



Fakulteten för ekonomi, kommunikation och IT
Avdelningen för Datavetenskap

Mathias Andersson
Henrik Bäck

Flerpunktspekskärm med användargränssnitt för multipla operationer

Examensarbete 30 högskolepoäng
Civilingenjörsprogrammet IT

Datum/Termin: XX-XX-XX
Handledare: Katarina Asplund
Examinator: Donald F. Ross
Ev. löpnummer: X-XX XX XX

Denna uppsats är skriven som en del av det arbete som krävs för att erhålla en civilingenjörsexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Mathias Andersson

Henrik Bäck

Godkänd, Date of defense

Opponent: 1

Opponent: 2

Handledare: Katarina Asplund

Examinator: Donald F. Ross

Sammanfattning

Flerpunktspeksskärmar¹ har under de senaste åren blivit populära inom hemelektronik. Den här uppsatsen fokuserar på problemen kring att skapa en applikation med användargränssnitt för flerpunktspeksskärmar. Uppsatsen visar också hur man kan skapa en flerpunktspeksskärm för under 3000 SEK av komponenter som kan köpas i detaljhandeln.

¹Engelska: Multi-Touch Screens

Multi-touch screen with interface for multiple operations

Put the text of your english abstract here

Tack

Ett stort tack till Magnus Andersson för tillverkningen av det chassi som all hårdvara monterats i.

Ett stort tack till Katarina Asplund vid Karlstads universitet för all den tid hon har lagt ned på detta arbete.

Tack till Peter Rönnlund vid Karlstads universitet som bjöd på fika i en trött stund.

Innehåll

1	Inledning	1
1.1	Problem	1
1.2	Motivering	2
1.3	Mål	2
1.4	Genomförande	3
1.5	Primära resultat	4
1.6	Denna uppsats disposition	4
2	Bakgrund	7
2.1	Introduktion till flerpunktspeksskärmar	7
2.1.1	Tillämpningar	8
2.2	Historia	9
2.2.1	En-punktsinmatning	9
2.2.2	Fler-punktsinmatning	11
2.3	Pekskämar och flerpunktspeksskärmar idag	14
2.4	Sammanfattning	15
3	Potentiella lösningar	17
3.1	Hårdvara	17
3.1.1	Pekpunktsregistrering	17

3.1.2	Bildåtergivning	21
3.2	Mjukvara	23
3.2.1	Optisk pekpunktsinformation	24
3.2.2	Applikation	25
3.2.3	Spårare och applikationskommunikation	27
3.3	Begränsningar och krav	28
3.3.1	Hårdvara	28
3.3.2	Mjukvara	28
4	Hårdvaruprototyp	29
4.1	Pekpunktsregistrering	29
4.1.1	Kamera	29
4.1.2	Beräkningar	32
4.1.3	Bildinhämtning	36
4.2	Pekning	36
4.2.1	Ljuskälla	36
4.2.2	Problem	37
4.2.3	Infraröd ljuspenna	38
4.3	Bildåtergivning	39
4.3.1	Val av lösning	39
4.4	Chassi	42
4.4.1	Mått	42
4.4.2	Intern uppbyggnad	42
4.4.3	Anslutningar	44
4.4.4	Start och avstängning	45
4.5	Sammanfattning	45
5	Mjukvaruprototyp	47

5.1	Översikt	47
5.2	Spårare	47
5.2.1	Funktionalitet	48
5.3	Spårare och applikationskommunikation	50
5.3.1	Om TUIO	51
5.4	Applikation	53
5.4.1	Uvecklingsmiljö	53
5.4.2	Implementation	54
5.5	Sammanfattning	69
6	Utvärdering av prototyp	71
6.1	Utvärdering av krav	71
6.1.1	Hårdvara	71
6.1.2	Mjukvara	73
6.2	Iakttagelser	75
6.2.1	Fördröjningar i interaktionen	75
6.2.2	Grafikproblem	77
6.3	Sammanfattning	77
7	Slutsats	81
7.1	Detta projekt	81
7.1.1	Hårdvara	81
7.1.2	Mjukvara	82
7.1.3	Summering	82
7.2	Framtida vidareutveckling	82
7.2.1	Användargränssnittsanpassning	82
7.3	Lärdomar	84
	Litteraturförteckning	87

