërarchische menu's wordt dit probleem duidelijk nog groter omdat er meerdere menu's getoond kunnen worden, vertrekkende van de positie waar de gebruiker het menu heeft opgeroepen.

De genoemde nadelen gelden uiteraard ook voor andere radiale menu's.

In 1988 reeds onderzochten onderzoekers in [24] de performantie van *pie* menu's: in een vergelijking met lineaire menu's kwamen zij tot de vaststelling dat *pie* menu's voor selectie significant sneller waren (15%) en minder gevoelig waren voor fouten (42%) dan de traditionele lineaire menu's.

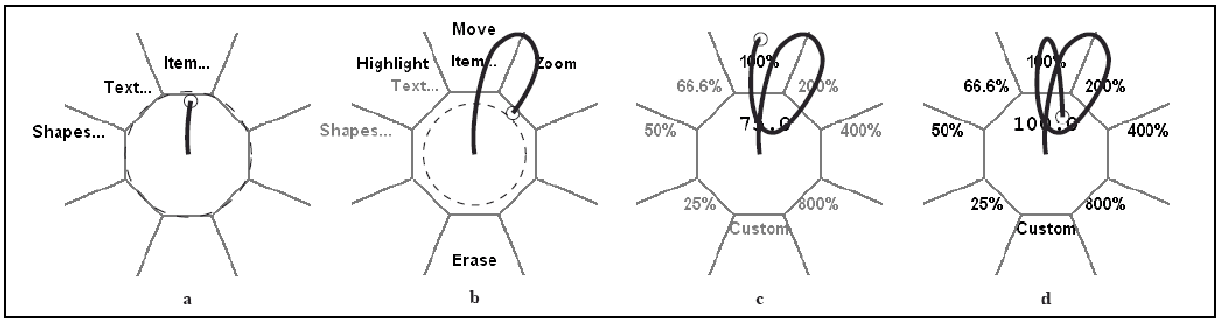
4.2.1.3 FlowMenu

De makers noemen FlowMenu een *command-entry system* dat geschikt is voor interactieve schermen waarbij invoer via een stylus gebeurt. FlowMenu is ontworpen voor de *Interactive Mural* (zie iRoom paragraaf 3.6), maar kan gebruikt worden voor elk apparaat dat stylusgebaseerde invoer ondersteunt, zij het directe of indirecte invoer (zie paragraaf 4.1.1.1).

FlowMenu integreert de selectie van commando's (menu-selectie), tekstinvoer en parameteraanpassing in één enkel mechanisme [26]. Net als *pie* menu's en *marking* menu's gebeurt bij FlowMenu de selectie van menu-items op de plaats waar de gebruiker bezig is met zijn stylus (*point of focus*) in plaats van in een menu dat zich op een vaste plaats (bijvoorbeeld bovenaan het scherm) bevindt.

Het FlowMenu is een radiaal menu dat bestaat uit acht sectoren (octanten) en een rustgebied in het midden (zie Figuur 4.4). Vertrekkende vanuit het rustgebied kiest de gebruiker een menu-item door de stylus naar het betreffende octant van zijn keuze te bewegen. Hierna toont het FlowMenu indien nodig het juiste submenu in de betreffende octanten en plaatst de teksten van deze submenu-items iets verder weg

van het centrum. Tegelijkertijd worden de andere submenu's in het lichtgrijs weergegeven. Selectie van een submenu-item gebeurt vervolgens door de stylus terug naar het centrale rustgebied te bewegen vanuit het gekozen octant waarna het FlowMenu verdwijnt. De gebruiker kan de selectie afbreken door gewoon de stylus weg van het scherm te bewegen (op te heffen van het scherm).

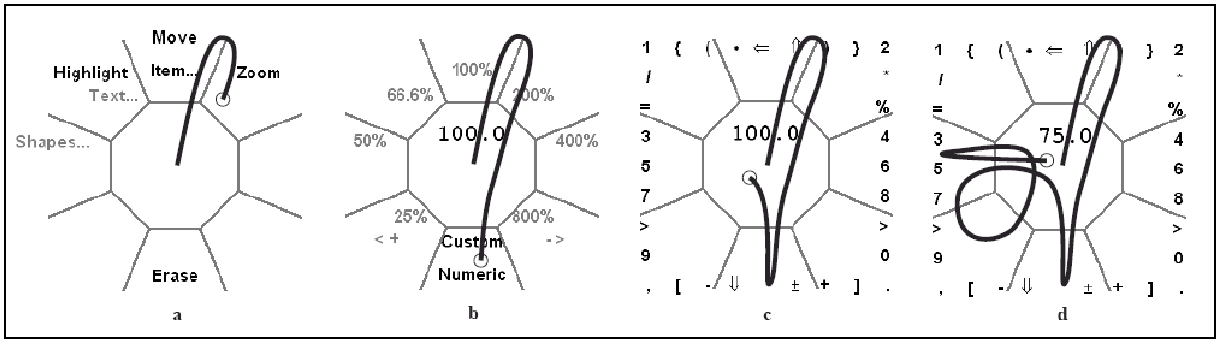


Figuur 4.4 FlowMenu [26]

Nemen we Figuur 4.4 als voorbeeld, dan zien we dat de gebruiker de stylus op het scherm heeft gedrukt waarna het FlowMenu verschijnt. Hierna beweegt de gebruiker -met de stylus nog steeds tegen het scherm- de stylus naar het octant dat de tekst 'Item…' draagt (a). Hierna wordt het submen dat bij 'Item…' hoort zichtbaar: 'Highlight', 'Move', 'Zoom' en 'Erase' (b). Merk op dat de menu's met de teksten 'Shapes…' en 'Text…' nu lichtgrijs zijn weergegeven (die opties waren immers toch niet gekozen). De gebruiker kiest in het voorbeeld voor het submenu 'Zoom' en keert vervolgens terug naar het rustgebied (b).

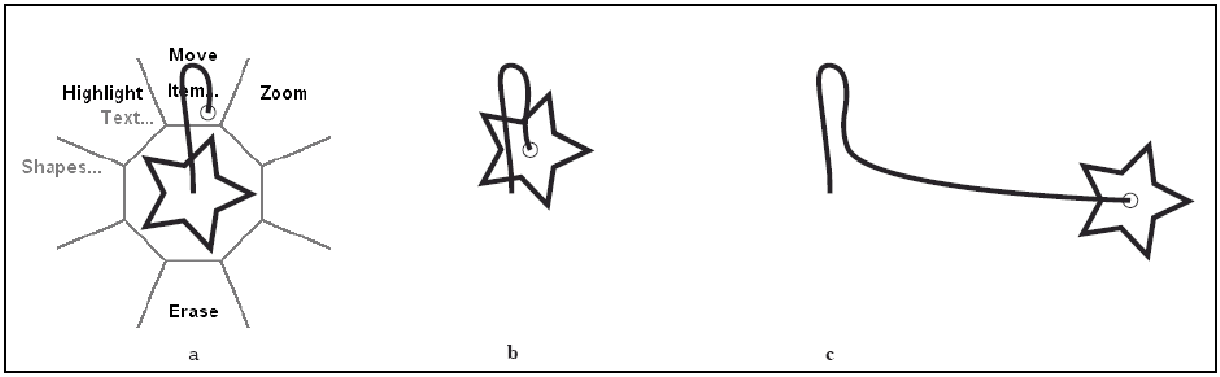
De gebruiker heeft in FlowMenu toegang tot acht octanten (voor hoofdmenu's) met telkens nog eens acht octanten (voor submenu's). Aangezien een selectie telkens eindigt met de muiscursor in het rustgebied kunnen menu's met een willekeurige diepte (wat betreft aantal hiërarchische niveaus) aan elkaar geregen worden. Dit is ook te zien op Figuur 4.4: nadat de gebruiker het menu-item 'Zoom' heeft geselecteerd, verschijnt een menu waarin hij kan kiezen tussen verschillende vergrotingswaarden (c). Wanneer de gebruiker terugkeert naar het rustgebied wordt het gekozen zoomniveau toegepast op het geselecteerde object, maar het menu wordt getoond tot de gebruiker de stylus beweegt weg van het scherm. Op deze manier kunnen verschillende waardes voor een bepaalde parameter uitgeprobeerd worden (d).

Het invoeren van parameters en het selecteren van menu-opties is op deze manier eenvoudig omdat deze twee acties van elkaar gescheiden zijn door het terugkeren met de muiscursor naar het rustgebied. Op deze manier kan de gebruiker ook alfanumerieke parameters invoeren, zoals te zien is in Figuur 4.5. Na de keuze voor 'Custom Numeric' toont het FlowMenu het menu waar een getal als zoomfactor mee ingevoerd kan worden, in dit geval is dit submenu geïmplementeerd met het *Quickwriting* systeem [25].



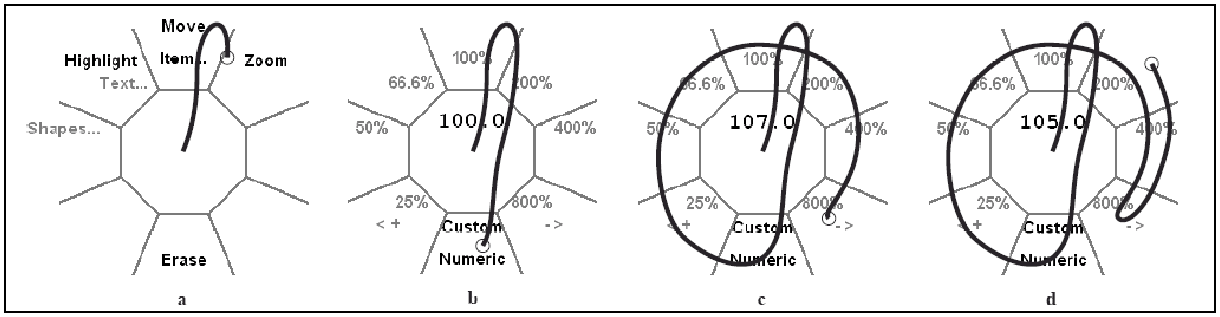
Figuur 4.5 FlowMenu: alfanumerieke invoer [26]

FlowMenu maakt ook de combinatie van menu-selectie en objectmanipulatie mogelijk. In Figuur 4.6 kiest de gebruiker (na het oproepen van het FlowMenu voor het object) voor de optie om het object te verplaatsen (a). Na deze selectie verdwijnt het FlowMenu, verplaatst het object zich naar de positie van de stylus (b) en kan de gebruiker het object naar believen verplaatsen (c).



Figuur 4.6 FlowMenu: objectmanipulatie [26]

Tot slot kan het FlowMenu ook gebruikt worden als een soort van draaiknop (*knob mode*) waarbij de waarde van de parameter verhoogd of verkleind wordt wanneer men van het ene octant naar het andere gaat in wijzerzin respectievelijk tegenwijzerzin. Zo is in Figuur 4.7 te zien hoe de gebruiker na selectie van zoomfactor '100.0' (b) zijn stylus beweegt van het octant 'Custom Numeric' in wijzerzin tot het octant '800%' en hierbij zeven octanten tegenkomt en dus de zoomfactor zeven maal verhoogt met 1. In (d) 'draait' de gebruiker twee octanten terug, wat resulteert in een waarde van '105.0'.



Figuur 4.7 FlowMenu: knob mode [26]

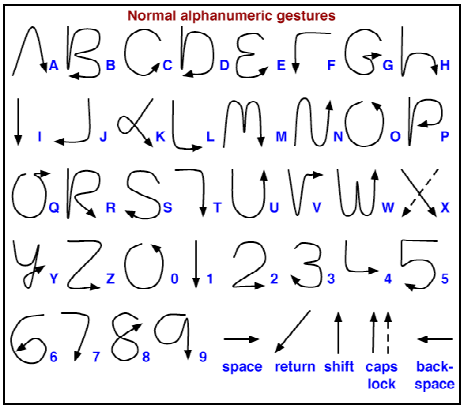
De makers van FlowMenu proberen het gebruik te vermijden van het octant dat zich rechtsonder bevindt omdat het geblokkeerd wordt door de hand van de gebruiker en daardoor niet zo heel goed zichtbaar is voor de gebruiker. Dit is trouwens een probleem dat geldt voor alle radiale menuvormen. Voor linkshandige gebruikers geldt uiteraard dat het octant linksonder vermeden moet worden.

In [27] onderzocht François Guimbretiere de performantie van het FlowMenu. Bij een experiment kregen proefpersonen een reeks punten op het scherm die ze met elkaar moesten verbinden en van een bepaalde kleur moesten voorzien. Uit het experiment bleek dat het FlowMenu significant sneller bleek te werken dan een klassiek *tool palet*. Met het FlowMenu konden de gebruikers de kleur kiezen op de *locus of attention* terwijl hiervoor met het *tool palet* een verplaatsing met de stylus nodig was naar het *palet*. De te verwachten snelheidswinst die gehaald kan worden door het integreren van selectie en directe manipulatie werd hiermee dus aangetoond.

4.2.2 Gestures

Letterlijk uit de Engelse taal vertaald, zijn *gestures* gebaren. Toegepast op het domein van HCI, is een *gesture* met een pen een soort tekening die men maakt, opgebouwd uit korte lijnen (*strokes*), die niet als een onderdeel van de tekening moet gezien worden, maar als een commando aan het interactief systeem. Niet enkel de soort van de operatie (commando) kan aan het systeem doorgeven worden, maar ook -en in dezelfde beweging- de operand (het object waarop het commando van toepassing is) en de parameters van de operatie.

Een bekend voorbeeld van *gestures* is Grafitti waarmee gebruikers van een Palm PDA (dat niet over een toetsenbord beschikt) invoer via het toestenbord kunnen simuleren (zie Figuur 4.8) door het gebruik van *gestures* [28].



Figuur 4.8 Graffiti gestures [40]

Invoermechanismen die gebruik maken van *gestures* bieden als belangrijk voordeel dat zij het mogelijk maken om sneller commando's aan het systeem te geven. Een *marking* menu bijvoorbeeld (zie 4.2.1.1) gebruikt *gestures* en is sneller dan een traditioneel menu onder andere omdat de af te leggen afstand kleiner is, want de verschillende menu-items bevinden zich immers in een cirkel rond de cursor.

In dit verband stelde Fitts reeds in 1954 dat de tijd die nodig is om vanuit een bepaalde startpositie naar een bepaald doelgebied te bewegen (om bijvoorbeeld een menuselectie te maken) afhankelijk is van de afstand tot en de grootte van dit doelgebied, wat in de literatuur bekend staat als de *wet van Fitts* [29].

Een van de problemen bij *gestures* is het onderscheid maken tussen een *gesture* (die door het systeem geïnterpreteerd moet worden) en een tekening (die getoond moet worden op het scherm). In de praktijk kan het maken van dit onderscheid aan de computer overgelaten worden, maar een andere manier is het onderscheid door de gebruiker te laten maken: voor het invoeren van *gestures* moet hij dan bijvoorbeeld eerst een bepaalde handeling uitvoeren (het drukken op een bepaalde knop bijvoorbeeld) zodat het systeem de *gesture* verwacht. Een andere oplossing is een speciaal daarvoor voorziene ruimte voor *gestures*: alles wat hier getekend wordt, is voor het systeem een *gesture*, en de rest is tekening.

Het herkennen van *gestures* door de computer is niet eenvoudig en er zijn vele manieren om dit te doen. Dit kan onder andere met patroonherkenning, neurale netwerken en statistische classificatie zoals bijvoorbeeld de herkenningsmethode van Rubine [30].

Gestures worden tegenwoordig gebruikt in verschillende spellen (bijvoorbeeld het spel *Black and White)*, maar ook in populaire toepassingen zoals bijvoorbeeld de webbrowser Mozilla Firefox. Zo is op Figuur 4.9 de *gesture* weergegeven (oranje lijn) die ervoor zorgt dat de browser de link opent die door de *gesture* gekruist wordt (link: 'Anno Domini').

Figuur 4.9 Gestures in FireFox [41]

4.3 Floor control

4.3.1 Definitie

In [31] wordt *floor control* als volgt gedefinieerd: "… *the protocol which determines which user has control and how to take turns when multiple people share a limited resource such as a single cursor in a synchronous task."* Floor control is dus een mechanisme dat bepaalt welke gebruiker controle heeft, en hoe de andere gebruikers die controle kunnen veroveren. Essentieel bij floor control is dat hetgeen waarover men controle kan en wil verwerven niet tegelijkertijd kan gebruikt worden door meerdere gebruikers.

Toegepast op interactieve whiteboards gaat het hierbij over een mechanisme waarmee gestuurd wordt wie wanneer invoer kan leveren aan het whiteboard indien dit door meerdere gebruikers gebruikt wordt.

Er zijn whiteboards waarbij het mogelijk is om met meerdere invoerapparaten tegelijk te werken. Dergelijke systemen gebruiken vaak verschillende styli, die door de applicatie ook als verschillende invoerapparaten behandeld worden: aan elke stylus is dan een unieke identificatie verbonden. Het probleem van floor control stelt zich hier dus niet omdat er geen *limited* resource te delen valt.

Vele systemen, waaronder het SmartBoard, ondersteunen echter geen simultane invoer van meerdere styli, en gebruiken dus wel een limited resource, waardoor het probleem van floor control zich wel stelt.

Voor de volledigheid dient bij een behandeling van floor control nog vermeld te worden dat ook de fysieke ruimte aan het bord een limited resource is die gedeeld moet worden door meerdere gebruikers, maar in deze paragraaf beperk ik me tot de pen als middel tot invoer van informatie als limited resource.

4.3.2 Toepassingsgebied

Refererend aan de paragraaf over synchrone en asynchrone collaboratie (paragraaf 2.4), kennen niet alle vormen van collaboratie het probleem van floor control: bij asynchrone collaboratie is er per definitie geen conflict over een invoerapparaat omdat de interactie van de verschillende deelnemers niet tegelijkertijd gebeurt. Floor control kan echter wel gebruikt worden voor samenwerking die synchroon verloopt. Wanneer bijvoorbeeld tijdens een vergadering een prototype van een ontwerp wordt gepresenteerd op een laptop en de deelnemers hierbij regelmatig het prototype willen bewerken, dan moeten de deelnemers dit om beurten doen.

Wat de ruimtelijke context van collaboratie (zie paragraaf 2.5) betreft, is er geen verschil: het probleem van floor control kan zich bij beide vormen stellen. Wanneer de deelnemers zich in dezelfde ruimte bevinden (co-located), is het duidelijk dat floor control een probleem is: de deelnemers moeten de muiscursor delen. Maar ook bij gedistribueerde collaboratie komt floor control om de hoek kijken. Wanneer de deelnemers bijvoorbeeld een gedeeld werkblad gebruiken kan dit tot problemen leiden.

4.3.3 Technieken

Er zijn verschillende technieken om het probleem van floor control aan te pakken. In deze paragraaf worden een aantal van die technieken besproken. Technieken voor floor control zijn op te delen langs drie dimensies [31]:

- hoe geven deelnemers de controle af?
- hoe verwerven deelnemers de controle?
- wat gebeurt er als controle gevraagd wordt?

De manier waarop de deelnemers aan een collaboratieve taak controle afgeven kan op verschillende manieren gebeuren. De gebruiker kan deze controle actief afgeven, en deze controle kan ook van de gebruiker passief afgenomen worden (verliezen).

Bij het verliezen van controle is er een mechanisme dat bepaalt wanneer dit gebeurt (een time-out of moderator). Bij het actief afgeven van controle kan dit door de gebruiker aangegeven worden (de gebruiker drukt bijvoorbeeld op een knop om de controle af te geven) of niet (de applicatie detecteert bijvoorbeeld geen beweging meer van de gebruiker en leidt hieruit af dat de gebruiker geen controle meer nodig heeft).

Eens de controle vrijgegeven is, kan deze weer aan een (niet noodzakelijk andere) gebruiker toegewezen worden. Dit toewijzen of verwerven van de controle kan ook weer op verschillende manieren gebeuren: via een moderator, door het drukken op een knop, door detectie van beweging of op basis van regels (zoals bijvoorbeeld *round robin*).

Tot slot bestaan er ook technieken die focussen op wat er gebeurt wanneer een gebruiker controle vraagt en deze niet beschikbaar is, met andere woorden: of, en hoe de aanvraag afgehandeld wordt. Dit verzoek om controle kan onmiddellijk toegewezen worden, via een wachtrij, maar kan ook genegeerd worden.

Door het combineren van de hierboven geschetste technieken, kunnen bestaande technieken van floor control opgebouwd worden zoals in [31]. Zo ook bijvoorbeeld de floor control techniek *pause detection*: bij deze techniek wordt impliciete vrijgave (de applicatie tracht te detecteren wanneer de gebruiker de controle niet meer nodig heeft aan de hand van de beweging van de gebruiker) gecombineerd met impliciete aanvraag (weer op basis van de beweging van de gebruiker) en het negeren van verzoeken. Wanneer een (controle hebbende) gebruiker klaar is met zijn acties, wordt de controle automatisch vrijgegeven en kan een andere gebruiker controle verkrijgen. Wanneer een gebruiker iets wil doen terwijl de controle niet vrij is, wordt dit genegeerd.

4.3.4 Onderzoek

Onderzoekers aan de Carnegie Mellon University hebben aan de hand van een eenvoudige studie een aantal van de in de vorige paragraaf vermelde technieken onder de loep genomen [31]. Hiervoor werd aan een aantal proefpersonen de opdracht gegeven samen te werken aan een bepaalde taak, meerbepaald het maken van een legpuzzel op een computer. Tijdens de studie trachtten de onderzoekers zowel het co-operatieve als het competitieve aspect te benadrukken door de proefpersonen als groep financieel te belonen indien men binnen een bepaalde tijd de puzzel kon maken, en ook elke persoon individueel naar gelang het aantal stukjes de persoon zelf legde.

Om deze taak uit te kunnen voeren beschikte elke proefpersoon over een PDA die met de computer verbonden was met een kabel en waarmee de muiscursor van de computer bewogen kon worden. Voor de duidelijkheid: het probleem van floor control stelde zich in deze situatie dus omdat elke proefpersoon de (enige) muiscursor kon aansturen. De proefpersonen waren gemiddeld 23 jaar oud, kenden elkaar, en hadden geen of weinig ervaring met het werken met een PDA.

Er werden een aantal verschillende floor control technieken uitgeprobeerd, maar ook een aantal verschillende controle-situaties waarbij geen floor control techniek gebruikt werd, zoals bijvoorbeeld de situatie waarbij elke gebruiker wél over zijn eigen individuele muiscursor beschikte die hij onafhankelijk van de andere gebruikers kon gebruiken (en dus niet zorgde voor floor control problemen).

Een van de belangrijkste conclusies van dit onderzoek is dat bij taken waarbij de deelnemers van een gemeenschappelijke taak in parallel aan hun deel van de taak werken, het de meest effectieve (snelste) techniek blijkt te zijn om elke gebruiker zijn eigen individuele muiscursor te geven. Van alle floor control technieken die in het onderzoek aan bod kwamen, kwam geen enkele techniek echt naar voor als meest effectieve. Bovendien werd ook de voorkeur van de proefpersonen voor de verschillende technieken via een vragenlijst onderzocht en bleek ook deze voorkeur niet bepaald uitgesproken te zijn in de richting van een bepaalde techniek.

4.4 Taken op een interactief whiteboard

4.4.1 Typische taken

Elizabeth D. Mynatt heeft onderzoek gedaan naar wat precies de typische taken zijn waarvoor een whiteboard zoal gebruikt wordt [6]. Zij kwam tot de vaststelling dat whiteboards doorgaans gebruikt worden voor de volgende soorten taken: *thinking tasks* en *quick capture tasks*.

Quick capture tasks

Het bord wordt bij dit soort taken gebruikt om bijvoorbeeld snel even een telefoonnummer of een URL te noteren, die tijdens een gesprek naar boven is gekomen.

In de vorm van een *to-do* lijst of gewoon wat minder gestructureerde notities op het bord, plaatst de gebruiker ook herinneringen op het bord voor toekomstige taken [6]. Hierdoor hoeft de gebruiker hier niet meer aan te denken en kan hij zich op andere taken concentreren. Bij herinneringen aan taken die op whiteboards geplaatst worden, gaat het doorgaans om taken op korte termijn.

Thinking tasks

Bij dit soort taken wordt het whiteboard gebruikt als hulpmiddel bij denkprocessen (*pre-production tasks)*, waarbij de nadruk ligt op het begrijpen van ideeën en concepten. Het effectief uitwerken van deze ideeën gebeurt vervolgens op een andere manier (en via een ander medium). Voorbeelden van *pre-production tasks* zijn het uitwerken van een algoritme dat later gecodeerd wordt op een desktop computer, het opmaken van een planning voor een bepaalde taak, het noteren van ideetjes voor de inhoud van een tekstdocument. Het resultaat van deze taken kunnen we geen kladversie noemen, maar het is hetgeen wat nog vóór de kladversie komt.

Terwijl bij het onderzoek van Elizabeth Mynatt in [6] eerder het individuele, informele gebruik van whiteboards aan bod kwam, is Tivoli (zie paragraaf 3.3 en [12]) specifiek bedoeld om informele vergaderingen van een beperkt aantal personen te ondersteunen. Volgens de ontwikkelaars van Tivoli is de primaire functie van het *LiveBoard* (een interactief whiteboard) het ondersteunen van interactie tussen verschillende personen. En zoals ik reeds aanhaalde in paragraaf 3.3, schuiven zij als voornaamste criteria voor de gebruikersinteractie naar boven dat de interactie onbewust en vloeiend moet gebeuren.

5 Interactieproblemen

In dit laatste hoofdstuk komen de problemen aan bod die zich kunnen stellen bij het werken met interactieve whiteboards. In een eerste paragraaf wordt ingegaan op de problemen van de klassieke whiteboards omdat dit kan helpen bij het begrijpen van de problemen van de interactieve variant. Vervolgens worden enkele problemen behandeld die optreden bij het gebruik van grote beeldschermen.

In een volgende paragraaf komt het probleem van de tekstinvoer aan bod. Verder gaat het in dit hoofdstuk nog over de mate waarin interactieve whiteboards meerdere gebruikers ondersteunen en tot slot wordt nog een korte paragraaf gewijd aan de drempelvrees van gebruikers om aan de slag te gaan met interactieve whiteboards.

5.1 Problemen van een klassiek whiteboard

Met klassieke whiteboards worden in deze context whiteboards aangeduid die niet over interactieve eigenschappen beschikken. Het gaat hier om eenvoudige borden die meestal een glimmend -meestal wit- oppervlak hebben waarop met een speciale stift getekend kan worden. De markeringen en tekeningen die met deze stiften gemaakt zijn, kunnen gemakkelijk worden verwijderd. Vaak zijn de borden tevens magnetisch.

Omdat dergelijke klassieke whiteboards de voorloper zijn van de interactieve whiteboards, loont het de moeite om even stil te blijven staan bij de tekortkomingen van deze klassieke whiteboards, want die tekortkomingen verklaren voor een deel het ontstaan van de moderne interactieve whiteboards.

Een eerste nadeel is dat men speciale stiften nodig heeft om op klassieke whiteboards te gebruiken. Deze stiften verspreiden een sterke geur, en mocht men per vergissing een verkeerde stift gebruiken dan zijn de gemaakte markeringen niet of nauwelijks van het bord te verwijderen.

Nog een nadeel is dat men met een in de ruimte beperkte oppervlakte werkt. Dit lijkt niet zo vanzelfsprekend, maar ook al is een klassiek whiteboard op zich relatief groot (zeker ten opzichte van een blad papier of een computerscherm), toch moeten gebruikers doorgaans al snel vaststellen dat ze te weinig plaats hebben voor al hun aantekeningen.

Dat deze beperking van de ruimte een ernstig probleem vormt, is onder andere op te maken uit het feit dat klassieke schoolborden (ook al is dit een onderwijzend hulpmiddel dat meestal slechts door één persoon gehanteerd wordt) vaak uit verschillende verschuifbare of openklapbare panelen bestaan, en het bestaan van de zogeheten *flip-charts*: dit lijken wel schildersezels, maar ze hebben vellen papier die als een leeg boek op het bord bevestigd zijn, waarvan men naar believen de bladzijden kan omslaan.

Een ander nadeel is dat de informatie die na een brainstormingsessie bijvoorbeeld op het bord is verschenen (de 'vrucht' van de taak) bijzonder moeilijk te transporteren is naar een ander bruikbaar medium. Een gemaakt diagram waar men misschien wel verschillende uren aan gezwoegd heeft, moet gefotografeerd of overgetekend worden om bewaard te worden, wat een extra last is voor de deelnemers aan de taak. Zeker wanneer er regelmatig gewist wordt om plaats te maken op het bord, is het moeilijk om alle belangrijke informatie over te schrijven of te tekenen. Vaak is dit laatste dan ook de taak van een vooraf aangewezen persoon die verder weinig inbreng kan hebben in de eigenlijke taak, net zoals een notulist bij een vergadering een verslag neerschrijft.

Een ander nadeel dat zich vandaag de dag meer en meer laat voelen, is dat er bij een klassiek whiteboard geen mogelijkheid bestaat om digitale informatie weer te geven, laat staan te bewerken. Een grafiek moet afgedrukt worden om op het bord te kunnen laten zien en de gegevens die achter de grafiek zitten kunnen niet aangepast worden. Het gebrek aan ondersteuning voor digitale informatie impliceert ook dat er geen connectiviteit mogelijk is: vanop afstand deelnemen aan een brainstormingsessie is er niet bij.

5.2 Problemen grote schermen en mogelijke oplossingen

Bij het gebruik van interactieve whiteboards hebben we de beschikking over een veel groter schermoppervlak dan bij normale desktopschermen, waardoor we informatie veel groter kunnen weergeven, en -bij grotere resolutie- meer informatie tegelijk kunnen tonen op het scherm. Bij het ontwerpen van interactie met dergelijke beeldschermen kunnen we echter niet zomaar de interface reproduceren van een desktopscherm. Er kleven ook nadelen aan het gebruik van grote schermen waar we ons bewust van moeten zijn [32].

5.2.1 Muiscursor uit het oog verliezen

Een eerste probleem dat zich stelt heeft te maken met de muis als invoerapparaat. Wanneer de muis gebruikt wordt op een zeer groot vlak, dient men de relatie tussen beweging van de muis en beweging van de muiscursor aan te passen. Door het aanpassen van deze relatie gaat de muiscursor sneller bewegen, met het risico dat de gebruiker de muiscursor uit het oog verliest. Bovendien is het op een groot scherm moeilijker om een stilstaande muiscursor te lokaliseren.

Een mogelijke oplossing voor dit probleem is het tonen van een spoor van de muis (*mouse trail*). Hierbij is het echter van groot belang dat het muisspoor gedetailleerd genoeg wordt weergegeven (genoeg tussenpunten). Een oplossing voor het zoeken van een stationaire muiscursor is deze te accentueren wanneer de gebruiker ernaar op zoek is: de gebruiker drukt dan op een bepaalde toets en er verschijnt bijvoorbeeld een fel gekleurde cirkel rond de muiscursor [32].

5.2.2 Manipulatiebereik

Bij een interactief whiteboard kan het gebeuren dat een gebruiker aan de rechterkant van het scherm bezig is, en een icoon wil aanklikken dat zich helemaal aan de linkerkant bevindt: de gebruiker moet zich dan naar de linkerkant verplaatsen. Dit

probleem stelt zich bij interactieve whiteboards vaak bij traditionele menusystemen zoals menubalkjes die bovenaan het scherm geplaatst zijn.

Om het bereik van de muiscursor te vergroten, bestaan een aantal oplossingen [32]. Via een bepaald commando kan bijvoorbeeld een virtuele *ray* (straal) gestuurd worden vanuit de huidige muispositie, en wanner deze straal een object raakt, kan dit object geselecteerd worden.

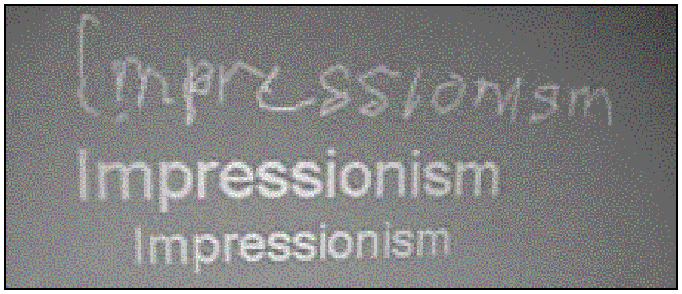
Een variant hierop is dat in plaats van een straal, er door middel van een korte muisbeweging een soort van sprong gemaakt wordt met de muiscursor, maar deze laatste variant is niet ideaal omdat een slecht resultaat (met name naast het doelobject springen) pas zichtbaar wordt als het reeds te laat is (de locatie van de muiscursor is reeds veranderd), terwijl bij de straal wel feedback mogelijk is alvorens de actie uit te voeren: een object dat door de straal geraakt wordt, licht bijvoorbeeld op in een bepaalde kleur.

Twee andere oplossingen manipuleren niet zozeer de muiscursor, maar wel de omgeving waarin de doelobjecten zich bevinden. Bij *drag-and-pop* verplaatsen doelobjecten die relevant zijn zich tijdelijk tot kort bij de muiscursor. Bij de *tablecloth* techniek kan de desktop verschoven worden tot het gewenste object zich vlak bij de muiscursor bevindt: bij het bewegen van de muis blijft de muiscursor stationair maar de rest van de desktop wordt verschoven en volgt de beweging van de muis [32].

Bovenstaande oplossingen lenen zich minder goed of helemaal niet goed voor de toegang tot menu's. Hiervoor zijn dan ook andere oplossingen bedacht. Een van die oplossingen is een soort van contextmenu, waarbij het menu verschijnt op de plaats waar de gebruiker geklikt heeft zoals beschreven in paragraaf 4.2.1.

5.3 Tekstinvoer

Wanneer gebruikers met een interactief whiteboard werken, gebruiken ze vaak een pen om tekst op het scherm te schrijven, net zoals dat ook op een klassiek whiteboard gebeurt. De elektronische, interactieve whiteboards zijn niet bijzonder goed uitgerust om deze elementaire activiteit te ondersteunen, zo stelt Rekimoto in [1]. Het schrijven op interactieve whiteboards gaat niet zo goed als bij de klassieke variant omwille van de beperkte nauwkeurigheid: de geschreven teksten zien er minder vloeiend uit en zijn ook moeilijker te lezen dan op een klassiek whiteboard (zie Figuur 5.1).



Figuur 5.1 Geschreven tekst op een whiteboard [1]

Dat de tekst moeilijker te lezen is (omdat de tekst minder zuivere contouren heeft) bemoeilijkt trouwens ook de taak van eventuele software voor automatische (handgeschreven) tekstherkenning (*OCR, optical character recognition*). Bovendien zorgt het feit dat de geschreven teksten er niet zo mooi en vloeiend uitzien er ook voor dat gebruikers niet snel geneigd zijn om voor een -al is het beperkt- publiek tekst op het bord te schrijven en zich beperken tot diagrammen en schema's [1].

Er bestaan interactieve whiteboards die uitgerust zijn met een toetsenbord dat net onder het scherm geplaatst wordt (zoals bijvoorbeeld het Xerox LiveBoard), maar deze opstelling zorgt ook voor problemen want slechts één gebruiker tegelijkertijd kan tekst invoeren en moet hiervoor bovendien een behoorlijk lastige lichaamshouding (voorovergebogen) aannemen. Daarenboven moet de gebruiker ook expliciet aangeven waar op het scherm de tekst moet verschijnen.

Voor informele vergaderingen is het niet echt een probleem dat het niet zomaar mogelijk is om karakters neer te schrijven die door OCR herkend kunnen worden. In dat soort van situaties kunnen handgeschreven teksten volstaan, doch het ontbreken van honderd procent betrouwbare OCR voor handgeschreven tekst –zoals vermeld in [33]- beperkt jammer genoeg het potentieel van elektronische whiteboards. Tekst die wel door OCR herkend wordt, is veel makkelijker te verplaatsen, kopiëren, verwijderen en te doorzoeken dan andere en opent mogelijkheden voor interactie met applicatiesoftware: geschreven getallen worden bijvoorbeeld automatisch bij elkaar opgeteld.

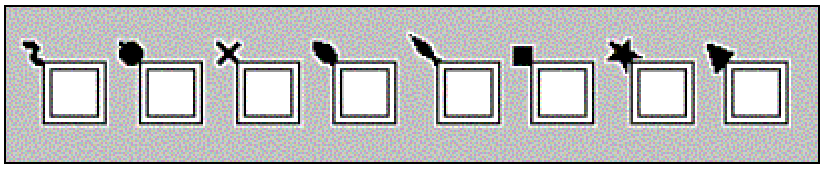
5.4 Ondersteuning van meerdere gebruikers tegelijkertijd

De meeste applicaties voor interactieve whiteboards zijn ontworpen om gebruikt te worden door slechts één gebruiker tegelijkertijd [1]. Bijgevolg is het bij deze applicaties vaak onmogelijk om met meerdere gebruikers simultaan op het whiteboard te werken. Gebruikers kunnen niet samen het menu gebruiken, of tegelijkertijd tekst invoeren op verschillende plaatsen op het whiteboard. Dit gebrek aan ondersteuning voor parallelle activiteiten tussen de verschillende deelnemers werkt niet productief: gebruikers moeten vaak op elkaar wachten, wat de natuurlijke interactie verstoort.

Er zijn echter ook applicaties ontworpen die meerdere gebruikers tegelijkertijd ondersteunen. PebblesDraw is dan misschien wel geen applicatie voor een interactief whiteboard (*Pebbles* is een acroniem voor *PalmPilots for Entry of Both Bytes and Locations from External Sources*), toch confronteerde PebblesDraw zijn ontwikkelaars met een gelijkaardig probleem [11]. Bij PebblesDraw zaten meerdere gebruikers voor een computerscherm en konden zij tekeningen maken op het scherm, maar als invoerapparaat had elke gebruiker zijn eigen PDA: door de stylus over het scherm van zijn PDA te bewegen, bewoog men de cursor op het computerscherm.

De ontwikkelaars van PebblesDraw constateerden dat de gebruikelijke *widgets*, zoals selectiekaders, kleurenpaletten, *pop-up* en *pull-down* menu's niet meer effectief

waren van zodra meerdere gebruikers tegelijkertijd op hetzelfde scherm actief zijn. Het eerste probleem loste men op door aan elke gebruiker een bepaalde vorm te koppelen. Zoals op de afbeelding is te zien, wordt deze vorm aan de selectiekader gekoppeld zodat aan de hand van de bijhorende vorm duidelijk is bij wie de selectiekader hoort.



Figuur 5.2 Selectiekaders meerdere gebruikers [11]

Voor de problemen van de pop-up en pull-down menu's werd slechts voor een gedeelte een oplossing geformuleerd. De applicatie PebblesDraw werd ontworpen om dit soort van menu's gewoon zo veel mogelijk te vermijden. Bovendien ondersteunt PebblesDraw ook *gestural input* (zie paragraaf 4.2.2) waardoor commando's niet langer in een menu gezocht en geselecteerd moeten worden, maar via een *gesture* aan het systeem kunnen doorgegeven worden.

Een ander probleem bij het gebruik van whiteboards door meerdere gebruikers tegelijkertijd is dat het kan voorkomen dat de gebruikers 'elkaar voor de voeten lopen'. De ene gebruiker staat bijvoorbeeld voor het midden van het bord waar hij een tekening aan het maken is, terwijl een andere gebruiker aan de linkerkant van het bord staat en daar een schema aan het opstellen is. Wanneer nu de gebruiker die aan de linkerkant staat zijn schema middels drag-and-drop naar de rechterkant van het bord wil verplaatsen, dan zal hij de andere gebruiker moeten verzoeken even aan de kant (of achteruit) te gaan of moeten wachten tot deze klaar is met zijn tekening vooraleer hij fysiek toegang heeft tot de ruimte die hij op het bord nodig om de gewenste operatie uit te voeren.

5.5 Drempelvrees gebruikers

Nogal wat onderzoeken hebben uitgewezen dat bij de publieke interactie (interactie met grote schermen in openbare ruimten) zich een hardnekkig probleem stelt: het is niet eenvoudig om gebruikers uit te nodigen om tot interactie over te gaan met grote schermen. Er lijkt dus een soort van barrière te bestaan die de mensen afremt. Nochtans stelde Agamanolis reeds in 2002 in [34] het volgende: *"Half the battle in designing an interactive situated or public display is designing how the display will invite that interaction"*. Meer dan genoeg reden dus om ook even stil te staan bij het fenomeen dat in de literatuur sociale schaamte (*social embarrassment*) genoemd wordt.

Onderzoekers van de universiteit van Sussex in Engeland hebben onderzoek gedaan naar waarom vele mensen zich niet zo gemakkelijk laten verleiden tot het gebruik van interactieve grote schermen die in publieke ruimtes zijn geplaatst. Harry Brignull en Yvonne Rogers voerden enkele experimenten uit tijdens informele bijeenkomsten waar zij een interactief groot scherm hadden geplaatst en konden

vaststellen dat de deelnemers drie graden van betrokkenheid vertoonden met het interactief systeem [35]:

- ofwel waren ze zich nauwelijks bewust van het systeem en hielden ze zich bezig met andere zaken,
- ofwel waren gebruikers zich wel bewust van het systeem en praatten ze er ook over, maar gebruikten ze het niet (het was regelmatig onderwerp van gesprek maar meer niet),
- ofwel interageerden de gebruikers rechtstreeks met het interactief systeem (en bevonden ze zich dus voor het scherm).

De onderzoekers stelden ook vast dat gebruikers zich de volgende vragen stelden wanneer ze overgaan tot de hoogste graad van betrokkenheid:

- Hoe lang duurt de interactie?
- Welk voordeel kan ik er uit halen?
- Welke stappen moet ik ondernemen?
- Is het een comfortabele ervaring?
- Kan ik er snel en zonder veel last (voor mezelf en anderen) van wegwandelen als ik wil stoppen?

Ook al ligt bij het onderzoek van Brignull en Rogers de focus voornamelijk op het gebruik van publieke, grote schermen, zijn de vaststellingen ook voor interactieve whiteboards relevant. Het is nuttig te beseffen dat er factoren kunnen zijn waarom mensen geremd kunnen zijn in hun interactie met grote schermen in de nabijheid van andere personen. Ook bij interactieve whiteboards gaat het om grote schermen en zijn er andere personen betrokken. Hierbij dient wel opgemerkt dat bij interactieve whiteboards de mate van vertrouwdheid met de omgeving en de andere personen groter is en dat het doel van de interactie bij whiteboards (collaboratie) en de tijdsduur van de interactie verschillen als we ze vergelijken met de onderzochte activiteiten bij publieke grote schermen.