A	Figurreferenser	91
B	Ritningar för chassi	93

Figurer

1.1	Schematisk översikt över systemets uppbyggnad	3
2.1	Historisk tidslinje för alternativa inmatningssystem	9
2.2	The Rand Tablet	10
2.3	PLATO 1978	12
2.4	reactTable*	14
2.5	iPhone 3G	15
2.6	Microsoft Surface	16
3.1	Utspridd upplysning	19
3.2	Frustrerad total intern reflektion	20
3.3	Lysdiodspenna	21
4.1	Hårdvaran	30
4.2	XBOX Live Vision	31
4.3	Kamerans synfält	33
4.4	Kamerans synfält utökas genom en spegel	35
4.5	Utspridd belysning uppmonterad	37
4.6	Pekpennas utseende och uppbyggnad	39
4.7	Plattskärmens bildpanel	40
4.8	Lager för att bland annat ge vit färg	41
4.9	Plattskärm monterad i chassi. På bilden syns undersidan av skärmen.	41

4.10	Hårdvarans uppbyggnad internt	43
4.11	Kameran är monterad ovan plattskärmens kretskort	44
4.12	T.v intag till chassit och t.h. intern koppling	45
5.1	Översikt över mjukvaruprototyp	48
5.2	Bildsegmentering med tre sammanhängande områden	50
5.3	TUIO, Meddelandetyper	52
5.4	TUIO, Profiler	53
5.5	Mjukvaruprototypens delar	55
5.6	Rotation kring ankarpunkt	64
5.7	Nytt koordinatsystem vid ankarpunkten	65
5.8	Skalning av lager	66
6.1	Bildåtergivningen fungerar	72
6.2	En ögonblicksbild av gränssnittet	74
6.3	Rotation och skalning	75
6.4	Skärmavbild av pianot	76
7.1	Ett dåligt användargränssnitt för pekning	84

Tabeller

3.1	Olika spårare som implementerar TUIO	24
3.2	Meddelandetyper, internt i applikation	25
4.1	Kameraspecifikationer	31
4.2	Mätvärden för kamerans synfält	33
4.3	Komponenter för konstruktion av infraröda ljuspennor	38
4.4	Plattskärmsspecifikationer	42
4.5	Spegelspecifikation	44
5.1	OSC Argumenttyper	52
5.2	TUIO, Meddelandeargument	54
6.1	Prislista	73

Programkod

5.1	Hantering av meddelanden	57
5.2	Hantering av "set"-meddelande	58
5.3	Hantering av "alive"-meddelande	59
5.4	Kollision- och lagerkommunikation, frågor	60
5.5	Meddelanden för pekpunkter	61
5.6	Skapa nytt koordinatsystem för rotation	66
5.7	Beräkning av ny skalfaktor	67
5.8	Det informella protokollet Touching	68
6.1	Två objekt placeras	78
6.2	LimePianos gränssnitt	79

Kapitel 1

Inledning

Den här uppsatsen kommer att visa hur man kan lösa problemet med att mata in flera simultana pekpunkter i ett system via en och samma in-dataenhet. Uppsatsen kommer att visa på ett sätt att konstruera en in-dataenhet men framför allt visa på hur en mjukvara för ändamålet kan konstrueras.

Som in-dataenhet används en flerpunktspeksskärm¹ vilken har tillverkats utifrån standardkomponenter. Mjukvaran som tas fram och implementeras i denna uppsats har till syfte att vara enkel att byggas ut.

I det här kapitlet ges en inledning till vad denna uppsats kommer att behandla. Först diskuteras problemställningen samt motivation och målet för uppsatsen. Senare tas viktiga delar i genomförandet upp samt de primära resultat som framkom under utvärderingen upp. Sist finns en information över hur denna uppsats är disponerad.

1.1 Problem

Klassisk mus-interaktion ger enbart en pekpunkt åt gången. Pekaren kan flyttas mellan olika objekt och klicka på dessa. Interaktionen med de flesta applikationer blir, på grund av detta, sekventiellt vilket kan vara en begränsning.

I vissa situationer är det önskvärt att kunna ha flera simultana punkter att

¹Engelska: Multi-Touch Screen, ej officiell översättning men korrekt enligt Svenska Datatermsgruppen [5].

arbeta med, detta kan exempelvis ge snabbare interaktion [27].

1.2 Motivering

Om ett system kan detektera flera simultana pekpunkter öppnas nya möjligheter för vilka operationer som kan utföras på systemet [27]. Dessutom öppnar det upp möjlighet för flera personer att samtidigt, beroende eller oberoende av varandra, använda systemet.

Om mjukvaran som systemet använder också är lätt att göra om för olika behov ger detta en stor fördel. Samma grund kan användas till många olika tillämpningar och endast mindre modifikationer behöver göras vilket skulle innebära att mer tid kan ägnas åt design av användargränssnittet än åt tekniken bakom.