6 Implementatie

6.1 Inleiding

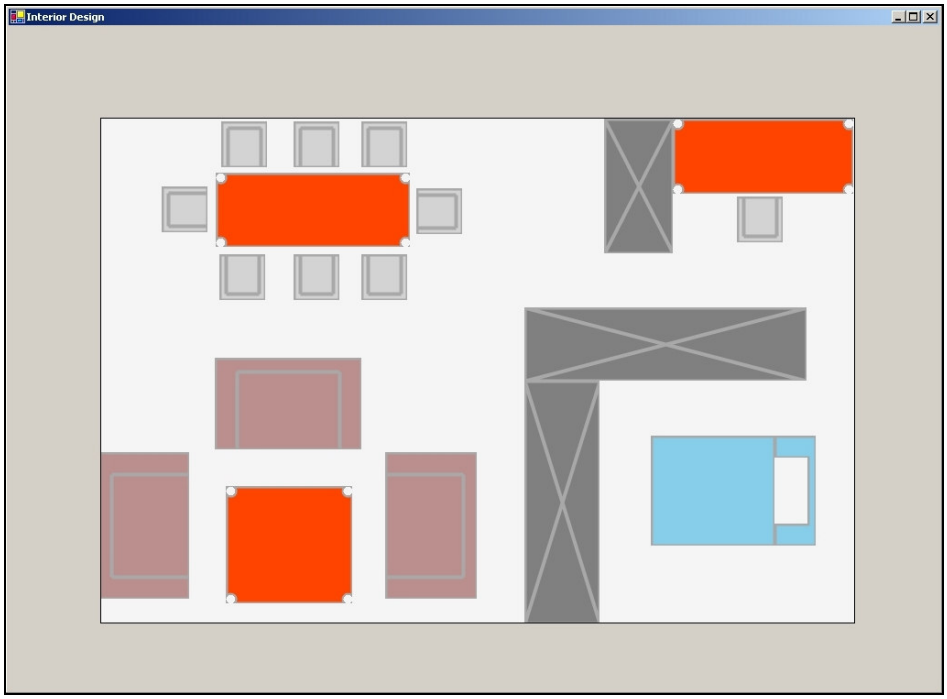
In de voorbije hoofdstukken is getracht een overzicht te geven van de verschillende interactiemogelijkheden van digitale whiteboards zoals ik die in de wetenschappelijke literatuur heb teruggevonden. Ook zijn de problemen besproken die zich kunnen voordoen bij het interageren met deze technologie. In dit hoofdstuk komt het tweede deel van de thesis aan bod, namelijk de implementatie van een applicatie die gebruik maakt van een digitaal whiteboard en waarin verschillende interactietechnieken aan bod komen.

In dit implementatiegedeelte zal eerst een beschrijving gegeven worden van de applicatie die is ontwikkeld. Vervolgens wordt de interactie van de gebruiker met de applicatie behandeld en de verschillende technieken die daarbij gebruikt zijn. Hierbij wordt zowel aandacht besteed aan de concrete implementatie van de verschillende technieken alsook aan de moeilijkheden die daarbij de kop opstaken en de gekozen oplossingen. Een aantal van de interactietechnieken zijn op informele wijze door een aantal gebruikers uitgeprobeerd: een bespreking van deze informele gebruikers-studie is in het volgend hoofdstuk opgenomen.

Aangezien de beschrijving van de interactie met een digitaal whiteboard gemakkelijker met beelden is weer te geven dan met woorden, zijn ter illustratie een aantal schermafdrukken in dit hoofdstuk opgenomen.

6.2 Applicatie

De applicatie is bedoeld voor het ontwerpen van het interieur van een huiskamer (en is aldus gedoopt *InteriorDesign)* met een SmartBoard. Op het scherm wordt de kamer weergeven alsof de gebruiker van bovenaf de kamer bekijkt, zoals op figuur Figuur 6.1 te zien is.



Figuur 6.1 Applicatie: InteriorDesign

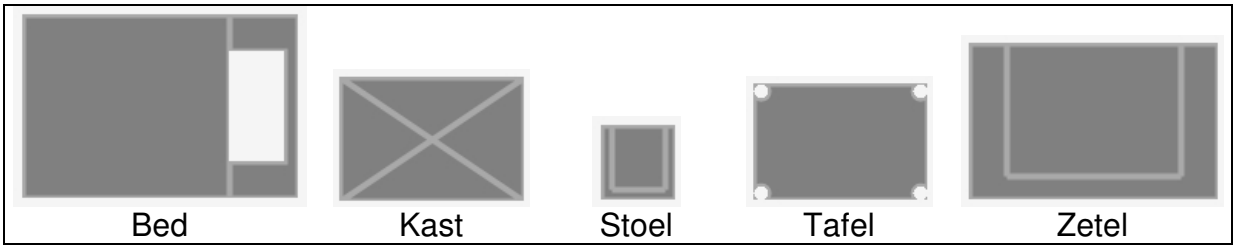
In Figuur 6.1 is verder ook duidelijk te zien dat het programmavenster bestaat uit twee gebieden: een gebied waarin de gebruiker objecten kan plaatsen en manipuleren (het 'canvas' waarin de kamer wordt weergegeven) en een gebied rondom het eerste gebied. Bij andere applicaties (bijvoorbeeld eenvoudige tekenprogramma's) is deze laatste ruimte (de rand rond het feitelijke tekengebied) niet aanwezig. Bij de applicatie die ontwikkeld is, en die gebruik maakt van het FlowMenu is deze ruimte echter noodzakelijk: het FlowMenu moet immers altijd volledig weergeven kunnen worden (voor meer uitleg hierover zie paragraaf 4.2.1).

De applicatie -zoals ze in deze thesis is opgezet- heeft niet als hoofddoel een volwaardig ontwerpprogramma te zijn, maar fungeert eerder als een 'achtergrond' voor de verschillende interactietechnieken die de applicatie biedt en in dit hoofdstuk aan bod komen.

6.2.1 Interieurobjecten

Een kamer kan uit verschillende objecten bestaan (*shapes*) en in de echte wereld is een lijst van dergelijke objecten quasi eindeloos: stoel, tafel, bureau, kast, boekenrek, muziekinstallatie, lamp, radiator, koelkast, enz. Voor de applicatie is ervoor gekozen om het aantal shapes dat de gebruiker tot zijn beschikking heeft, te beperken. De keuze om slechts met een beperkt aantal shapes te werken, is gemaakt omdat een groter aantal weinig toegevoegde waarde zou geleverd hebben aan de applicatie en aan het doel van de applicatie, namelijk de interactie met een digitaal whiteboard.

De applicatie bevat volgende interieurobjecten: bed, kast, stoel, tafel en zetel zoals weergeven in Figuur 6.2.



Figuur 6.2 De verschillende interieurobjecten

De structuur van de interieurobjecten is zeer eenvoudig gehouden: allemaal zijn ze rechthoekig van vorm. Naast de verschillen qua grootte en verhouding tussen breedte en hoogte, heeft elke shape toch een specifiek kenmerk. Zo is bijvoorbeeld bij een bed het hoofdkussen getekend, en zijn bij de kast de diagonalen getekend.

6.2.2 Eigenschappen van de interieurobjecten

Elke shape wordt bepaald door een set van eigenschappen, waarvan de volgende de belangrijkste zijn:

- locatie (ruimtelijke plaats in het interieur)
- kleur
- breedte
- hoogte
- rotatie
- familie

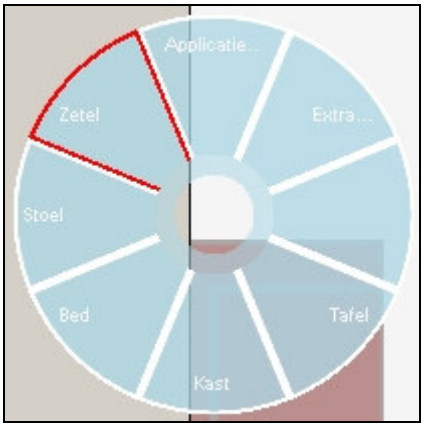
Hoe de gebruiker deze eigenschappen kan wijzigen, wordt verder in de volgende paragraaf besproken.

6.3 Interactie

In deze paragraaf wordt de manier behandeld waarop de gebruiker interageert met de applicatie en de interieurobjecten.

6.3.1 Interactie met het FlowMenu

De applicatie beschikt niet over een traditionele menubalk, maar maakt gebruikt van een FlowMenu (zie paragraaf 4.2.1.3). Net zoals bij de uitwerking van het FlowMenu door Guimbretière in [26], is er ook bij deze implementatie voor gekozen om in één menu acht verschillende octanten te tonen voor de verschillende beschikbare menukeuzes (zie Figuur 6.3).



Figuur 6.3 FlowMenu

Zoals reeds in een vorige paragraaf aangehaald, is het FlowMenu de reden waarom de ontwerpruimte van de applicatie (het 'canvas') met een brede rand is omgeven. Aan de hand van Figuur 6.3 is goed uit te leggen waarom precies dit het geval is. Aangezien het FlowMenu getoond moet worden op de plaats waar de gebruiker het heeft opgeroepen (het zogenaamde *point of focus,* zie [26]) en cirkelvormig is, moet er rond elke mogelijk *point of focus* (dus elk punt van de ontwerpruimte van de applicatie) voldoende ruimte zijn, wat zich vertaalt in een brede rand aan de vier zijden van de ontwerpruimte die zo breed is als de straal van de cirkel van het FlowMenu. Indien deze rand niet voorzien zou zijn, dan zou bijvoorbeeld in Figuur 6.3 het linker gedeelte van het FlowMenu niet binnen het applicatievenster vallen.

6.3.1.1 Het tonen van het FlowMenu

Het menu wordt, net als bij een *marking menu* van Kurtenbach in [20], niet onmiddellijk na het aanraken van het scherm getoond, maar na een korte *press-and-wait* periode. Net als bij een *marking menu* is ervoor gekozen om het menu pas te tonen na ongeveer een derde van een seconde. Visueel wordt de voortgang van deze interne klok aan de gebruiker teruggekoppeld middels een kleine visuele indicator naast de muispointer.

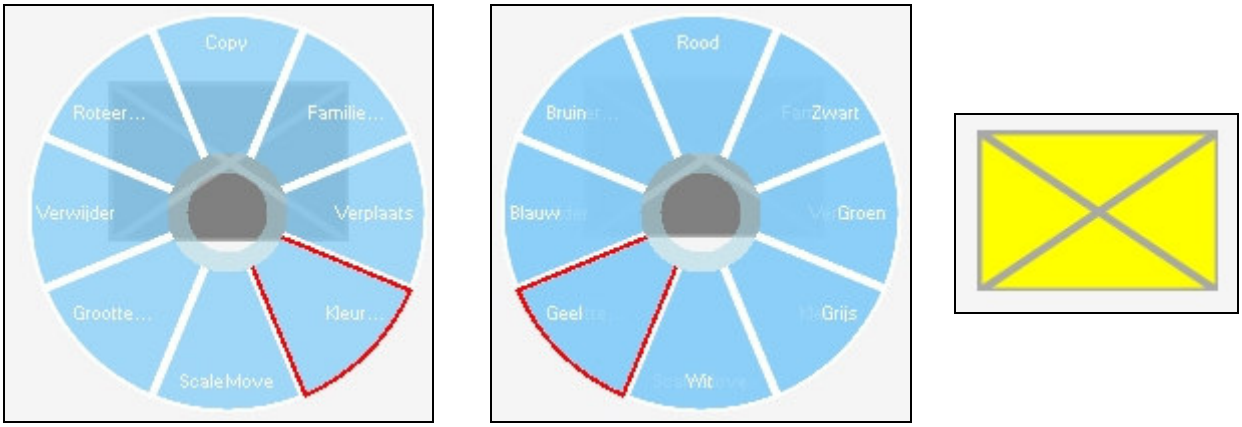
Wanneer de gebruiker tijdens de *press-and-wait* de vinger van het scherm opheft, of beweegt naar een andere plaats zonder de vinger van het scherm te halen, wordt het FlowMenu dus niet getoond. Dit mechanisme is echter een beetje aangepast moeten worden. Het bleek immers al snel tijdens het testen van de applicatie op het SmartBoard dat het niet makkelijk is om de vinger perfect stil op het bord te houden (met 'stil' wordt hier bedoeld dat de coördinaten hetzelfde blijven). Hierdoor werd de interne klok regelmatig gestopt zonder dat je als gebruiker die bedoeling had. De oplossing die hiervoor is gemaakt, is eenvoudig: pas wanneer de gebruiker de vinger over een bepaalde voldoende afstand heeft verplaatst, wordt de klop gestopt.

6.3.1.2 Kiezen in het FlowMenu

De gebruiker maakt in het FlowMenu als volgt zijn keuze: wanneer het FlowMenu getoond wordt, beweegt hij zijn vinger naar het menu-item van zijn keuze en vervolgens opnieuw naar het midden (het centrale rustgebied). Het octant dat de

gebruiker gekozen heeft, wordt visueel aangepast weergeven: het octant in kwestie is omrand met een andere kleur (zie Figuur 6.3).

Na het kiezen van een menu-item, kunnen er twee verschillende zaken gebeuren afhankelijk van het feit of het een menu-item betreft dat submenu-items bevat of niet. Indien dit wel het geval is, wordt het bijhorende submenu getoond. Dit is bijvoorbeeld te zien in Figuur 6.4 waar de gebruiker eerst kiest voor het menu-item 'Kleur' (waarbij een submenu hoort), en vervolgens het submenu getoond wordt, waarna de gebruiker kiest voor 'Geel'. Net als in [26] wordt door het toevoegen van drie puntjes achter de tekst van het menu-item aangegeven dat het betreffende item een sub-menu bevat.



Figuur 6.4 FlowMenu menuselectie

In tegenstelling tot het FlowMenu in [26], is er bij deze implementatie niet voor gekozen om de submenu-items boven de bijhorende menu-items te plaatsen (zie Figuur 4.4). Deze keuze is gemaakt omdat de ruimte tamelijk beperkt is: een enkel menu-item bestaat al vrij vlug uit tien letters of meer (bijvoorbeeld 'applicatie'), en het toevoegen van een submenu-item dat ook van dergelijke lengte zou zijn was niet mogelijk binnen de ruimtelijke grenzen van het FlowMenu. Vandaar dat er voor gekozen is om de teksten van het hoofdmenu een heel pak lichter weer te geven, en de tekst van het submenu-item gewoon op dezelfde plaats te zetten wanneer een submenu getoond moet worden.

6.3.1.3 Tekst in het FlowMenu

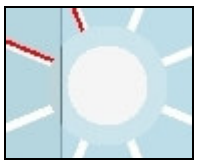
In paragraaf 4.2.1 is reeds aangehaald dat het cirkelvormige karakter van een *pie menu* tot problemen kan leiden wat betreft het plaatsen van tekst: hoe meer tekst in dergelijk menu weergegeven moet worden, hoe groter het menu moet zijn. En aangezien in de implementatie uitgegaan is van een FlowMenu van vaste grootte, vertaalt dit probleem zich in een beperking wat betreft de hoeveelheid tekst die weergegeven kan worden. Een variabele grootte zou immers leiden tot een hogere graad van complexiteit omwille van de eis dat op elk *point of focus* het volledige menu weergegeven moet kunnen worden.

Over dit laatste nog dit ter vergelijking: in [26], het FlowMenu van Guimbretière, is de langste tekst van de menu-items die in de voorbeelden zijn weergeven negen letters

lang (wat voor een doorsnee 'klassiek' lineair menu wel erg kort zou zijn). Een mogelijk antwoord op dit probleem (het is niet echt een oplossing, maar eerder een verlichting van het probleem), zou zijn om toe te staan dat de tekst van een menu-item de grenzen van het betreffende octant zou mogen overschrijden, wat dan weer wel -als nadeel- zou kunnen leiden tot het overlappen van teksten van de verschillende octanten. Dit zou een mogelijke uitbreiding zijn van het FlowMenu zoals het in deze implementatie is uitgewerkt.

6.3.1.4 Bepaling van keuzes in het FlowMenu

In Figuur 6.5, een uitsnede van het centrale gedeelte van Figuur 6.3, is de binnenste ring goed te zien. Deze ring bepaalt wanneer een menuoptie gekozen wordt. Wanneer een gebruiker de pointer beweegt naar een van de acht octanten en vervolgens terugbeweegt naar het midden van het FlowMenu, zal de keuze effectief gemaakt worden zodra de pointer zich in deze binnenste ring bevindt.

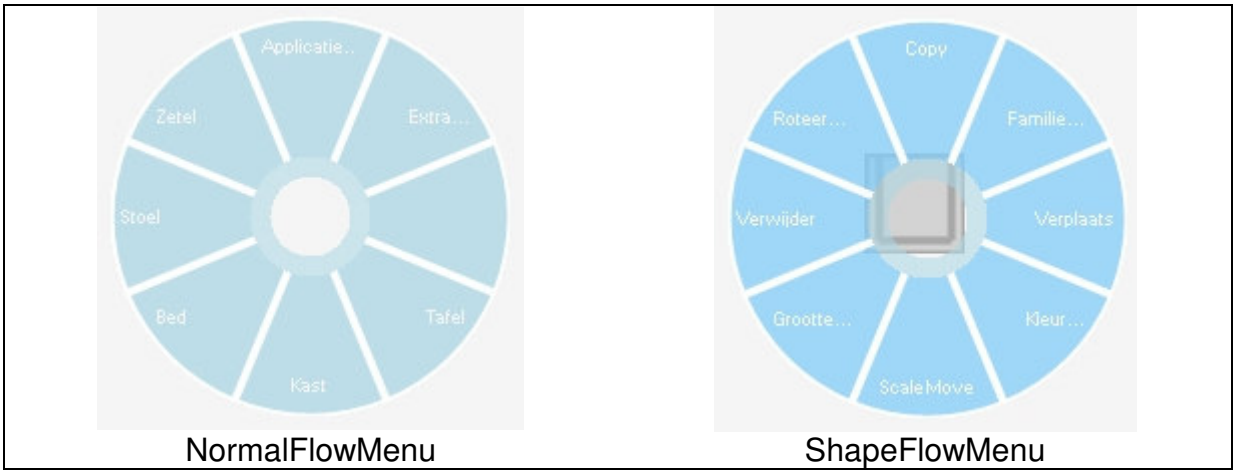


Figuur 6.5 Binnenste ring FlowMenu

Opgemerkt dient hier dat dit een vereenvoudiging is van het FlowMenu zoals Guimbretière het in [26] heeft uitgewerkt waarbij gebruik gemaakt wordt van twee cirkels om de grens aan te geven tussen het centrale rustgebied en een octant. De overgang van het centrale rustgebied naar een octant wordt bepaald aan de hand van een cirkel met een iets grotere straal dan de overgang van het octant naar het centrale rustgebied, zoals te zien is in Figuur 4.4 (a) en Figuur 4.4 (b). In deze implementatie van het FlowMenu wordt dus slechts één en dezelfde cirkel voor beide overgangen gebruikt: de buitenste rand van de ring in Figuur 6.5. Oorspronkelijk was het de bedoeling om het FlowMenu ook te voorzien van twee verschillende overgangscirkels, maar tijdens het werken met het FlowMenu zoals het is geïmplementeerd, zijn geen tekortkomingen naar boven gekomen van het gebruik van één cirkel die dergelijke uitbreiding zouden rechtvaardigen.

6.3.1.5 Verschillende types FlowMenu

Via het FlowMenu kan de gebruiker shapes toevoegen en manipuleren. De gebruiker roept het FlowMenu op door eenvoudigweg het scherm aan te raken. Afhankelijk van de plaats waar de gebruiker het FlowMenu oproept (op een shape of op een plaats waar geen interieurobject staat), wordt er een ander menu getoond. Op deze manier zijn er dus afhankelijk van de context twee verschillende menu's: het ene menu gaat uit van de veronderstelling dat de gebruiker iets met het aangeraakte object wil doen (het *ShapeFlowMenu*), terwijl het andere menu (het *NormalFlowMenu*) die veronderstelling niet maakt (zie Figuur 6.6). Voor de volledigheid dient hier vermeld dat onder sommige omstandigheden er nog een derde soort van menu getoond kan worden, namelijk het FamilyFlowMenu, wat verderop in deze paragraaf nog zal behandeld worden.



Figuur 6.6 De twee soorten FlowMenu's

6.3.1.5.1 NormalFlowMenu

Het NormalFlowMenu wordt dus zoals reeds vermeld, getoond wanneer de gebruiker het scherm aanraakt op een locatie waar zich geen shape bevindt. De applicatie maakt op basis hiervan de veronderstelling dat de gebruiker geen shape wil manipuleren, maar een shape wil toevoegen. Bijgevolg bevat het NormalFlowMenu bijna allemaal menu-items die onmiddellijk leiden tot het toevoegen van een shape aan het ontwerpvenster, en wel precies op de plaats waar het NormalFlowMenu werd opgeroepen. Op deze manier kan de gebruiker snel -en bovendien zonder zich in submenu's te moeten begeven- een kast, een tafel, een zetel, een bed of een stoel toevoegen aan de applicatie (zie Figuur 6.2).

Naast het toevoegen van objecten biedt het NormalFlowMenu ook de mogelijkheid om de applicatie te beëindigen, om opnieuw te beginnen, en om een vooraf gedefinieerd ontwerp in te laden. Deze laatste ontwerpen zijn vast in de applicatie verwerkt, en zijn niet door de gebruiker aan te passen. Ook het bewaren van ontwerpen is niet mogelijk. Het niet opnemen van het laden en bewaren van ontwerpen in de implementatie is gemotiveerd door het doel van de applicatie zoals aangehaald in paragraaf 6.2. Het aanbieden van deze functionaliteit zou de applicatie wel beter 'dagelijks' bruikbaar maken.

Tot slot biedt het NormalFlowMenu ook nog een submenu 'Extra's', waaronder een aantal interactietechnieken gegroepeerd zitten die in de volgende paragrafen nog aan bod komen.

6.3.1.5.2 ShapeFlowMenu

Het ShapeFlowMenu onderscheidt zich dus van het NormalFlowMenu doordat het steeds gelinkt is aan het interieurobject waarop de gebruiker het FlowMenu heeft opgeroepen. De applicatie gaat ervan uit dat de gebruiker iets met de betreffende shape wil doen, en toont dus het ShapeFlowMenu dat de relevante manipulatieopties bevat die de applicatie aanbiedt.

Aangezien het ShapeFlowMenu nogal wat submenu's bevat, wordt hierna de structuur van het menu weergegeven in een boomstructuur:

- Verwijder
- Verplaats
- Grootte...
  - Schalen
  - Uitrekken

  (Grootte-submenu)
- Kleur...
  - Grijs
  - Wit
  - Geel
  - Blauw
  - Bruin
  - Rood
  - Zwart
  - Groen

  (Kleur-submenu)
- Roteer...
  - 90° CW[1]
  - 90° CCW[2]
  - 180°
  - Vrij

  (Rotatie-submenu)
- Copy
- Familie...
  - Keuken
  - Living
  - Slaapkamer
  - Bureau
  - Geen

  (Familie-submenu)
- ScaleMove

Verwijderen en verplaatsen

Er is niet veel verbeelding nodig om zich iets van de meeste menu-opties te kunnen voorstellen: de opties 'Verwijder' en 'Verplaats' doen exact wat ze beloven, met hierbij nog de opmerking dat bij de keuze voor het verplaatsen van een interieurobject de gebruiker de shape kan verplaatsen zolang hij de vinger tegen het scherm houdt. Zodra de gebruiker de vinger van het scherm neemt, en als het ware de shape 'loslaat', beweegt de shape niet verder.

Kleur wijzigen

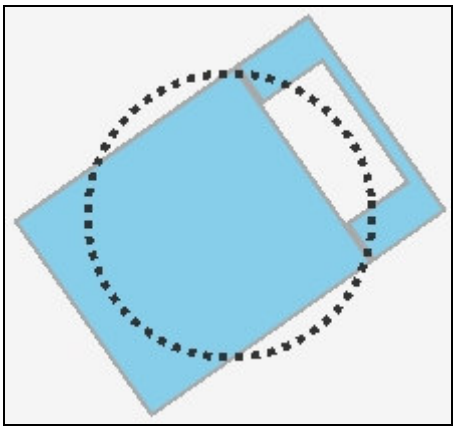
Het wijzigen van de kleur is reeds getoond in Figuur 6.4. Het aantal kleuren waaruit de gebruiker kan kiezen is beperkt: acht vooraf ingestelde kleuren kunnen gekozen worden.

Rotatie

Wanneer de gebruiker de shape wil roteren, kan hij dat op verschillende manieren doen: via een vaste hoek (90 graden in of tegen wijzerzin en 180 graden) of vrij roteren. Bij dit laatste verschijnt er een 'hulpcirkel': door de vinger langs de rand van

[1] CW: *Clockwise*, in wijzerzin
[2] CCW: *Counterclockwise*, in tegenwijzerzin

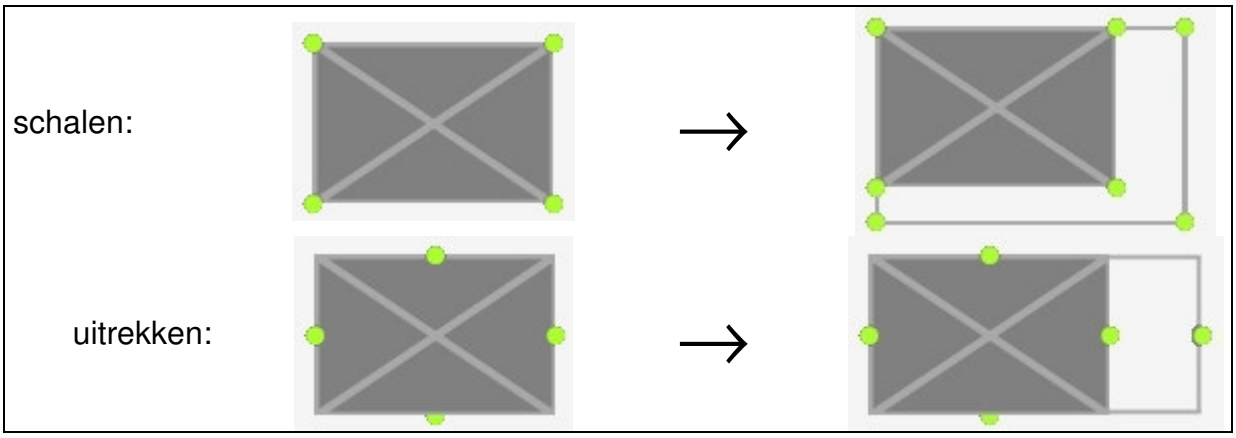
de cirkel te bewegen geeft de gebruiker aan hoe ver hij de shape wil draaien (zie Figuur 6.7). Overigens hoeft de gebruiker niet echt langs deze hulpcirkel te bewegen: de applicatie gebruikt enkel het middelpunt van de shape en de positie van de vinger van de gebruiker om de hoek te berekenen. Zodra de gebruiker de vinger van het scherm opheft, houdt de rotatie op.



Figuur 6.7 Vrije rotatie

Grootte wijzigen

De opties ‘Schalen’ en ‘Uitrekken’ van het menu ‘Grootte…’ bieden zoals verwacht de gebruiker de mogelijkheid om de grootte van het interieurobject te wijzigen. Bij het kiezen van één van beide mogelijkheden, verschijnen hulpmarkeringen op de shape (zie Figuur 6.8). Deze hulpmarkeringen blijven aanwezig tot de gebruiker de vinger opheft van het scherm. Door een van die markeringen te verslepen, past men de grootte van het object aan.



Figuur 6.8 De grootte van een shape wijzigen

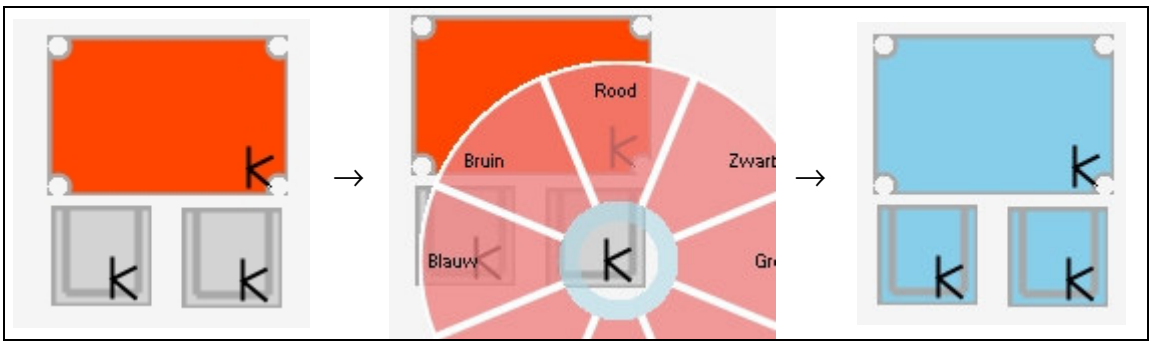
Net als bij de vrije rotatie, wordt de actie (in dit geval het aanpassen van de grootte) uitgevoerd zolang de gebruiker de vinger op het scherm houdt. Wanneer de vinger van het scherm wordt gehaald, verdwijnen de markeringen en is het aanpassen van de grootte niet meer mogelijk. Voor deze opzet is gekozen omdat op deze manier de shapes zonder markeringen of hulpcirkels te bekijken zijn nadat de actie is afgelopen (de markeringen en hulpcirkels horen immers niet bij het interieur) en omdat dit een vlotte manipulatie niet in de weg staat. De informele gebruikerstest wees overigens in de richting dat dit -ook al is er een zeer korte gewenningsperiode voor nodig- geen problemen geeft (zie paragraaf 7).

Kopiëren

In een vroege versie van de applicatie bestond er nog niet de mogelijkheid om een shape te kopiëren. Wanneer een gebruiker bijvoorbeeld een lange tafel met daarrond acht stoelen wou plaatsen, moest hij telkens de volgende actie herhalen: in het menu kiezen om een stoel toe te voegen, in het menu kiezen om de stoel te verplaatsen, de stoel op de juiste plaats neerzetten, de stoel roteren indien nodig, en tot slot de stoel een gepaste kleur geven indien nodig. Dit bleek al snel behoorlijk omslachtig en hinderlijk voor een vlotte interactie. Daarom is de optie toegevoegd om een shape te kopiëren: bij het kiezen van deze optie wordt onmiddellijk een nieuwe shape toegevoegd (als kopie van de shape waarop het ShapeFlowMenu opgeroepen werd) die de gebruiker vervolgens op de juiste plaats kan neerzetten. Deze nieuwe shape neemt de kleur, grootte en rotatie en familie (zie volgende alinea) van het bronobject over en versnelt het beschreven proces aanzienlijk. Door het opheffen van de vinger van het scherm, krijgt de shape tenslotte zijn juiste plaats.

Familie

De optie ‘Familie…’ laat de gebruiker toe om een shape aan een bepaalde ‘familie’ toe te wijzen. Met familie wordt hier een soort van klasse bedoeld waaronder objecten in een interieur gegroepeerd kunnen worden zoals living, keuken, slaapkamer en bureau. Dergelijke familie of klasse bestaan min of meer uit een vaste samenstelling van objecten. Een slaapkamer bijvoorbeeld bestaat veelal uit een bed en een of meer kasten. In de applicatie is de optie ingebouwd om objecten toe te kunnen wijzen aan een bepaalde familie, en om operaties uit te voeren op een familie i.p.v. een individueel object. Op deze manier kan bijvoorbeeld de living door middel van één enkele handeling een andere kleur krijgen (zie Figuur 6.9).



Figuur 6.9 Familie-operaties

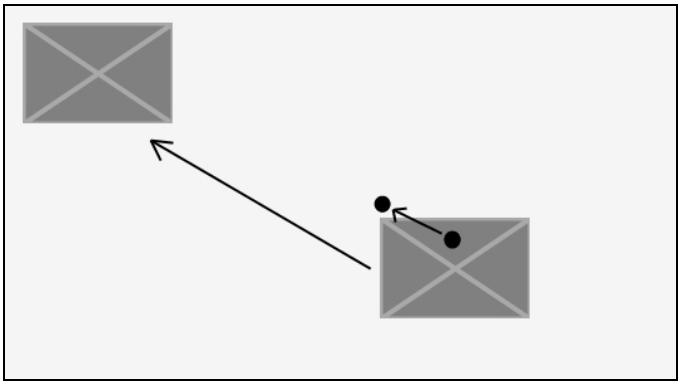
Zoals te zien in Figuur 6.9 wordt de familie van een shape aangegeven door een familiemarker: een letter in de rechterbenedenhoek (de eerste letter van de naam van de familie). Indien een object tot geen enkele familie behoort, wordt er geen familiemarker getoond. Indien de gebruiker de vinger op de familiemarker drukt, wordt het FamilyFlowMenu getoond (zie in het midden van Figuur 6.9). De keuze die in dit menu gemaakt wordt, wordt toegepast op alle leden van de familie.

Deze implementatie wil vooral het principe demonstreren, vandaar dat niet alle mogelijkheden benut zijn die het werken met dergelijke klassen kunnen bieden: het aantal families en de namen ervan zijn vast (bureau, living, slaapkamer en keuken) en de operaties die erop uitgevoerd kunnen worden beperken zich in deze implementatie tot het wijzigen van de kleur. Deze techniek verdient in een eventuele uitbreiding dan ook zeker de nodige verder aandacht.

ScaleMove

De laatste operatie die geboden wordt door het ShapeFlowMenu is de ScaleMove. Deze techniek is ingegeven door een specifieke eigenschap van digitale whiteboards, namelijk de grootte van een whiteboard. Het is niet voor iedereen op elk whiteboard vanzelfsprekend om elk plaatsje op het whiteboard te kunnen bereiken. In paragraaf 5.2 is dit probleem al aangehaald en zijn tevens een aantal oplossingen vermeld die hiervoor in de literatuur terug te vinden zijn.

Met ScaleMove is het mogelijk om objecten snel over een grote afstand te verplaatsen: de beweging van de pointer (met andere woorden de vinger op het bord) wordt versterkt met een bepaalde factor (zie Figuur 6.10 waarin het zwarte bolletje de vinger van de gebruiker voorstelt) zodat een verplaatsing van de vinger van 10 pixels op de X-as vertaald wordt in een verplaatsing van het object van bijvoorbeeld 50 pixels.



Figuur 6.10 ScaleMove

Het belangrijkste nadeel van deze methode is het gebrek aan nauwkeurigheid. Bij mijn implementatie wordt de verplaatsing van de pointer eenvoudigweg vermenigvuldigd met een vaste waarde (vandaar de naam). Dit betekent dat het te verplaatsen object zich met deze techniek steeds in stappen zal verplaatsen die nooit kleiner kunnen zijn dan deze vaste waarde. De gebruikte translatie is dus eenvoudig maar 'grof'. Meerdere alternatieven zijn mogelijk, maar het zou bijvoorbeeld al beter kunnen zijn indien gebruik gemaakt zou worden van een formule die rekening houdt met de factor tijd: wanneer de snelheid van de verplaatsing laag is, zou een één-op-één mapping van beide bewegingen (van shape en pointer) voor de nodige nauwkeurigheid kunnen zorgen.

Wanneer nauwkeurigheid een belangrijke eis is, dan schiet deze techniek dus een beetje tekort. Van de andere kant is het wel zo dat deze techniek niet noodzakelijk gebruikt moet worden om objecten naar locaties te verplaatsen waar de gebruiker niet of nauwelijks bij kan. De techniek kan immers ook gebruikt worden om een shape snel naar de andere kant van het bord te bewegen, waarna de gebruiker dan andere technieken kan kiezen voor het fijnere afstelwerk. Bovendien wens ik ook nog op te merken dat locaties waar de gebruiker fysiek niet makkelijk bij kan, locaties kunnen zijn waar de gebruiker geen goed zicht op heeft (de kijkhoek naar de locatie zou allesbehalve loodrecht kunnen zijn, wat de waarneming sterk bemoeilijkt). Bijgevolg zal de gebruiker niet veel gebaat zijn met een zeer nauwkeurige verplaatsing van een object dat hij toch niet zo goed kan zien.

Tot slot bestaat er nog een ander voordeel van deze techniek: indien meerdere personen aan hetzelfde whiteboard werken, dan kan deze techniek het in paragraaf 5.4 genoemde probleem verhelpen waar gebruikers in elkaars weg staan. Door het loskoppelen van de één-op-één relatie tussen de beweging van het te verplaatsen object en de beweging van de vinger, kan een gebruiker immers een object verplaatsen in een gebied waar hij fysiek niet bij kan (of niet bij wil omdat hij dan een andere gebruiker zou storen).

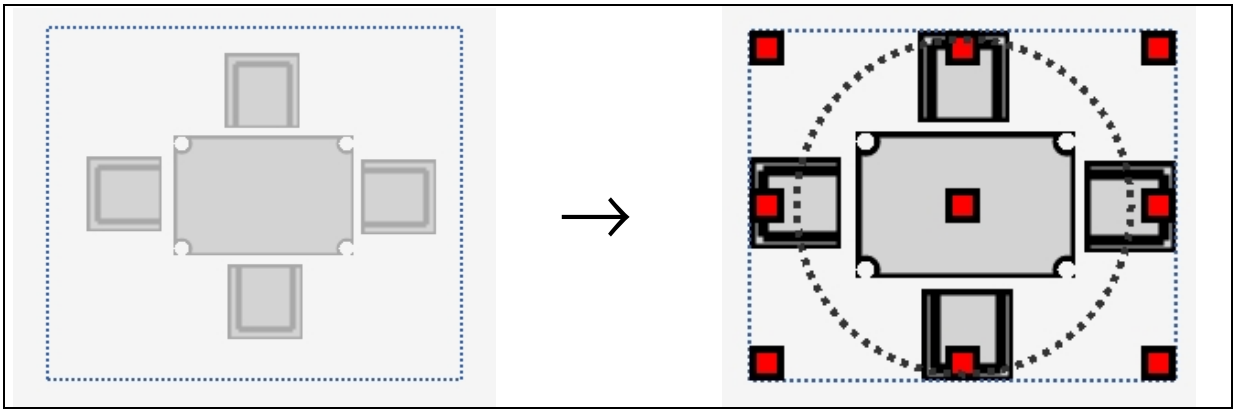
6.3.2 Andere interactiemanieren

In de vorige paragrafen lag de focus voornamelijk op interactie op een enkel object. In de praktijk zal een gebruiker echter vaak een operatie willen uitvoeren op meerdere objecten tegelijk, zoals bijvoorbeeld een groep shapes verplaatsen. In de volgende paragrafen worden enkele interactietechnieken voorgesteld die voor dergelijke gebruikersintenties een antwoord pogen te bieden, naast nog enkele andere interactiemanieren. Voor de volledigheid dient vermeld te worden dat het wijzigen van de kleur van een familie objecten reeds aan bod kwam in 6.3.1.5.2 waar het samen behandeld is met het instellen van de familie van een enkel object.

6.3.2.1 SelectionBox

Een eerste manier die het werken met meerdere shapes tegelijk moet vergemakkelijken is de SelectionBox. Een gebruiker hoeft geen expliciete keuzes te maken om een SelectionBox te kunnen tekenen: wanneer hij de vinger op het scherm drukt en vervolgens zijn vinger beweegt (terwijl hij de vinger op het scherm houdt), zal de applicatie een rechthoek tekenen tussen het punt waar de gebruiker de eerste keer het scherm aanraakte en het punt waar de vinger van de gebruiker zich bevindt. Deze rechthoek wordt uiteraard steeds aangepast wanneer de gebruiker zijn vinger op het scherm beweegt.

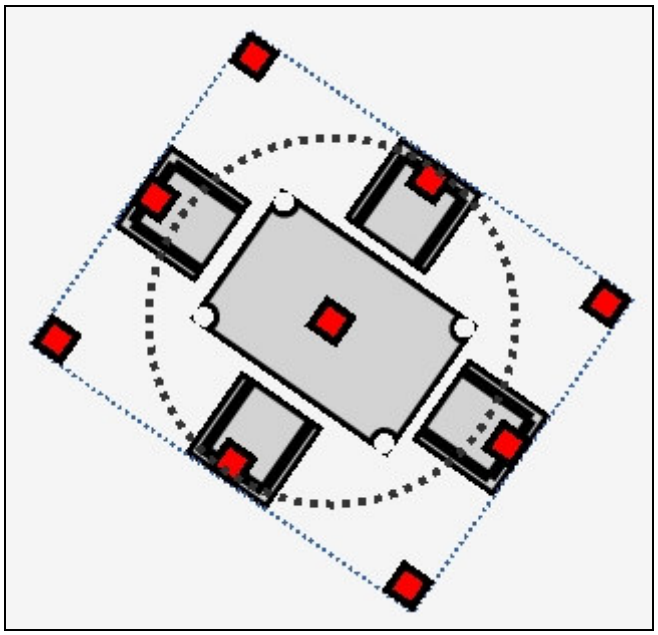
Wanneer de gebruiker de vinger van het scherm opheft, gaat de applicatie kijken naar welke shapes volledig omsloten worden door de SelectionBox en wordt de grootte van de SelectionBox zodanig verkleind dat het de minimale rechthoek -de minimale bounding box- vormt die de shapes kan bevatten (zie Figuur 6.11). Indien zich in de SelectionBox geen enkele shape bevindt, wordt deze verwijderd. Op de SelectionBox worden vervolgens een aantal markeringen aangebracht. Met deze markeringen kan de gebruiker alle shapes die ingesloten zitten in de SelectionBox tegelijk manipuleren. De shapes die zich binnen de SelectionBox bevinden worden visueel aangepast weergeven: de applicatie tekent deze met dikkere lijnen. Verderop in dit hoofdstuk (paragraaf 6.3.2.6) wordt een andere techniek behandeld waarbij deze geselecteerde shapes een rol spelen.



Figuur 6.11 SelectionBox: minimale bounding box

Wanneer de SelectionBox zijn grootte heeft aangepast om alle shapes te omvatten (en de gebruiker de vinger dus van het scherm heeft genomen) blijft de SelectionBox zichtbaar (wat niet gebeurt met de markeringen wanneer een individuele shape vergroot of verkleind wordt, zie paragraaf 6.3.1.5.2). Pas nadat de gebruiker een manipulatie van een van de markeringen heeft uitgevoerd, verdwijnt de SelectionBox, inclusief markeringen.