1.3 Mål

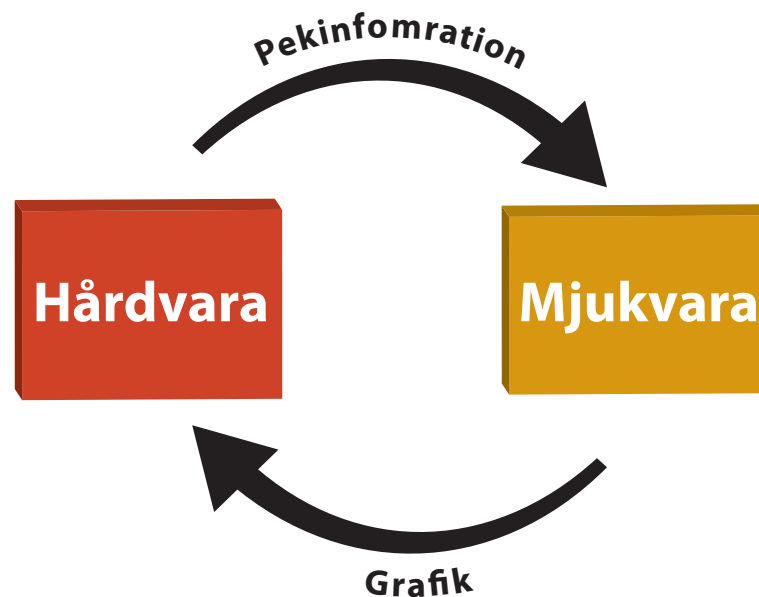
Målet med denna uppsats är att skapa ett komplett system för hantering av flera simultana pekpunkter.

Målet består av två delar, en prototyp för hårdvaran och en för mjukvaran. Hårdvaruenheten skall både kunna återge mjukvarans användargränssnitt och detektera flera simultana pekpunkter. Mjukvaran måste kunna ta hand om flera simultana pekpunkter och utföra olika operationer beroende på pekpunkters rörelse individuellt eller i förhållande till varandra.

Mjukvaran som utvecklas skall dessutom vara lätt att bygga ut för att täcka fler behov.

1.4 Genomförande

Genomförandet av arbetet delades upp i tre delar. Två av delarna bestod i prototypframställning och den sista delen var utvärdering av dessa två. Figur 1.1 föreställer en schematisk bild över hur delarna i systemet samverkar. Till vänster syns hårdvaruprototypen, vars uppgift är att registrera pekpunkter och ge återkoppling i form av grafik. Till höger syns mjukvaruprototypen vars uppgift är att ta emot pekpunktsinformation från hårdvaran samt ge återkoppling på den data som tas emot.



Figur 1.1: Schematisk översikt över systemets uppbyggnad

Hårdvaruprototypen har tagits fram från standardkomponenter. Alla komponenter har kunnat köpas i detaljhandeln och den totala kostnaden för framtagningen av prototypen har kostat under 3000 SEK. Hårdvaruprototypen tar emot pekpunkter via pekpenor som tagits fram enbart för detta projekt.

Mjukvaruprototypen är designad utifrån att olika objekt på skärmen representeras av lager, vilka skall kunna innehålla valfri funktionalitet. Då varje lager

definierar sin egna funktionalitet behöver inte programmets stomme programmeras om för att definiera ny funktionalitet.

1.5 Primära resultat

Hårdvaruprototypen som tagits fram är byggd utifrån komponenter som går att köpa i detaljhandeln och konstruktionen har kostat under 3000 SEK. Hårdvaran levererar information till mjukvaran om var pekpunkter befinner sig på skärmtan.

Mjukvaran som byggts har inbyggt stöd för tre stycken operationer på grafiska objekt. Dessa är rotation, skalning och förflyttning. Det är också möjligt att introducera nya sorters objekt som har stöd för andra operationer utan att skriva om mjukvarans stomme. Mjukvaran har i grunden stöd för att visa text, bilder och video vilka också kan manipuleras med de inbyggda operationerna. Andra typer av media kan också introduceras i och med att införa nya typer av objekt.

1.6 Denna uppsats disposition

Den här uppsatsen består av sju kapitel. Detta kapitel (nummer 1) är till för introduktion av uppsatsen och dess innehåll.

I kapitel 2 introduceras begreppet flerpunktspeksskärmar och en kort historik kring datainmatning ges. Kapitel 3 ger en överblick av vilka delar som måste utredas samt vilka olika lösningar som är potentiellt möjliga. Några av de potentiella lösningar som finns i kapitel 3 väljs ut och deras implementation förklaras i kapitel 4 och 5.

Prototypkonstruktionen har delats upp i två kapitel för att göra arbetet mer överskådligt. Kapitel 4 handlar enbart om den hårdvara som konstruerats och hur denna fungerar. Samma sak gäller för mjukvaran, som finns i kapitel 5.