In eerste instantie was enkel voorzien om de grootte te kunnen wijzigen: hiertoe werden in het midden van elke zijde, en in elke hoek van de SelectionBox markeringen geplaatst. Al snel bleek echter dat dit niet meteen de meest wenselijke manipulatiemogelijkheden waren: het verplaatsen en roteren van meerdere shapes tegelijk zijn operaties die vaker nodig zijn. Zo kan een gebruiker bijvoorbeeld gemakkelijk een tafel met daarrond vier stoelen negentig graden draaien of naar een andere locatie verplaatsen. Hiervoor is -net als bij de rotatie van een individuele shape- een rotatiehulpcirkel toegevoegd, en een markering in het midden van de SelectionBox zoals te zien is in Figuur 6.12.



Figuur 6.12 SelectionBox: rotatie

Wanneer de gebruiker de vinger neerlegt op een van de markeringen van de SelectionBox en vervolgens de vinger beweegt, zullen de shapes de gevraagde operatie uitvoeren, en ook de SelectionBox zelf volgt: wanneer de beweging bijvoorbeeld op de rotatiecirkel gebeurt, zullen zowel de shapes als de SelectionBox geroteerd worden. Als de gebruiker daarna de vinger van het scherm haalt, verdwijnt de SelectionBox en behouden de shapes uiteraard hun zopas gewijzigde locatie, grootte en rotatie.

6.3.2.2 Bot

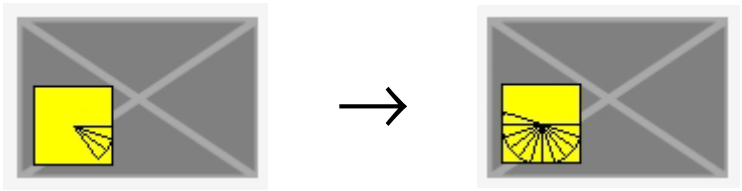
In paragraaf 6.3.1 kwam de ScaleMove al aan bod als een antwoord op het probleem waar de grootte van interactieve whiteboards toe kan leiden, zoals uiteengezet in paragraaf 5.2. De Bot is een andere techniek die hetzelfde probleem tracht op te vangen, maar in tegenstelling tot ScaleMove gaat het hierbij om het *selecteren* van objecten die zich op moeilijk of niet bereikbare plaatsen bevinden in plaats van het *verplaatsen* van objecten naar locaties die moeilijk of niet bereikbaar zijn.

Wanneer de gebruiker in het FlowMenu (meerbepaald het NormalFlowMenu) kiest voor de Bot (in het 'Extra's' submenu), dan verschijnt er een klein vierkant blokje (zie Figuur 6.13 dat een Bot toont tijdens het selectieproces). Dit blokje kan de gebruiker bewegen over het scherm door middel van een beweging van de vinger: de bot volgt de bewegingen van de vinger net als een muiscursor. Echter, de verplaatsingen van de Bot zijn groter dan die van de vinger. De verplaatsing van de vinger wordt met een bepaalde factor vermenigvuldigd om zo de verplaatsing van de bot te bepalen. Op deze manier kan de gebruiker plaatsen op het whiteboard bereiken zonder zich effectief op die plaatsen te moeten begeven (wat zoals reeds aangehaald het geval kan zijn bij grote borden). De Bot fungeert als een verlengstuk van de vinger van de gebruiker en het is de gebruiker die er de controle over uitoefent, vandaar de gekozen naam Bot wat een verkorting is voor robot.

Het implementatietechnisch bepalen van het object dat de gebruiker via de Bot wil selecteren ligt niet voor de hand. Voor een normale selectie (zonder Bot) volstaat het om eenvoudigweg de vinger op het bord te plaatsen op de plaats waar een shape staat (op de shape zelf dus): implementatie-technisch is op basis van de *mouse click* en de locatie op het bord van die *mouse click* vervolgens eenvoudig om te bepalen wanneer de gebruiker een object op de normale manier wil selecteren (voor een SmartBoard is het contact van de vinger met het bord hetzelfde als een ingedrukte muisknop).

Voor de Bot moest echter een andere methode gebruikt worden voor het bepalen van de selectie want de gebruiker heeft de vinger al op het bord voor het verplaatsen van de Bot. Een mogelijke oplossing zou zijn om niet het neerzetten (*mouse down)* maar het opheffen (*mouse up)* van de vinger te gebruiken. Maar het opheffen van de vinger wordt -naar analogie met andere technieken, zoals het verplaatsen van een object- consistent doorheen de applicatie gekoppeld aan het beëindigen van een operatie, in dit geval het verwijderen van de Bot. Om deze reden is voor het implementeren van de methode die bepaalt wanneer een object via de Bot geselecteerd is, gekozen voor een *timer*. Zodra de Bot zich over een shape bevindt,

start de Bot een interne klok die blijft lopen zolang de Bot dezelfde shape onder zich heeft. Tijdens het lopen van de klok wordt de gebruiker van deze voortgang op de hoogte gehouden door middel van een cirkel die zich met lijnen vult zoals te zien is in Figuur 6.13. Indien de gebruiker tijdens het verplaatsen van de Bot de vinger van het scherm haalt, verdwijnt de Bot.

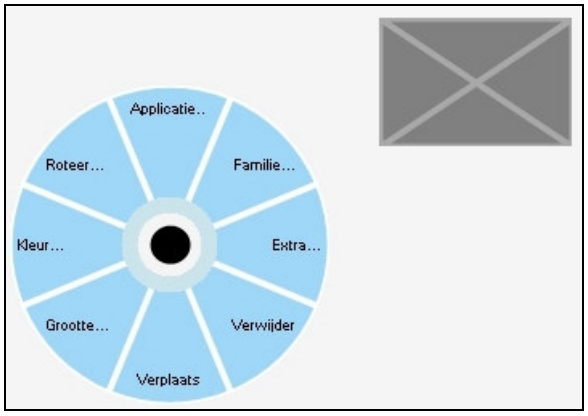


Figuur 6.13 Bot: selectie

Wanneer de interne timer van de Bot afloopt, veronderstelt de applicatie dat de gebruiker de betreffende shape wil selecteren, verdwijnt vervolgens de Bot en toont de applicatie het ShapeFlowMenu. Dit FlowMenu wordt niet getoond op de locatie van de Bot, maar op de plaats waar de vinger van de gebruiker zich bevindt, zoals te zien is in Figuur 6.14 (het zwarte bolletje -dat achteraf aan de afbeelding is toegevoegd en in de applicatie niet zichtbaar is- duidt hierbij de plaats aan waar de vinger zich bevindt). Hierdoor heeft de gebruiker snel toegang tot het menu doordat het menu vlakbij zijn hand wordt getoond.

De objectmanipulaties die via het ShapeFlowMenu mogelijk zijn, werken uiteraard ook indien het menu via de Bot is opgeroepen, maar er zijn een aantal verschillen. Zo wordt de rotatiehulpcirkel niet getoond rond de shape zelf (zoals wel het geval indien het menu met de vinger op de shape wordt opgeroepen), maar op de plaats waar het menu werd getoond. De rotatiecirkel is immers niet zo bruikbaar indien deze zich op een locatie zou bevinden waar de gebruiker niet goed bij kan (= de locatie van de shape). Ook het verplaatsen van de shape na selectie met de Bot verdient een toelichting. Kiest de gebruiker voor ‘Verplaats’ dan verplaatst de shape zich onmiddellijk naar de hand van de gebruiker waarna deze de shape verder kan verplaatsen. Kiest de gebruiker voor ScaleMove dan blijft de shape staan waar hij stond, maar vertaalt elke verdere beweging van de vinger zich in een (versterkte) verplaatsing van de shape in kwestie (zoals uiteengezet in paragraaf 6.3.1.5.2).

Het wijzigen van de grootte na selectie via de Bot is in de huidige implementatie niet aangepast aan het gegeven -dat mag toch verondersteld worden op basis van het gebruik van de Bot- dat de geselecteerde shape zich niet in de nabijheid van de gebruiker bevindt. Na het kiezen voor het uitrekken of schalen van het object, zullen dus de markeringen getoond worden, net alsof het object rechtstreeks (niet via Bot) geselecteerd is, en kan de gebruiker deze markeringen verplaatsen (en op deze wijze dus de grootte van de shape veranderen). Dat deze markeringen niet in het bereik van de gebruiker liggen, heeft in de huidige implementatie geen passend antwoord gekregen. In een verdere uitwerking zou gekeken kunnen worden naar hoe de Bot ook hier een rol zou kunnen spelen als ‘verlengstuk’ van de vinger van de gebruiker.



Figuur 6.14 FlowMenu na selectie via Bot

Deze techniek vertoont flink wat overeenkomsten met de ScaleMove. Het nadeel van de ScaleMove geldt ook voor de Bot: de nauwkeurigheid is beperkt. De Bot kan slechts bewogen worden in stappen die gelijk zijn aan de versterkingsfactor. Doch aangezien het hier enkel gaat om selectie (en niet manipulatie), weegt dit nadeel niet zo zwaar. De Bot start zijn interne klok (om het selectieproces te starten) wanneer zijn middelpunt zich boven een shape bevindt, dus enkel in het -in de praktijk onwaarschijnlijke- geval dat de gebruiker een shape wil selecteren die kleiner is dan de versterkingsfactor, zorgt dit voor het probleem dat een shape niet geselecteerd kan worden (indien bovendien geen enkele van de tussenstappen in de beweging van de Bot zich op de shape zouden bevinden). Ook het vermelde voordeel van ScaleMove, namelijk het vermijden dat meerdere simultane gebruikers elkaar fysiek hoeven te storen, geldt voor de Bot.

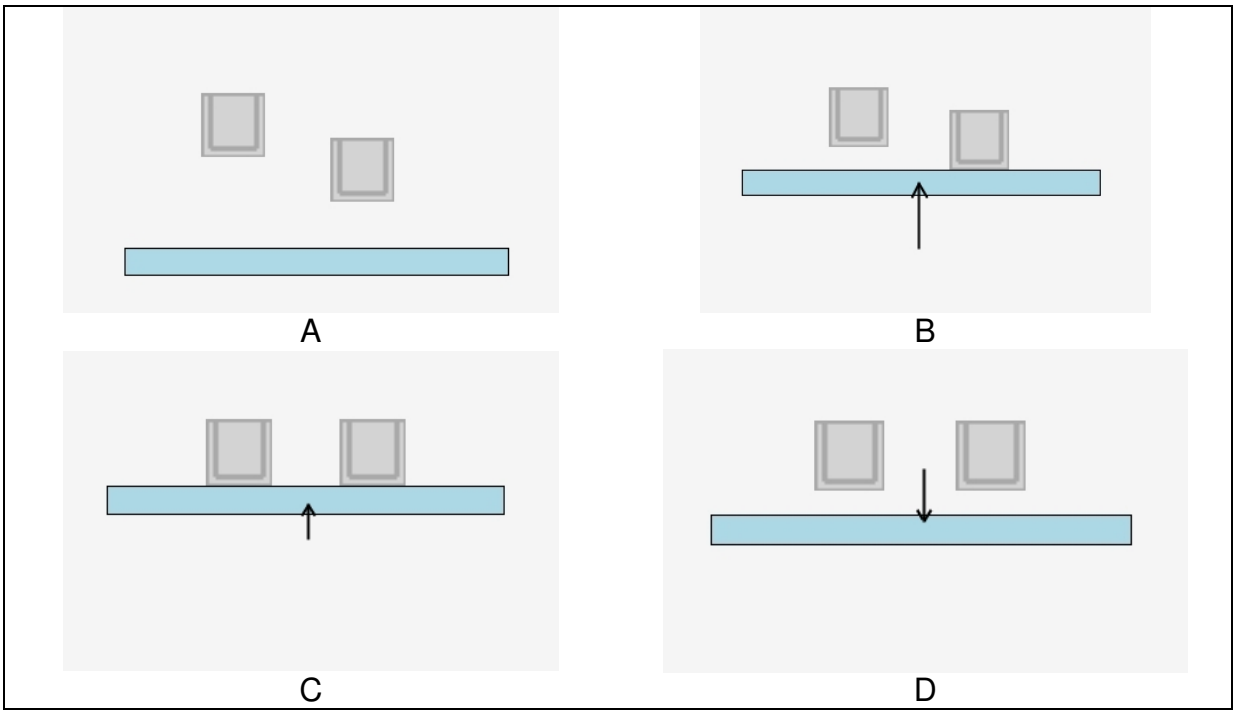
Tijdens het werken met de applicatie bleek al snel dat het precies de combinatie is van ScaleMove en de Bot die tot nuttige interacties kan leiden. Indien de gebruiker een object dat veraf staat naar een positie vlak voor hem op het bord wil verplaatsen, dan kan hij via de Bot de shape selecteren, en vervolgens via ScaleMove de shape naar zich toehalen.

6.3.2.3 Sweeper

De volgende interactietechniek die in deze paragraaf behandeld wordt, is terug te vinden in het submenu ‘Extra’s’ van het NormalFlowMenu en is ‘Sweeper’ genoemd. Deze techniek maakt het mogelijk om (een of meer) objecten te verplaatsen op een bijzondere manier. Met de ‘Sweeper’ kan de gebruiker een of meerdere shapes opzij ‘vegen’, en hij dankt dan ook zijn naam aan de gelijkenis met een borstel. Op het scherm wordt de Sweeper eenvoudig weergegeven: het is een rechthoekige vorm en heeft een lichtblauwe kleur. Wanneer de gebruiker de Sweeper beweegt en de Sweeper tijdens deze beweging in contact komt met een of meerdere shapes, dan zullen deze shapes in kwestie de beweging van de Sweeper volgen: de shapes die de Sweeper raakt, worden als het ware opzij geveegd door de Sweeper zoals te zien is in Figuur 6.15.

In Figuur 6.15 is een illustratie te zien van het gebruik van de Sweeper (de beweging van de gebruiker is met een zwarte pijl aangegeven). In (A) bevindt de Sweeper zich onder twee shapes. In (B) heeft de gebruiker de Sweeper naar boven bewogen en

heeft hierbij al een shape geveegd, namelijk de rechtse stoel. In (C) heeft de gebruiker de Sweeper nog verder naar boven geveegd: de twee objecten worden nu omhoog geduwd ten gevolge van het bewegen van de Sweeper. In (D) heeft de gebruiker de Sweeper naar onder bewogen. Omdat door deze beweging geen shapes geraakt worden, blijven de twee shapes op hun plaats.



Figuur 6.15 Sweeper: objecten vegen

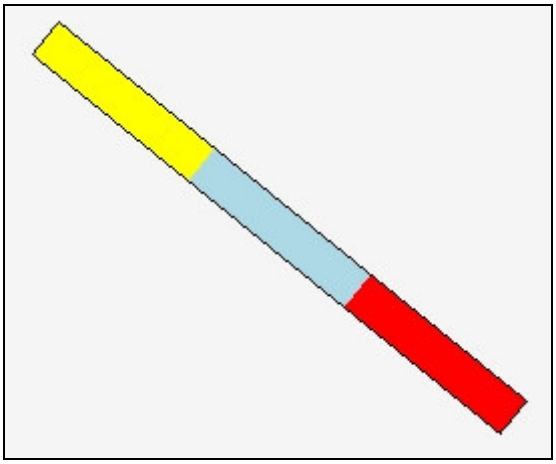
Na het kiezen van de Sweeper in het NormalFlowMenu, verschijnt de Sweeper op de plaats van de vinger en volgt de Sweeper de locatie van de vinger zolang de gebruiker de vinger op het scherm houdt. Wanneer de gebruiker de vinger van het scherm opheft, verdwijnt de Sweeper niet. Voor het verwijderen van de Sweeper dient gewoon opnieuw dezelfde menukeuze gemaakt te worden. Hiervoor is gekozen omdat ik de gebruiker de mogelijkheid wou bieden om de Sweeper te roteren en om de breedte ervan aan te passen.

Het is niet onmogelijk om toch deze functionaliteit aan te bieden en wél te werken met een techniek die consistent is met mijn betrachting -zoals in een vorige paragraaf reeds aangehaald- om het opheffen van de vinger consequent te koppelen aan het einde van een operatie. Een techniek die het roteren en het wijzigen van de grootte mogelijk maakt zonder dat de gebruiker daarvoor ooit zijn vinger van het bord hoeft te halen, heb ik zelfs uitgeprobeerd, maar bleek al snel zeer lastig voor de gebruiker. De gebruiker moet dan zowel de shape breder en smaller kunnen maken als kunnen roteren, bovenop de belangrijke eis dat hij de Sweeper moet kunnen verplaatsen. Dit alles bleek te omslachtig te werken en bovendien moeilijk te implementeren.

Om de hierboven aangehaalde reden, is ervoor gekozen om de Sweeper twee verschillende werkingsmodi te geven: een verplaats- en een manipulatiemodus. De

verplaatsmodus is reeds weergeven in Figuur 6.15: door de vinger op de Sweeper te plaatsen (en op het scherm te houden), kan de gebruiker de Sweeper verplaatsen, waarbij de Sweeper eventueel shapes opzij veegt. De manipulatiemodus is bedoeld voor het wijzigen van de rotatie en de breedte van de Sweeper. Het schakelen tussen beide modi gebeurt door middel van twee korte tikjes (*double click)* op de Sweeper.

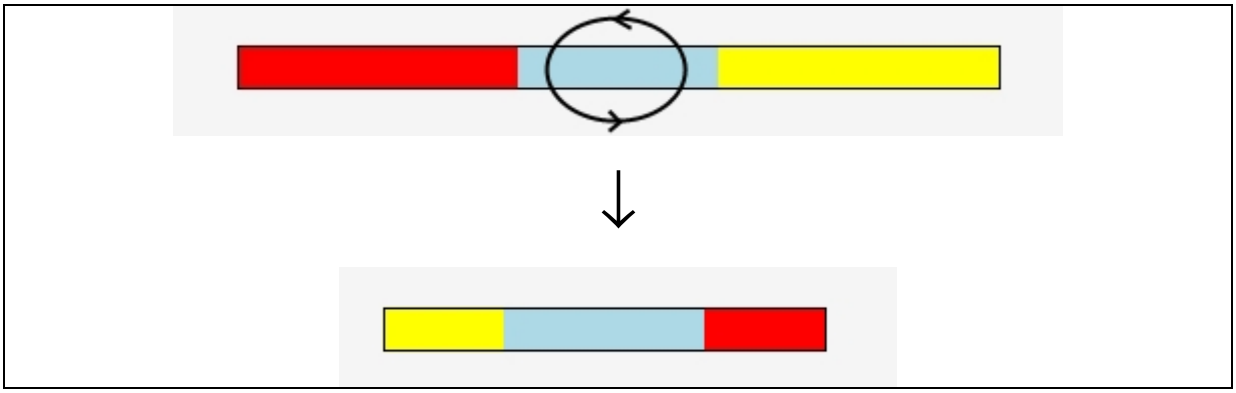
Wanneer de Sweeper zich in de manipulatiemodus bevindt, zal deze knipperen: de kleuren aan de zijkant Sweeper wisselen voortdurend tussen rood, geel, en lichtblauw (de normale kleur van de Sweeper). Op deze manier weet de gebruiker perfect de actuele modus van de Sweeper. In de manipulatiemodus kan de gebruiker de plaats van de Sweeper niet veranderen, enkel rotatie en breedte kunnen gewijzigd worden. De Sweeper is in de manipulatiemodus verdeeld in drie gedeeltes: de twee zijkanten en het midden (zie Figuur 6.16).



Figuur 6.16 Sweeper: manipulatiemodus

Deze drie gedeeltes spelen elk hun rol bij de manipulatie van de Sweeper. De zijkanten worden gebruikt om de rotatie te wijzigen. Wanneer de gebruiker in dit gedeelte de vinger plaatst en vervolgens beweegt, zal de Sweeper roteren rond zijn middelpunt en hierbij de beweging van de vinger volgen. Zo zou bijvoorbeeld Figuur 6.16 het resultaat kunnen zijn een gebruiker die zijn vinger in het rechtergedeelte van een (op dat moment) horizontale Sweeper heeft geplaatst en vervolgens naar beneden heeft bewogen.

Het midden van de Sweeper verzorgt het wijzigen van de breedte: de gebruiker plaatst hiertoe de vinger op het middelste gedeelte van de Sweeper, en maakt vervolgens cirkelvormige bewegingen met de vinger (het maakt hierbij niet uit of de vinger tijdens het maken van deze cirkels de Sweeper verlaat). Een cirkel getekend in wijzerzin leidt tot een toename van de breedte van de Sweeper, terwijl een cirkel in tegenwijzerzin de breedte vermindert. Figuur 6.17 illustreert het minder breed maken van de Sweeper.



Figuur 6.17 Sweeper: breedte wijzigen

De Sweeper is nuttig in situaties waarbij men op een snelle manier een bepaalde regio van het interieur wil vrijmaken, bijvoorbeeld wanneer men een aantal objecten opzij wil schuiven om er andere in de plaats te zetten. Maar ook voor het aligneren van objecten kan de Sweeper gebruikt worden. De objecten 'plakken' immers aan de Sweeper en volgen de beweging van de Sweeper zolang die beweging zorgt voor een contact tussen object en Sweeper. Deze aligneringsmogelijkheid tezamen met de beperking dat objecten de Sweeper niet langer volgden wanneer die van de objecten weg wordt bewogen, heeft me geïnspireerd om de techniek van de 'GlueStick' te ontwikkelen, die in de volgende paragraaf zal besproken worden.

Tot slot van deze bespreking over de Sweeper, wou ik graag nog een probleem aanhalen waar ik mee geconfronteerd werd bij het werken met de Sweeper. In een eerste implementatie werd de Sweeper op dezelfde locatie weergegeven als waar de betreffende menukeuze gemaakt werd met een vooraf ingestelde breedte. Hierdoor kon het gebeuren (bijvoorbeeld wanneer de Sweeper opgeroepen werd in een gebied waar reeds vele shapes staan) dat de Sweeper reeds onmiddellijk contact maakte met een object (overlapping) en dus meteen shapes opzij veegde bij het verplaatsen van de Sweeper.

Dit probleem is nog niet volledig opgelost: bij het creëren van de Sweeper gaat de applicatie nu enkel na of de vooraf ingestelde breedte zou leiden tot een overlapping met een shape en indien nodig wordt de breedte verkleind. Doch dit lost het probleem niet volledig op: een Sweeper heeft altijd een minimumbreedte nodig (de drie interactiegedeeltes -zijkanten en midden- moeten ten allen tijde gebruikt en dus getoond kunnen worden) dus de kans op overlap is nooit volledig weg te werken. Een mogelijke doorontwikkeling van de applicatie zou eventueel het risico op initiële overlapping kunnen verkleinen door ook de rotatie en de dikte van de Sweeper aan te passen naargelang de context van de shapes die zich in de nabijheid bevinden, of eventueel de eis opgeven dat de Sweeper na de menukeuze op de locatie van de vinger van de gebruiker moet getoond worden.

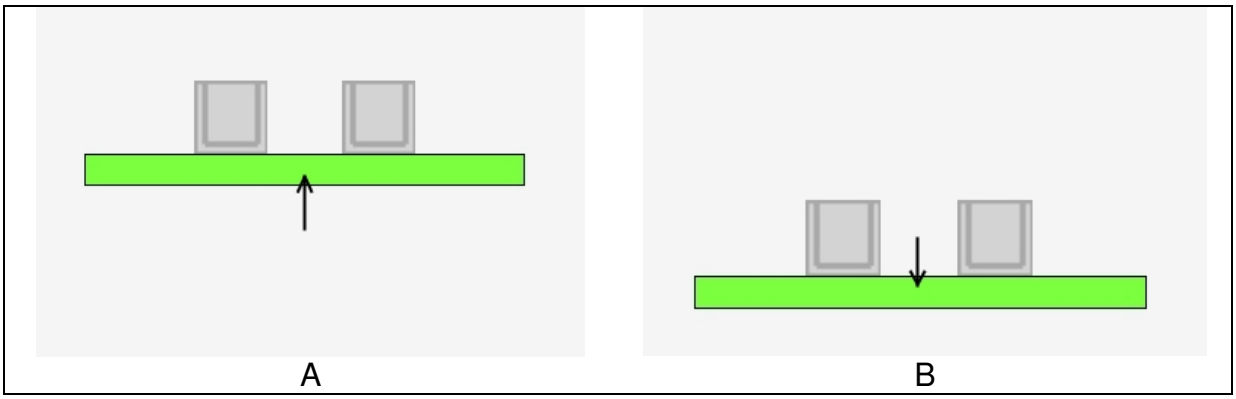
6.3.2.4 GlueStick

Zoals reeds aangehaald in de vorige paragraaf, is het idee voor de GlueStick ontsproten uit de vaststelling dat het aligneren van objecten door middel van de Sweeper (ook al is dit niet de core-functionaliteit van de Sweeper) het gebrek

vertoonde dat shapes niet meer mee bewogen met de Sweeper van zodra de Sweeper weg van de shapes bewoog.

Visueel is de GlueStick van de Sweeper te onderscheiden door zijn lichtgroene kleur, voor het overige is er geen enkel uiterlijk verschil. Ook de interactie met de GlueStick zelf verloopt net als bij de Sweeper: men kan de GlueStick verplaatsen, en de breedte en rotatie ervan wijzigen. Ook de plaats in het menu is dezelfde: submenu 'Extra's'. De genoemde voor- en nadelen gelden dus voor beide objecten, met als belangrijkste verschil dat de GlueStick shapes vasthoudt, ook wanneer de GlueStick weg van de objecten bewogen wordt.

In Figuur 6.18 is het verschil met de Sweeper duidelijk: terwijl in Figuur 6.15 de shapes blijven staan als de Sweeper naar beneden bewogen wordt (zie Figuur 6.15 D), is dat bij de GlueStick niet het geval: indien de shapes eenmaal contact hebben gemaakt met de GlueStick (Figuur 6.18 A), blijven ze aan de GlueStick kleven en zullen de GlueStick steeds volgen wanneer de gebruiker de GlueStick verplaatst, ook al is dit een richting waarbij de shapes niet in de weg staan (zie Figuur 6.18 B).



Figuur 6.18 GlueStick

Omdat de objecten aan de GlueStick blijven kleven, is het veel gemakkelijker om objecten te aligneren: de gebruiker hoeft enkel de shapes die hij op één lijn wil krijgen te 'vangen' met de GlueStick en op de juiste plaats te zetten. Toch bleek dat er nog een probleem de kop opstak: dit probleem treedt op wanneer een gebruiker een shape aan de GlueStick wil toevoegen dat zich niet op een makkelijk toegankelijke locatie bevindt. Een shape kan immers op een moeilijk toegankelijke plaats staan omdat het bijvoorbeeld omgeven is door andere objecten, waardoor het moeilijk of zelfs onmogelijk wordt om het gewenste object -en enkel dit- aan de GlueStick toe te voegen.

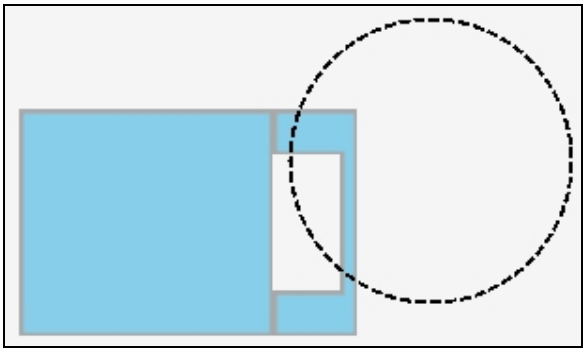
Daarom is ervoor gezorgd dat de shapes niet enkel kunnen beginnen te 'kleven' aan de GlueStick wanneer de gebruiker deze laatste beweegt, maar ook dat de shapes beginnen te kleven wanneer de gebruiker een shape beweegt tot tegen de GlueStick. Op deze manier kan een shape die moeilijk toegankelijk is, gemakkelijk weggehaald worden van zijn locatie en aan de GlueStick gekleefd worden.

6.3.2.5 FishNet

De volgende interactietechniek die in deze thesis behandeld wordt, is de techniek die *FishNet* is genoemd, een techniek die het mogelijk maakt objecten die veraf gelegen zijn tot bij de (hand van de) gebruiker te halen. De motivatie voor deze techniek is gevonden in de vaststelling dat bij het opbouwen en aanpassen van het interieur het vaak nodig is om shapes te verplaatsen over een grote afstand, en wel naar de locatie waar de gebruiker zich bevindt.

Bij deze techniek selecteert de gebruiker een shape die zich niet in zijn nabijheid bevindt (anders is de standaard verplaatsingstechniek sneller en meer voor de hand liggend) waarna de shape van plaats automatisch verandert: de shape komt onmiddellijk te staan daar waar de gebruiker zich bevindt.

Ook deze techniek is in het 'Extra's' submenu geplaatst. Wanneer de gebruiker deze techniek kiest, verschijnt er een cirkel op het scherm die over het scherm bewogen kan worden door de vinger op het scherm te verplaatsen, zoals te zien is in Figuur 6.19. Net als bij de Bot, is de verplaatsing van het FishNet versterkt ten opzichte van de beweging van de vinger om het probleem van de reikwijdte op te vangen, waarvan zowel de voor- als de nadelen reeds in paragraaf 6.3.2.2 besproken zijn.



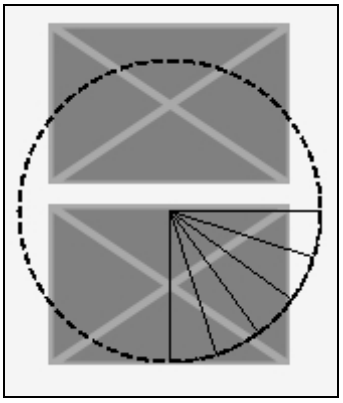
Figuur 6.19 FishNet

Ook de selectie van shapes gebeurt net als bij de bot: er wordt een interne timer gebruikt die start wanneer het midden van het FishNet zich boven een shape bevindt, en die de selectie uitvoert. Ook de feedback aan de gebruiker gebeurt op dezelfde manier. Voor de motivatie van deze techniek verwijs ik weer naar 6.3.2.2. Het verschil met de bot is dat er na selectie bij de bot een menu getoond wordt, terwijl bij het FishNet de geselecteerde shape zich direct verplaatst tot bij de gebruiker. Daar waar de bot dus als een verlengstuk van de vinger kan beschouwd worden, is het FishNet in de eerste plaats een verplaatsingstechniek.

Tijdens het uitproberen van deze techniek, bleek dit een techniek die goed werkte zolang de te verplaatsen shape niet kort in de buurt van andere shapes stond. Indien de shape zich immers in een regio bevond waar nog vele andere shapes stonden, bleek het niet zo eenvoudig meer om als gebruiker te bepalen welk object precies geselecteerd zou worden: voor de gebruiker die geconfronteerd wordt met de situatie zoals weergegeven in Figuur 6.20, is het bijvoorbeeld niet zonder meer duidelijk of hij nu de onderste of de bovenste kast aan het selecteren is.

Dit komt door de verhouding tussen de grootte van de cirkel en de afstand tussen het middelpunt van de cirkel en de randen van beide kasten. Het middelpunt van de cirkel wordt trouwens pas impliciet visueel gemarkeerd wanneer de selectie start en moet anders door de gebruiker zelf ingeschat worden. In het voorbeeld is de afstand tot de randen van de kasten slechts zeer klein ten opzichte van de grootte van de cirkel waardoor het als waarnemer moeilijk is uit te maken boven welke kast de cirkel zich precies bevindt.

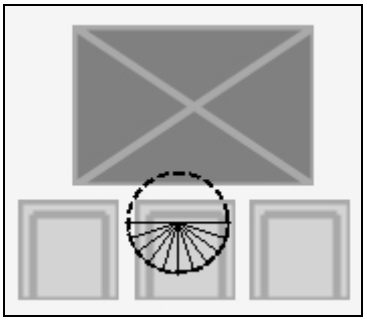
Gelieve hierbij trouwens op te merken dat Figuur 6.20 slechts een uitsnede is van het ganse ontwerpvenster en dat de kasten uit de figuur zich op een locatie bevinden waar de gebruiker niet zomaar bij kan (dit is althans de veronderstelling die deze techniek maakt, indien dit niet zo zou zijn, is de standaard verplaatsingstechniek meer aangewezen).



Figuur 6.20 FishNet: onduidelijke selectie

Om deze onduidelijkheid weg te werken, is ervoor gekozen om de grootte van het FishNet aan te passen afhankelijk van de omgeving: indien er andere shapes in de buurt zijn, wordt het FishNet kleiner, zoals te zien is in Figuur 6.21 (in de figuur is een kast opgenomen met dezelfde afmetingen als de kasten in Figuur 6.20 om de veranderde grootte van het FishNet te kunnen vaststellen). Voor het FishNet is geen traploze aanpassing van de grootte voorzien, maar worden drie verschillende groottes gebruikt, die proefondervindelijk zijn vastgelegd.

Achteraf bekeken, kunnen er wel wat bedenkingen gemaakt worden bij de manier waarop het FishNet is ontworpen. Ten eerste kan men zich afvragen waarom het FishNet zo groot moet weergegeven worden indien het tot onduidelijkheid bij de gebruiker kan leiden over welk object geselecteerd zal worden. Indien een kleiner FishNet dit probleem van onduidelijkheid niet heeft, dan pleit er nog weinig voor een groot FishNet, behalve misschien dat een groter FishNet makkelijk vanop een grotere afstand zichtbaar is. Doch -ten tweede- verbetert de situatie er niet echt op door het FishNet kleiner te maken: het bepalen van de selectie blijft moeilijk. De waarneming van de gebruiker staat immers opnieuw onder druk: dit keer niet omdat het middelpunt moeilijk in te schatten is, maar eenvoudigweg omdat het FishNet klein is en zich -voortvloeiend uit de ontwerpveronderstelling- op enige afstand van de gebruiker bevindt.



Figuur 6.21 FishNet: aanpassing grootte

Om aangehaalde bedenkingen, lijkt het huidige ontwerp niet optimaal. Het bepalen van het middelpunt van het FishNet (en dus het object dat geselecteerd zal worden) zou expliciet weergeven moeten worden, en wel in een opvallende kleur en grootte. Of misschien nog beter: in plaats van de gebruiker beter te informeren over het FishNet, zou de shape die het FishNet gaat 'vangen' geaccentueerd kunnen worden. Een shape die zich onder het FishNet bevindt, zou dan bijvoorbeeld in een andere kleur kunnen oplichten zodat de gebruiker meteen weet welk object hij naar zich toe gaat halen. Deze bedenkingen verdienen in een verdere uitwerking zeker de nodige aandacht.

6.3.2.6 DoubleTap

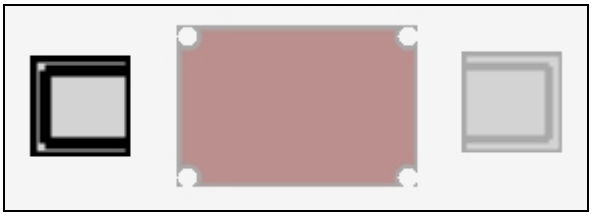
Bij de technieken die tot hiertoe in deze thesis behandeld zijn, ging het telkens om een interactietechniek waarbij de gebruiker met de (vinger van de) hand contact diende te maken met het bord. Die technieken deden slechts beroep op één hand van de gebruiker. Met de DoubleTap techniek wou ik iets ontwikkelen waarmee een gebruiker door *beide* handen te gebruiken kon komen tot een rijkere interactie met de applicatie.

Bij het beschouwen van de mogelijkheden hiervoor, werd ik al meteen geconfronteerd met een belangrijke beperking die gevormd wordt door het SmartBoard: het bord kan slechts van één inputdevice gegevens verwerken. Voor het besturingssysteem waarop de applicatie draait is er dus slechts één inputdevice (wanneer een gebruiker contact maakt met het bord is dit voor het besturingssysteem eigenlijk gewoon een beweging van een muis met ingedrukte muisknop) en dus ook slechts één coördinaat om te verwerken.

Voor de interactie betekent deze beperking dat het niet mogelijk is om met twee handen (tweemaal een coördinaat) tegelijk te werken, wat bijvoorbeeld handig zou zijn bij het plaatsen van shapes: een gebruiker zou tegelijkertijd de translatie en rotatie en/of het schalen van een shape kunnen uitvoeren. Dergelijke parallelle taken bleken dus niet mogelijk met het SmartBoard, waardoor gefocust is op een seriële taak die wel perfect is uit te voeren op het SmartBoard en waarbij de gebruiker beide handen kan gebruiken: het verplaatsen van één of meer objecten.

Door het snel na elkaar aanraken van een shape op het scherm (het principe is hetzelfde als een *double click* van de muis) wordt de DoubleTap geactiveerd. Om te bepalen of de twee contacten snel genoeg na elkaar gebeurden, wordt nagegaan of dit binnen de 400 milliseconden van elkaar was. Dit is dezelfde waarde die bij Unix en Microsoft Windows standaard gehanteerd wordt voor een *double click*. Door een waarde te nemen waarmee de gemiddelde gebruiker vertrouwd is, kan ervan uitgegaan worden dat de gebruiker deze methode snel onder de knie heeft.

Maar wat gebeurt er nu precies met de DoubleTap? Een shape waarop de gebruiker de DoubleTap uitvoert, wordt vervolgens geselecteerd en visueel aangepast weergegeven zoals te zien is in Figuur 6.22. Indien de gebruiker vervolgens op een andere locatie in het ontwerpvenster de DoubleTap uitvoert, dan zal het geselecteerde object onmiddellijk naar deze nieuwe locatie verplaatst worden.



Figuur 6.22 DoubleTap selectie

Ook het verplaatsen van meerdere objecten is mogelijk: de gebruiker voegt gewoon objecten toe aan de selectie door er een DoubleTap op uit te voeren. Het verwijderen van een object uit de selectie is tevens eenvoudig: nogmaals een DoubleTap op een reeds geselecteerde shape maakt de selectie ongedaan. Op deze manier kan de gebruiker makkelijk de selectie van een shape *togglen*.

Indien er echter meerdere shapes geselecteerd zijn en verplaatst worden omdat de gebruiker op een lege locatie van het ontwerpvenster heeft geklikt, ontstaat er een probleem met verschillende mogelijke oplossingen: er zijn x aantal objecten geselecteerd en de doellocatie bestaat uit één coördinaat. Een mogelijke oplossing is om het middenpunt te bepalen -voor de verplaatsing- van alle geselecteerde shapes en de shapes dezelfde verplaatsing te geven als de vector die gevormd wordt door dit middenpunt en de doellocatie. Een andere oplossing zou als verplaatsingsvector voor de shapes de vector kunnen nemen die gevormd wordt door het laatst geselecteerde object en de doellocatie, of door het eerst geselecteerde object en de doellocatie.

In de implementatie is voor het laatste gekozen: de eerst geselecteerde shape komt terecht op de doellocatie en de andere shapes maken dezelfde verplaatsing als deze shape. Alle genoemde alternatieven hebben als nadeel dat het voor de gebruiker moeilijk is om het uiteindelijke resultaat van de verplaatsing in te schatten (verplaatsing via SelectionBox heeft dit probleem niet). Ook is het mogelijk dat door de verplaatsing shapes buiten het ontwerpvenster vallen en niet meer zichtbaar zijn. Dit wordt in de applicatie niet opgevangen, in een verdere implementatie zou bijvoorbeeld een verplaatsing die er voor zorgt dat shapes buiten het venster vallen,

geblokkeerd kunnen worden met de nodige feedback. Of de applicatie zou de shapes in kwestie automatisch een andere plaats kunnen geven.

Een probleem bij de gekozen methode is dat de gebruiker zelf dient te onthouden welke shape hij als eerste heeft geselecteerd. Deze shape is immers bepalend voor de afstand die alle geselecteerde shapes zullen afleggen. In de huidige implementatie is deze shape nog niet visueel te onderscheiden van de andere shapes: dit zou in een volgende versie kunnen aangepast worden door de shape die bepalend is voor de verplaatsing visueel aangepast weer te geven.

Het verplaatsen van objecten met DoubleTap is ontwikkeld vanuit de doelstelling een *bi-manual* interactietechniek aan te bieden, maar uiteraard is de techniek ook perfect met één hand bruikbaar.

Persoonlijk vind ik ten slotte dat deze techniek in beperkte mate een echte hulp kan zijn voor de gebruiker: de gebruiker kan snel objecten verplaatsen door een DoubleTap met de ene hand na een DoubleTap met de andere hand en hierbij kunnen de bewegingen van de handen voor een stuk parrallel uitgevoerd worden (behalve uiteraard de DoubleTap zelf). Toch is de interactie beperkt: de verplaatsing van vele shapes tegelijk is voor de gebruiker moeilijk voorspelbaar.

6.3.2.7 Bag gesture

De interactietechniek die in deze paragraaf zal behandeld worden, maakt gebruik van *gestures* (zie 4.2.2). Zoals in paragraaf 6.3.2.6 uit de doeken is gedaan, is het op een SmartBoard niet mogelijk om invoer van meer dan één apparaat te verwerken. Om deze reden is collaboratie op het SmartBoard beperkt tot collaboratie waarbij de deelnemers hun activiteiten serieel uitvoeren ten opzichte van elkaar, en niet parallel. Met deze beperking in het achterhoofd ben ik op zoek gegaan naar een techniek die collaboratie ondersteunt.

Via de Bag kunnen gebruikers objecten met elkaar uitwisselen. Stel bijvoorbeeld dat twee gebruikers aan het bord staan en samen een interieur aan het ontwikkelen zijn (en hun taken serieel ten opzichte van elkaar uitvoeren). In dat geval wou ik een manier aanbieden die het mogelijk maakt dat gebruikers makkelijk shapes met elkaar kunnen uitwisselen zonder dat ze de shape in kwestie moeten verplaatsen naar de andere gebruiker.

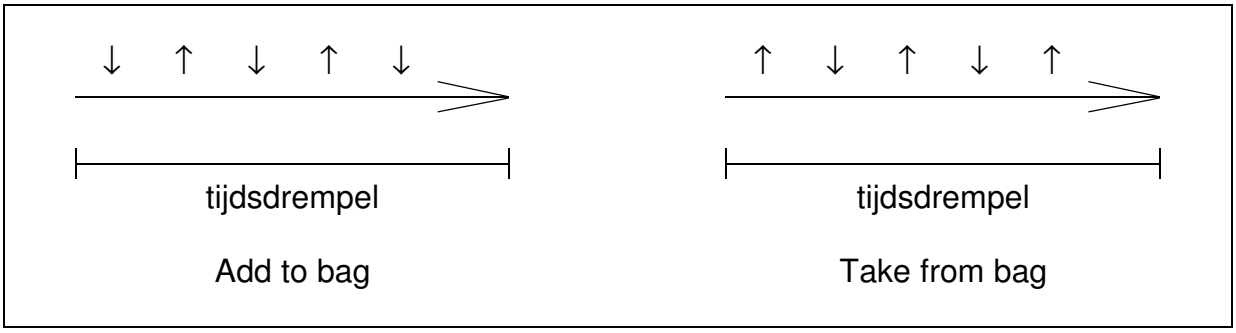
De Bag wordt gedeeld door alle gebruikers en een gebruiker kan er een shape in bewaren en een (niet noodzakelijk) andere gebruiker kan deze shape er vervolgens weer uithalen, ook al bevindt deze gebruiker zich op een plaats die ver van de locatie van de andere gebruiker verwijderd is.

Het toevoegen en verwijderen van shapes aan de Bag gebeurt door middel van *gestures*. Deze gestures moeten op een shape uitgevoerd worden -althans de eerste beweging van de gesture- of op een lege regio van het ontwerpvenster voor het toevoegen respectievelijk verwijderen van een shape, en zijn allebei bewegingen langs de verticale as. De hoofdbeweging voor het toevoegen van een shape aan de Bag is een neerwaartse beweging (net als wanneer een persoon echt iets in een tas

stopt), terwijl dit voor het uit de Bag nemen van een shape een opwaartse beweging is. Merk op dat deze operatie vergelijkbaar is met de klassieke *cut* en *paste* functies zoals die in vele programma's terug te vinden zijn.

Uiteraard konden de gestures niet zomaar gedefinieerd worden als een enkele opwaartse of neerwaartse beweging die start op een shape: dit zou immers leiden tot foute veronderstellingen omdat de gebruiker bijvoorbeeld niet noodzakelijk iets uit de zak wil halen wanneer hij een opwaartse beweging maakt in een lege regio van het ontwerpvenster (hij zou in dit geval een selection box kunnen starten).

Hierom zijn de gestures gedefinieerd zoals weergeven in Figuur 6.23. De gesture om een shape aan de Bag toe te voegen start én eindigt met een *neer*waartse beweging, terwijl dit voor het uitnemen van een shape uit de Bag een *op*waartse beweging is. Bovendien is er ook een bepaalde tijdsdrempel toegevoegd aan het herkenningsproces: de gestures moeten binnen de drie seconden uitgevoerd worden. Deze waarde (drie seconden) is proefondervindelijk bepaald als zijnde een ideale tijdspanne: een kortere tijdspanne legt de gebruiker een te zware eis op wat betreft zijn uitvoeringssnelheid, en een langere tijdspanne leidt tot een te groot risico voor foute interpretaties (acties interpreteren als gestures terwijl dit niet de bedoeling van de gebruiker was).



Figuur 6.23 Bag gesture

De actuele implementatie van de Bag biedt momenteel enkel de mogelijkheid om één enkel object in de Bag te stoppen. Het loont echter de moeite om in een eventuele toekomstige uitwerking ook het bewaren van meerdere shapes te ondersteunen. Hiervoor dient dan wel de afhandeling van het tevoorschijn halen van shapes te worden uitgebreid: er zitten immers meerdere shapes in de Bag. Wanneer een gebruiker in dergelijk geval de gesture uitvoert om een shape uit de Bag te halen, zouden deze bijvoorbeeld in miniatuurweergave in een FlowMenu getoond kunnen worden, waarin dan de gebruiker zijn keuze kan maken.

Een andere mogelijke uitbreiding bestaat erin om -refererend naar de gemaakte vergelijking met *cut* en *paste*- ook de *copy*-functie via gestures te ondersteunen: wanneer de gebruiker *op een shape* de gesture uitvoert om een shape uit te Bag te halen, dan wordt een kopie van de shape in de Bag gestopt. Bovendien zou men zich de vraag kunnen stellen of het voldoende is te steunen op sociale protocollen om de gebruikers elkaar te laten weten dat ze een object nodig hebben of dat ze een

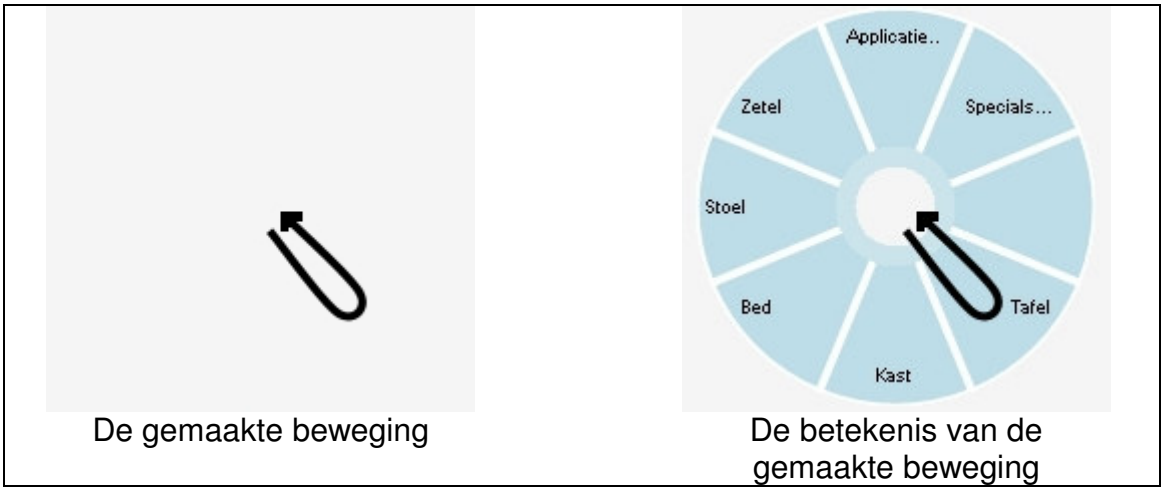
object in de Bag hebben gestopt. Een visuele representatie van de Bag, en een feedbackmechanisme voor elke gebruiker wanneer de inhoud van de Bag wijzigt, zijn aandachtspunten voor een eventueel verder onderzoek en uitwerking.

Het beoordelen van de waarde van deze techniek voor collaboratieve doeleinden is lastig: het SmartBoard leent zich sowieso al moeilijk voor dergelijke taken (herinner de beperking van seriële taken). Doch het is niet zo moeilijk om zich voor te stellen dat -er even vanuitgaande dat een interactievlak wél meerdere simultane gebruikers ondersteunt- het principe van de gedeelde opslagruimte zijn nut heeft indien de gebruikers fysiek niet in elkaars buurt staan.

6.3.2.8 Marking ahead

Voor een snellere interactie is het FlowMenu ook voorzien van ondersteuning voor *marking ahead*: hierbij kan de gebruiker een menukeuze maken zonder dat hij hiervoor moet wachten tot het FlowMenu verschijnt. In plaats daarvan kan de gebruiker meteen de beweging maken die tot de gewenste menukeuze leidt. Deze beweging is exact dezelfde wat betreft afgelegde weg als de beweging die gemaakt zou zijn indien de gebruiker via het FlowMenu zijn keuze gemaakt zou hebben. Het enige bijkomende verschil -naast het niet tonen van het menu- is een verschil qua tijdsverloop: bij *marking ahead* is er geen *press-and-wait* (zie paragraaf 6.3.1.1).

In Figuur 6.24 is te zien hoe *marking ahead* precies werkt: de linkerkant van de figuur geeft weer welke beweging de gebruiker maakt met de vinger op het bord (weergegeven door de zwarte pijl) om een tafel toe te voegen. Het valt op, en dat is ook de essentie van het *marking ahead*, dat er geen FlowMenu getoond wordt. De rechterkant van de figuur toont -ter verduidelijking- hoe dezelfde beweging in het FlowMenu zou geleid hebben tot het toevoegen van een tafel, maar merk op dat het FlowMenu bij *marking ahead* dus niet getoond wordt.



De gemaakte beweging

De betekenis van de gemaakte beweging

Figuur 6.24 Marking ahead

Wanneer de gebruiker de vinger op het bord legt en vervolgens beweegt, gaat de applicatie onmiddellijk na of de gemaakte beweging tot een menukeuze kan leiden. Indien dit het geval is, zoals in Figuur 6.24, dan wordt de keuze uitgevoerd. In tegenstelling tot het 'tragere' gebruik van het FlowMenu is er dus geen time-out die moet verstrijken vooraleer het menu verschijnt en een keuze gemaakt kan worden, waardoor het toevoegen van shapes bijzonder snel kan gebeuren. De enige voorwaarde voor een vlot en foutloos gebruik van deze techniek is dat de gebruiker voldoende kennis heeft van de plaatsing van de menu-items. Hierdoor is *marking ahead* weggelegd voor de gevorderde gebruiker.

Deze techniek is de laatste techniek die toegevoegd is aan de applicatie en zorgde voor nogal wat integratiemoeilijkheden met technieken die reeds aanwezig waren (vandaar ook dat deze techniek als laatste aan bod komt in dit implementatie-hoofdstuk). Met name de aanwezigheid van de Bag-gestures legde nogal wat beperkingen op aan het gebruik van *marking ahead*. Dit is eigenlijk ook niet verwonderlijk: allebei zijn het technieken waarbij korte bewegingen (gestures) verbonden worden aan een bepaalde functionaliteit.

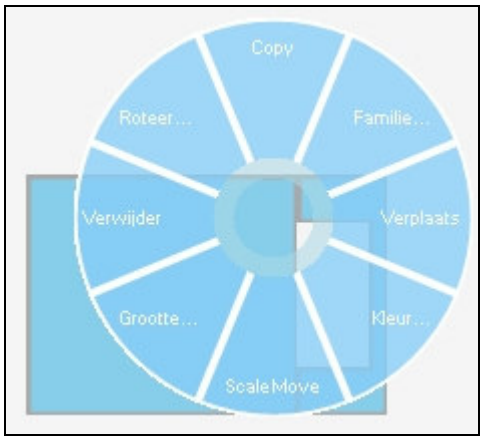
Ter herinnering: bij de Bag-gestures (6.3.2.7) zijn het korte bewegingen langs de verticale as die leiden tot ofwel het toevoegen van een shape aan de Bag, ofwel het uitnemen van een shape uit de Bag. Maar ook het toevoegen van een kast via *marking ahead* gebeurt met een verticale beweging (naar onder en vervolgens weer naar boven).

Een Bag-gesture kan in twee situaties uitgevoerd worden: op een shape of op een lege plaats van het ontwerpvenster. In het eerste geval begint de gesture met een neerwaartse beweging (om de shape in de Bag te stoppen) en in het tweede geval met een opwaartse beweging (om de shape uit de Bag te nemen).

Aangezien *marking ahead* vooral bedoeld is om aan gevorderde gebruikers een snelheidswinst te kunnen aanbieden voor veelgebruikte operaties, is ervoor gekozen om niet alle operaties via *marking ahead* aan te bieden: enkel het toevoegen van

shapes, het verplaatsen van een shape en het verwijderen van een shape zijn onderstuend. Deze menu-items zijn zodanig in het FlowMenu geplaatst dat er geen conflict kan ontstaan met *marking ahead*.

Ter illustratie: aan de rechterkant van Figuur 6.24 is het FlowMenu weergegeven dat getoond wordt wanneer een gebruiker op een lege plaats in het ontwerpvenster het menu oproept. Op dezelfde lege plaats zou de gebruiker ook een Bag-gesture kunnen uitvoeren (meerbepaald een gesture om een shape uit de zak te nemen). Deze beweging zou echter ook een menukeuze in het FlowMenu via *marking ahead* kunnen zijn. Vandaar dat op de locatie bovenaan in het NormalFlowMenu (‘op 12u’, zie de rechterkant van Figuur 6.24) een submenu geplaatst is dat niet via *marking ahead* bereikbaar moet zijn (het ‘Applicatie’ submenu).



Figuur 6.25 ShapeFlowMenu: aangepaste plaatsing menu-items

Op dezelfde manier is ervoor gezorgd dat de menu-keuzes voor het verwijderen en verplaatsen van een shape in het ShapeFlowMenu niet met de Bag-gestures kunnen conflicteren: zij kregen een plaats links (‘op 9u’) en rechts (‘op 3u’) in het ShapeFlowMenu.

7 Usability test

Als onderdeel van de thesis, heb ik een beperkte gebruikerstest gehouden om de in de vorige paragrafen beschreven implementatie uit te proberen. De deelnemers kregen hierbij een eenvoudige opdracht waarbij ze de verschillende interactie-technieken moesten uitproberen.

7.1 Setup

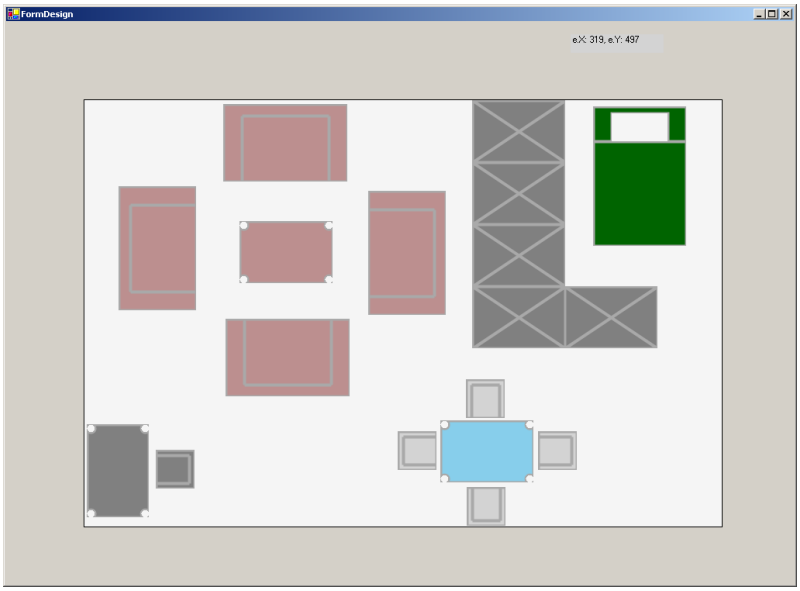
De groep deelnemers bestond uit vier personen: één laatstejaarsstudent Informatica (afstudeerrichting HCI) en drie medewerkers van het Expertisecentrum voor Digitale Media te Diepenbeek. Allemaal kunnen ze beschouwd worden als ervaren computer-gebruikers, en tevens zijn ze vertrouwd met het domein van HCI. Er was één vrouwelijke deelnemer.

De usability test heb ik gehouden op een ogenblik in het ontwikkelingstraject van de thesis waarop nog niet alle technieken volledig geïmplementeerd waren. De test zelf nam per proefpersoon ongeveer een kwartier in beslag.

Tijdens het eerste deel van de test legde ik kort de algemene werking van de applicatie uit: ik deed onder andere het FlowMenu uit de doeken, en behandelde de drie verschillende manieren waarop shapes te verplaatsen waren (de uiteindelijke implementatie biedt meer mogelijkheden, maar was ten tijde van de test nog niet helemaal afgerond).

Vervolgens liet ik de deelnemers kort de applicatie wat uitproberen om eraan te wennen. Daarna liet ik hen de opdracht weten en kregen de deelnemers een schermafdruk te zien van een vooraf gemaakte interieurinrichting (zie Figuur 7.1) met daarbij de opdracht deze na te bouwen startende vanaf een leeg ontwerpvenster. Ook stonden op het blad de verschillende manieren om een shape te verplaatsen nog eens herhaald. Om de opdracht uit te voeren, moesten de deelnemers de shapes toevoegen, roteren, verplaatsen en van kleur veranderen, de grootte van de shapes wijzigen was voor de opdracht niet nodig. Na het uitvoeren van de opdracht stelde ik de deelnemers vervolgens een aantal vragen over hun ervaringen.

De focus van de test heb ik gelegd bij de verschillende technieken waarmee shapes verplaatst kunnen worden: via het FlowMenu, via de DoubleTap en via de Bag (ook al is de Bag ontwikkeld vanuit een collaboratief perspectief, is deze perfect individueel bruikbaar als verplaatsingstechniek).



Figuur 7.1 Usability test: opstelling

7.2 Hypotheses

Bij de verplaatsing via het FlowMenu kan de gebruiker het resultaat van de verplaatsing zien tijdens het maken van de beweging en kan hij ten allen tijde bijsturen (tot hij de shape loslaat), terwijl hij bij de andere twee technieken het resultaat van de verplaatsing pas ziet nadat de techniek is uitgevoerd. Ik veronderstelde vooraf dat dit de deelnemers ook zou opvallen en dat ze hierdoor vooral de verplaatsing via het FlowMenu zouden gebruiken. En bij het gebruik van de DoubleTap ging ik ervan uit dat ze de snelheid ervan zouden appreciëren. Tot slot veronderstelde ik ook dat de deelnemers het werken met de Bag gestures intuïtief zouden vinden, m.n. de definities van de gestures, omdat de metafoor van de zak vertrouwd is.

7.3 Experiment en resultaten

Wat opviel als verschil tussen de werkwijze waarop de deelnemers de opdracht uitvoerden was dat de vrouwelijke deelnemer eerst alle stoelen die ze nodig had toevoegde aan het ontwerpvenster -dus vijf maal via het FlowMenu een stoel toevoegen- en vervolgens al deze stoelen één voor één de juiste positie gaf. Dit herhaalde ze vervolgens voor de andere soorten shapes. De mannelijke deelnemers pakten het anders aan: ze werkten regio per regio, en zetten na het toevoegen van een shape deze onmiddellijk op de juiste plaats, wat in totaal langer duurde. Toen ik de vrouwelijke deelnemer hierover een vraag stelde motiveerde ze haar strategie met de opmerking dat ze op deze manier voor elke operatie, het toevoegen of verplaatsen van een shape, slechts één maal de bijhorende menukeuze hoefde te zoeken in het FlowMenu en dit hierna een aantal maal kon herhalen.

Een algemene opmerking die naar voren kwam, was dat de verplaatsing via het FlowMenu de nauwkeurigste was: de shape kan hiermee precies op de juiste plaats

gezet worden, wat met de Bag of DoubleTap veel moeilijker is omdat het resultaat moeilijk te voorspellen is. Dit bevestigt wat ik vooraf dacht slechts voor een gedeelte: de nauwkeurigheid van het FlowMenu werd bevestigd, maar de gebruikers lieten de andere twee technieken niet links liggen, maar gebruikten de Bag of DoubleTap (vooral de DoubleTap) voor de groffe verplaatsingen en deden het fijne afregelwerk via het FlowMenu.

De DoubleTap werd onthaald als een techniek die vooral snelheid te bieden heeft, maar met flinke tekortkomingen voor kleine verplaatsingen heeft te kampen. Vooraf had ik geen woord gerept over de bestaansreden van deze techniek (die gemotiveerd is door *bi-manual* interactie) om te zien of de techniek uit zichzelf uitnodigt tot *bi-manual* interactie. Tijdens de proef heb ik echter niemand met beide handen zien werken. Een verklaring zou kunnen zijn dat de proef zich hier niet echt toe leende: de deelnemers moesten een voorafbepaalde opstelling nabouwen en konden dus steeds de shape toevoegen in de buurt van de doellocatie. Hierdoor werd de *bi-manual* interactie wellicht niet voldoende gestimuleerd.

Eén deelnemer merkte op dat het verplaatsen via DoubleTap van een shape over een afstand die kleiner is dan de grootte of breedte van de shape, niet intuïtief werkt: de DoubleTap doellocatie moet immers buiten een shape liggen. Dit bevestigt nogmaals de vaststelling dat de DoubleTap niet geschikt is voor korte verplaatsingen.

Het werken met de Bag was niet echt populair. De gesture op zich bleek wel gemakkelijk uit te voeren, maar ten opzichte van de DoubleTap bood deze techniek weinig toegevoegde waarde: beide technieken hebben -zoals reeds uitvoerig besproken- gemeen dat het voor de gebruiker moeilijk is het resultaat te voorspellen, maar blijkbaar maakt de uitvoeringssnelheid van de DoubleTap de Bag een beetje overbodig.

Over het FlowMenu zelf werden ook opmerkingen gegeven. Eén gebruiker stelde dat de plaatsing van de menu-items niet gebruiksvriendelijk was. Het octant dat zich voor deze rechtshandige gebruiker onder de hand bevond, bevatte een menu-item terwijl er andere octanten nog leeg waren. Hierdoor kan de gebruiker het deels bedekte menu-item moeilijk lezen.

Een andere gebruiker merkte op dat het misschien nuttig was om de selectie van menu-items niet alleen te laten gebeuren wanneer de vinger terugkeerde naar het midden (zie paragraaf 6.3.1.4), maar dat dit ook zou kunnen bij het verlaten van het FlowMenu langs de buitenkant van de cirkel omdat hij dit natuurlijker vond. De andere deelnemers hebben hierover echter geen opmerking gemaakt.

Wat de ruimtelijke context betreft, gaven een aantal gebruikers aan dat het gebruik van de technieken moeilijk wordt wanneer deze onderaan het bord uitgevoerd moeten worden. Bijvoorbeeld het uitvoeren van de gesture op een shape die zich helemaal onderaan in het ontwerpvenster bevindt, is zeer lastig omdat de gebruiker hierbij een relatief lang patroon van bewegingen moet uitvoeren (die door een weinig vergevingsgezinde herkenner verwerkt worden) en waarbij zijn hand zich ter hoogte van het middel bevindt waardoor de beweging moeilijk te coördineren is. Vooral voor

bewegingen met een hogere motorische complexiteit speelt dit nadeel sterk. Eenvoudigere bewegingen als de DoubleTap hebben van dit fenomeen veel minder last.

Tot slot gaven de deelnemers nog enkele algemene opmerkingen over wenselijke uitbreidingen. Eén gebruiker zou het nuttig gevonden hebben indien hij bij het toevoegen van een shape meteen het aantal shapes zou kunnen aangegeven hebben dat hij nodig heeft. Bij het maken van een eetkamer zou men op deze manier dan bijvoorbeeld vier stoelen tegelijk kunnen toevoegen. Een andere gebruiker wees erop dat het toevoegen van een shape en het verplaatsen van een shape best aan elkaar gekoppeld konden worden: na het toevoegen van een shape zou die dan vervolgens verplaatst kunnen worden zolang de gebruiker de vinger tegen het scherm houdt.

8 Conclusie en toekomstig werk

In hoofdstukken 6 en 7 is de implementatie, het tweede deel van de thesis, aan bod gekomen. Hierbij is de gemaakte applicatie beschreven en de verschillende interactietechnieken besproken die de applicatie biedt. Tevens is de interactie van de gebruiker met de applicatie behandeld en de moeilijkheden die daarbij de kop opstaken. Als laatste is in een apart hoofdstuk een beschrijving gegeven van de informele usability test, alsook een bespreking van de resultaten ervan.

In dit laatste hoofdstuk worden een aantal conclusies van het implementatiehoofdstuk op een rijtje gezet, en tevens de weg voorbereid voor mogelijke toekomstige doorontwikkelingen van de implementatie.

Als noot vooraf aan het formuleren van conclusies, dient nog even opgemerkt te worden dat de beschreven implementatie geen volwaardig programma is waarmee een interieur kan uitgewerkt worden. Hiervoor zijn de geboden functionaliteiten eenmaal ontoereikend. Met dit applicatiedomein is gekozen voor het ontwerpen van een interieur als 'kapstok' voor de beschreven interactietechnieken.

8.1 Conclusies

8.1.1 FlowMenu

Een eerste belangrijke vaststelling is dat het FlowMenu nogal wat beperkingen oplegt aan de gebruikersinterface. Ten eerste dient er steeds een zone rond de interactieruimte beschikbaar te zijn zodat het FlowMenu steeds getoond kan worden. Dit is schermruimte die -ook al worden de schermen en hun resoluties steeds groter- toch niet onbeperkt beschikbaar is waardoor het eigenlijk zonde is die ruimte te moeten opofferen. Doch men zou de ruimte voor GUI-elementen kunnen gebruiken die de gebruiker niet nodig heeft bij het gebruik van het FlowMenu (bijvoorbeeld toegang tot de online helpfunctie), of men zou kunnen afstappen van het idee van een volledig rond FlowMenu (zie paragraaf 4.2.1 voor de besproken alternatieven).

Wat me ook opvalt is dat het FlowMenu zoals het in de applicatie is geïmplementeerd toch behoorlijk wat interactiemogelijkheden kan aanbieden: verschillende soorten shapes kunnen worden toegevoegd en gemanipuleerd (rotatie, schalen, verplaatsen) en sommige eigenschappen kunnen worden aangepast. Men kan dus stellen dat een FlowMenu als menusysteem goed voldoet voor een applicatie voor interieurontwerp. Er zouden eventueel gemakkelijk nog menukeuzes kunnen worden toegevoegd, al is het niet vanzelfsprekend om in een menusysteem dat een vast aantal menukeuzes toont (acht octanten), steeds ervoor te zorgen dat de getoonde keuzes logisch samenhangen zonder te veel submenu's te moeten creëren. Als alternatief voor het onderbrengen van logisch samenhangende opties in een apart submenu, kan men overwegen om binnen hetzelfde submenu toch verschillende groepen van samenhangende opties te plaatsen en deze groepen van elkaar te scheiden door lege octanten te gebruiken.

Een andere belangrijke beperking die tijdens het ontwikkelen al snel naar boven kwam, is de tekst die in een FlowMenu weergegeven kan worden zoals eerder in hoofdstuk 6 aangehaald is. De ruimte is beperkt en bovendien anders georiënteerd voor elk van de acht octanten. De teksten roteren al naargelang het octant kan het probleem wellicht gedeeltelijk oplossen, maar introduceert het probleem van de leesbaarheid. Een alternatief is het toestaan dat tekst van een octant de grenzen van het octant mag overschrijden, maar ook dit leidt tot nieuwe risico's van overlappende teksten. Het feit is en blijft dat de octanten van een FlowMenu een vorm hebben die het plaatsen van tekst voor een ontwikkelaar niet zo eenvoudig maakt als bij een klassiek menu.

Ook een beperkende factor is de invoer waar je het als ontwikkelaar mee moet doen: een X- en een Y-coördinaat en de status van de muisknop (die *down* is als de gebruiker de vinger tegen het scherm drukt). Bijvoorbeeld bij het bepalen van de selectie met de Bot was dit een probleem: hoe bepalen wanneer de gebruiker een shape wil selecteren? Als oplossing is gekozen voor het afleiden van de intentie van de gebruiker uit de tijdsperiode dat de Bot zich boven een shape bevindt (met een timer).

8.1.2 Probleem reikwijdte

In hoofdstuk 6 zijn een aantal technieken geïntroduceerd die een antwoord wilden bieden op het probleem van de reikwijdte. De Bot, ScaleMove en het FishNet zijn alledrie technieken die het mogelijk maken om als gebruiker gebieden te bereiken die fysiek onbereikbaar (zouden kunnen) zijn. ScaleMove dient voor het verplaatsen van objecten, maar de gebruiker betaalt de vergrote reikwijdte met een verlies aan precisie. Deze afweging geldt ook voor de Bot en het FishNet, die objecten selecteren. Een mogelijke aanpassing van de wijze waarop de beweging van de hand vertaald wordt in de beweging van de shape, de Bot of het FishNet zou dit nadeel kunnen elimineren.

Bij de drie genoemde technieken moest tijdens de ontwikkeling echter ook vastgesteld worden dat de fysieke onbereikbaarheid eigenlijk ook wel eens zou kunnen impliceren dat er een soort van waarnemingsprobleem kan ontstaan: waar de gebruiker niet bij kan, zou wel eens een locatie kunnen zijn die hij niet zo goed kan zien. Met name bij de selectie van objecten (de functie die de Bot en het FishNet bieden) zorgt dit voor een probleem: hoe zorg je ervoor dat de gebruiker zeker is van de selectie die gemaakt wordt? Het gebruik van extra visuele markeringen op de shape in kwestie of het gebruik van auditieve feedback zou deze technieken zeker verbeteren.

8.1.3 Ondersteuning collaboratie en bi-manual input

De poging om een techniek te ontwikkelen voor het SmartBoard die collaboratief werk ondersteunt kan niet echt helemaal naar waarde geschat worden: deze techniek is niet door proefpersonen uitgeprobeerd in een collaboratieve context. Wat wel beweerd mag worden, is dat het SmartBoard (met het gebrek aan ondersteuning voor simultane invoer van meerdere gebruikers) niet geschikt is voor taken waarbij de verschillende deelnemers subtaken parallel uitvoeren.

Eenzelfde soort conclusie dringt zich op voor de DoubleTap: deze techniek was oorspronkelijk bedoeld om het volle potentieel van de gebruiker (naar gebruik van beide handen toe) te benutten. De genoemde beperking van het SmartBoard verhindert het gebruik van twee handen *tegelijk*. De DoubleTap is een poging om door middel van een serieel gebruik van beide handen het verplaatsen van objecten te versnellen. De gebruiker kan deze techniek trouwens ook met één hand gebruiken en kan zelfs meerdere objecten tegelijk verplaatsen, maar dit leidt tot moeilijkheden wat betreft het bepalen van de exacte doellocatie van alle shapes: de shapes zouden eventueel niet meer zichtbaar kunnen zijn omdat ze buiten het ontwerpvenster vallen. Hiervoor zou een volgende versie een oplossing kunnen vinden in bijvoorbeeld het annuleren van de operatie die tot die situatie leidt.

8.2 Usability test

De usability test leidde ook tot een aantal belangrijke conclusies en suggesties voor verbeteringen. Een belangrijke conclusie is dat de fysieke context de interactie kan bemoeilijken. Wanneer een gebruiker voor het SmartBoard staat, dan bevindt de onderkant van het bord zich ter hoogte van het middel van de gebruiker. Als de gebruiker een beweging wil maken aan de onderkant van het bord, dan is het moeilijk om een complexe beweging te maken: een tekening maken op ooghoogte lukt makkelijker dan ter hoogte van het middel. Iedereen heeft wellicht al ervaren (bij zichzelf of bij iemand anders) dat gebruikers die aan de onderkant van een bord een tekening maken door de benen buigen om er beter bij te kunnen of zich zijdelings draaien en het hoofd in een grotere hoek draaien ten opzichte van het bord zodat ze beter kunnen zien wat ze doen.

Het is dan ook licht ironisch dat de klassieke schoolborden verticaal verstelbaar zijn en dus een antwoord hebben voor dit probleem, terwijl de producent van het SmartBoard hier geen standaardoplossing voor biedt. Hierbij moet wel de opmerking gemaakt worden dat het technisch perfect mogelijk moet zijn om een SmartBoard te monteren op een constructie die wel verticaal verstelbaar is.

In ieder geval zorgt dit probleem ervoor dat gebruiker veel moeilijkheden hebben om motorisch complexe bewegingen uit te voeren onderaan het bord. De waarde van de beschreven interactietechnieken wordt hierdoor flink aangetast: het uitvoeren van de Bag gestures bleek bijvoorbeeld bijzonder moeilijk.

Een andere conclusie van de gebruikerstest is dat de geteste verplaatsings-technieken (Bag, DoubleTap, via FlowMenu) een verschillende toepassingsnuance hebben: voor groffe verplaatsingen zijn de Bag en de DoubleTap prima, maar voor het fijnere werk is verplaatsing via het FlowMenu nodig. De genoemde Bag bleek trouwens niet veel toegevoegde waarde te bieden, maar de Bag is dan ook niet op de eerste plaats bedoeld als een verplaatsingstechniek, maar als een hulpmiddel voor collaboratie.

De gebruikerstest legde ook nog eens het gevaar bloot waar Guimbretière in [26] voor waarschuwde: het octant dat zich rechtsonder bevindt zou zo veel mogelijk vrij

moeten gehouden worden. In de actuele implementatie is daar bijgevolg uiteraard rekening mee gehouden.

8.3 Toekomstig werk

Naast de genoemde toekomstige uitbreiding, zijn er nog een aantal mogelijke wijzigingen die het werken met de applicatie zouden kunnen verbeteren of vervolledigen. Zo is het momenteel niet mogelijk om de gemaakte opstelling op schijf te bewaren en weer in te laden.

Het wijzigen van de grootte van een shape via de Bot is in de huidige versie niet geïmplementeerd. Hierbij zou de Bot dan de vorm kunnen krijgen van een hand met uitgestoken wijsvinger om de metafoor 'verlengstuk van de vinger' kracht bij te zetten. Deze visuele weergave van de Bot zou trouwens sowieso (ook bijvoorbeeld bij gewone selectie via de Bot) bijdragen tot een beter begrip van de Bot-functie.

Een ander punt dat verdere aandacht verdient, is de moeilijkheid van het bepalen van de locatie en de grootte van de Sweeper en GlueStick. Actueel worden deze immers toegevoegd op de locatie waar de gebruiker de vinger tegen het scherm houdt, maar dit zou een plaats kunnen zijn waar andere shapes in de buurt staan, waardoor het risico bestaat op overlapping (wat ervoor zorgt dat de shapes steeds geveegd zullen worden). Een aanpassing van de breedte van de Sweeper of GlueStick kan het probleem niet volledig oplossen dus er zou onderzocht kunnen worden in welke mate rotatie en plaatsen van de Sweeper of GlueStick op een andere, nabije locatie een oplossing kunnen bieden.

Het werken met de Bag is al genoemd als zijnde een techniek die op het SmartBoard niet echt tot zijn recht kan komen. Als we er echter even van uitgaan dat het interactievlak wél meerdere simultane invoer ondersteunt, dan zou de Bag als interactietechniek tussen de verschillende gebruikers aanzienlijk verbeterd kunnen worden door het ondersteunen van meerdere objecten. Een gebruiker kan dan meerdere objecten bevatten, en bij het tevoorschijn halen van objecten uit de Bag zou de gebruiker dan een keuze kunnen maken uit de aanwezige objecten.

Tot slot stelden de deelnemers aan de usability test nog enkele waardevolle mogelijke uitbreidingen voor die ik niet heb weerhouden voor de implementatie, maar die zeker een nader onderzoek waard zijn. Zo is het combineren van het toevoegen en het verplaatsen van een shape zeker de moeite waard: de gebruiker hoeft dan niet telkens een extra actie in het menu uit te voeren om de shape te verplaatsen indien hij dit wenst. En als de gebruiker de shape niet zou willen verplaatsen, kan hij deze gewoon loslaten. Een uitbreiding die ook interessant zou zijn, is de mogelijkheid bieden om bij het toevoegen van een object het aantal gewenste objecten te kunnen specificeren.

9 Bibliografie

[1] Jun Rekimoto (1998) A Multiple Device Approach for Supporting Whiteboard-based interactions. In CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems, pages 344-351. ACM Press/Addison-Wesley Publishing Co.

[2] Daniel M. Russel, Rich Gossweiler (2001) On the Design of Personal & Communal Large Information Scale Appliances. In Proceedings of ACM UbiComp Conference, Atlanta, GA, USA. New York:ACM Press.

[3] Stephen Voida, Elizabeth D. Mynatt (2005) Context Histories, Activities, and Abstractions: Ubiquitous Computing Support for Individual and Collaborative Work. Position paper for the 1st International Workshop on Exploiting Context Histories in Smart Environments, held in conjunction with the 3rd International Conference on Pervasive Computing (PERVASIVE 2005), Munich, Germany, May 8–13.

[4] Stephen Voida, Elizabeth D. Mynatt, Blair MacIntyre, Gregory M. Corso (2002) Integrating Virtual and Physical Context to Support Knowledge Workers. IEEE Pervasive Computing, 1(3), 73–79.

[5] Brad Johanson, Armando Fox, Terry Winograd (2002) The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms. IEEE Pervasive Computing, 1 (2), 67-74.

[6] Elizabeth D. Mynatt, Takeo Igarashi, W. Keith Edwards, Anthony LaMarca (1999) Flatland: New Dimensions in Office Whiteboards. In Proc. CHI 1999, ACM Press, 346-353.

[7] Quentin Stafford-Fraser, Peter Robinson (1996) BrightBoard: A Video-Augmented Environment. Proceedings of CHI ’96. Vancouver, Canada, 134-131.

[8] Daniel Vogel, Ravin Balakrishnan (2004) Interactive Public Ambient Displays: Transitioning from Implicit to Explicit, Public to Personal, Interaction with Multiple Users. In Proc. UIST 2004. ACM Press, 137-146.

[9] Yvonne Rogers, Siân Lindley (2004) Collaborating around large interactive displays: which way is best to meet? Interacting With Computers, 16, 1133-1152.

[10] Hal Eden, Eva Hornecker, Eric Scharff (2002) Multilevel Design and Role Play: Experiences in Assessing Support for Neighborhood Participation in Design. In: Proc. DIS'2002, ACM Press (2002), 387-392.

[11] Brad A. Myers, Herb Stiel, Robert Gargiulo (1998) Collaboration Using Multiple PDAs Connected to a PC. In CSCW '98: Proceedings of the 1998

ACM conference on Computer supported cooperative work, pages 285-294. ACM Press.

[12] Eling Rønby Pedersen, Kim McCall, Thomas P. Moran, Frank G. Halasz (1993) Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. Proceedings of the InterCHI conference, Amsterdam April 1993, ACM, 1993.

[13] Gerd Kortuem, Christian Kray (2005) HCI issues of dispersed public displays. In Workshop on Distributed Display Environments at CHI 2005. 2005.

[14] Takeo Igarashi, Elizabeth D. Mynatt, W. Keith Edwards, Anthony LaMarca (1999) Demonstrating Flatland User Interfaces. In Proceedings of ACM SIGCHI '99, Conference on Human Factors in Computing Systems, pp. 27-28.

[15] Harry Brignull, Shahram Izadi, Geraldine Fitzpatrick, Yvonne Rogers, Tom Rodden (2004) The introduction of a Shared Interactive Surface into a Communal Space. Proceedings of CSCW 2004, ACM Press.

[16] Patrick Baudisch, Peter Tandler (2002) ContextWall: A multi-user workbench for independent work, peer-to-peer interaction, and whiteboard collaboration. Unpublished manuscript.

[17] Clifton Forlines, Daniel Vogel, Ravin Balakrishnan (2006) HybridPointing: Fluid Switching Between Absolute and Relative Pointing with a Direct Input Device. In Proceedings of the 19th Annual ACM Symposium on User interface Software and Technology, UIST '06. ACM Press, New York, NY, 211-2

[18] Clifton Forlines, Daniel Vogel, Nicholas Kong, Ravin Balakrishnan (2006) Absolute vs. Relative Direct Pen Input. Mitsubishi Electric Research Labs Tech Report, TR2006-066.

[19] Mark Ashdown, Peter Robinson (2005) Remote Collaboration on desk-sized displays. Computer animations and virtual worlds 16(1), pp 41-51.

[20] Gordon Paul Kurtenbach (1993) The design and evaluation of marking menus. Ph.D. Thesis. Toronto, Canada: Department of Computer Science, University of Toronto.

[21] Wiseman, N. E., Lemke, H. U., & Hiles, J. O. (1969) PIXIE: A New Approach to Graphical Man-machine Communication. Proceedings of 1969 CAD Conference Southhampton, 463, IEEE Conference Publication 51.

[22] Don Hopkins (1991) The Design and Implementation of Pie Menus, There're Fast, Easy, and Self-Revealing. Originally published in Dr. Dobb's Journal, Dec. 1991.

[23] DENIM. World Wide Web. http://dub.washington.edu/denim/

[24] Callahan, J., Hopkins, D., Weiser, M., & Shneiderman, B. (1988) An empirical comparison of pie vs. linear menus. Proceedings of the CHI '88 Conference on Human Factors in Computing Systems, 95-100, New York: ACM.

[25] K. Perlin (1998) Continuous Stylus-Based Text Entry. In *Proceedings of UIST'98,* pages 215-216, 1998.

[26] François Guimbretière, Terry Winograd (2000) FlowMenu: Combining Command, Text, and Data Entry. Proceedings of the 13th annual ACM symposium on User interface software and technology, p.213-216, November 06-08, 2000, San Diego, California, United States

[27] François Guimbretiere (2002) Measuring FlowMenu Performance. Technical report, Stanford CS, 2001. CS-TR-2001-02.

[28] Blinkenstorfer, C. H. (1995) Graffiti. Pen Computing, pp. 30–31

[29] Fitts, P.M. (1954) The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. Journal of Experimental Psychology 47, 381–391, 1954.

[30] Dean Rubine (1991) The automatic Recognition of Gestures. PhD thesis, Carnegie Mellon University, December 1991.

[31] Brad A. Myers, Yu Shan A. Chuang, Marsha Tjandra, Mon-chu Chen, Chun-Kwok Lee (2001) Floor control in a Highly Collaborative Co-Located Task. Unpublished.

[32] George Robertson, Mary Czerwinski, Patrick Baudisch, Brian Meyers, Daniel Robbins, Greg Smith, Desney Tan (2005) The Large-Display User Experience. IEEE Computer Graphics and Applications, vol. 25, no. 4, pp. 44-51, Jul/Aug, 2005.

[33] Isabelle Guyon, Colin Warwick (1997) Handwriting as computer interface. In Survey of the State of the Art in Human Language Technology. Cambridge University Press.

[34] Stefan Agamanolis (2003) Designing displays for Human Connectedness (book chapter), in Kenton O'Hara, Mark Perry, Elizabeth Churchill, and Daniel Russell (eds), Public and Situated Displays: Social and interactional aspects of shared display technologies, Kluwer.

[35] Harry Brignull, Yvonne Rogers (2003) Enticing People to Interact with Large Public Displays in Public Spaces. In Proceedings of INTERACT'03, 17-24.

[36] SMART Technologies. World Wide Web. http://www.smarttech.com

[37] Flatland: An Electric Whiteboard System. World Wide Web. http://www.mtl.t.u-tokyo.ac.jp/~takeo/research/flatland/flatland.html

[38] Contextual Animation of Gestural Commands. World Wide Web. http://groucho.siggraph.org.mx/buxton/Animcontex.htm

[39] Dynamo. A Communal, Multi-User Surface for Sharing & Exchanging Digital Media. World Wide Web. http://www.ux-design.net/dynamo-interactive.com

[40] Wikipedia: Graffiti (Palm OS). World Wide Web. http://fr.wikipedia.org/wiki/Graffiti_(Palm_OS)

[41] Mozdev. Mouse Gestures. World Wide Web. http://optimoz.mozdev.org/gestures/

10 Lijst figuren

11 Lijst tabellen

Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen, de gevraagde informatie in te vullen (en de overeenkomst te ondertekenen en af te geven).

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Brainstorming via interactive whiteboards
Richting: **Master in de informatica** Jaar: **2007**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Ik ga akkoord,

Bob Dijck

Datum: **19.08.2007**