I kapitel 6 presenteras resultat från utvärderingen av både hårdvaruprototypen och mjukvaruprototypen. Utvärderingen baseras på de krav som finns presenterade i kapitel 3. Slutsatsen för uppsatsen återfinns i kapitel 7.

Kapitel 2

Bakgrund

Flerpunktsinmatning är ingen ny teknik men det är först de senaste åren som den har börjat användas i olika applikationer [34]. Några mobiltelefoner [11] och datorer utrustas idag med flerpunktspeksskärmar av olika slag och användningsområdena varierar. I det här kapitlet ges först introduktion till begreppet flerpunktspeksskärmar. Efter detta följer en kort historik över olika inmatningsmetoder, relaterade till peksskärmar eller flerpunktspeksskärmar. Sist i kapitlet finns information om hur peksskärmar och flerpunktspeksskärmar används idag samt en sammanfattning för det här kapitlet.

2.1 Introduktion till flerpunktspeksskärmar

Till skillnad från peksskärmar har flerpunktspeksskärmar möjlighet att ta emot flera simultana pekpunkter. Detta öppnar nya möjligheter för vilka olika sorters operationer som kan utföras på systemet [27]. Peksskärmar har också fördelen att de erbjuder direkt manipulation då en användare arbetar med sina händer direkt på skärmen. En annan användare har då lättare att koordinera händerna och synintrycken än vid användande av tangentbord och mus. [33][34]

2.1.1 Tillämpningar

Flerpunktspeksskärmar kan ha flera potentiella användningsområden. De lämpar sig exempelvis bra som interaktiva whiteboards då de har möjlighet att ta emot inmatningar från flera samtidiga användare. Detta gör att flera personer samtidigt kan använda tavlan. En interaktiv whiteboard ger dessutom möjlighet att kunna lagra den information som matats in och på så sätt göra informationen på tavlan bestående.

Från whiteboard är steget inte alls långt till anslagstavlor. Det skulle kunna gå att tillverka stora anslagstavlor där flera personer samtidigt kan läsa och sätta upp digitala anslag. Anslagstavlor skulle kunna användas på företag, skolor eller i köpcentrum. Sammankopplade anslagstavlor ger användarna en möjlighet att snabbt publicera sitt anslag på flera fysiska platser.

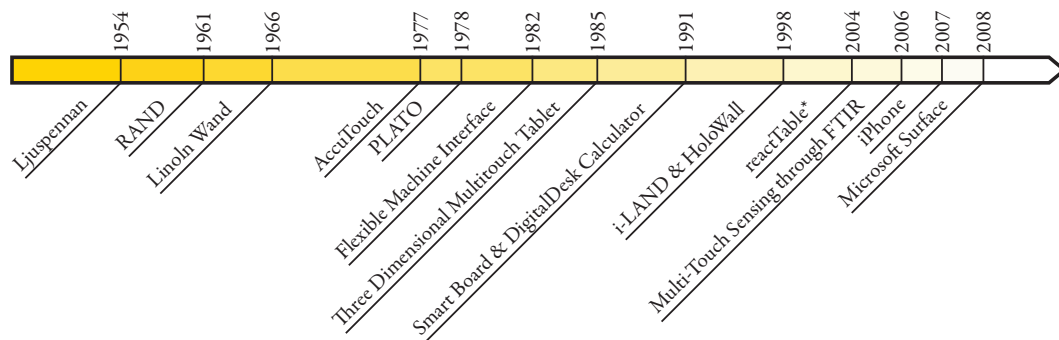
I presentationssammanhang finns flera andra tänkbara användningsområden. I nyhetssändningar analyseras olika scenarion i direktända program och exempelvis skulle en väderkarta inte behöva vara animerad i förväg för att visa närmare vyer av områden. Presentatören skulle i stället kunna navigera i kartan med bara händerna.

Tekniken lämpar sig också inom design. Det finns en fördel med att kunna arbeta med båda händerna när man gör kurvor. Det mer naturligt att arbeta med två händer och därmed går det också både enklare och fortare att utföra uppgifter [27]. Exempel på en teknik som passar in här är "TapeDrawing", där man använder båda händerna för att forma kurvor med hjälp av tejp. Man slipper då problemet med att konvertera de analoga ritningarna till digitalt senare.

2.2 Historia

Alternativa inmatningsmetoder till tangentbordet har funnits i över 50 år. I detta avsnitt presenteras historiska hållpunkter inom områden som kan relateras till pekskärmar och flerpunktspeksskärmar. Figur 2.1 visar en tidsaxel där de prototyper och produkter som tas upp är utplacerade.

Först i detta avsnitt kommer inmatningssystem relaterade till pekskärmar att presenteras och efter detta presenteras inmatningssystem relaterade till flerpunktspeksskärmar.



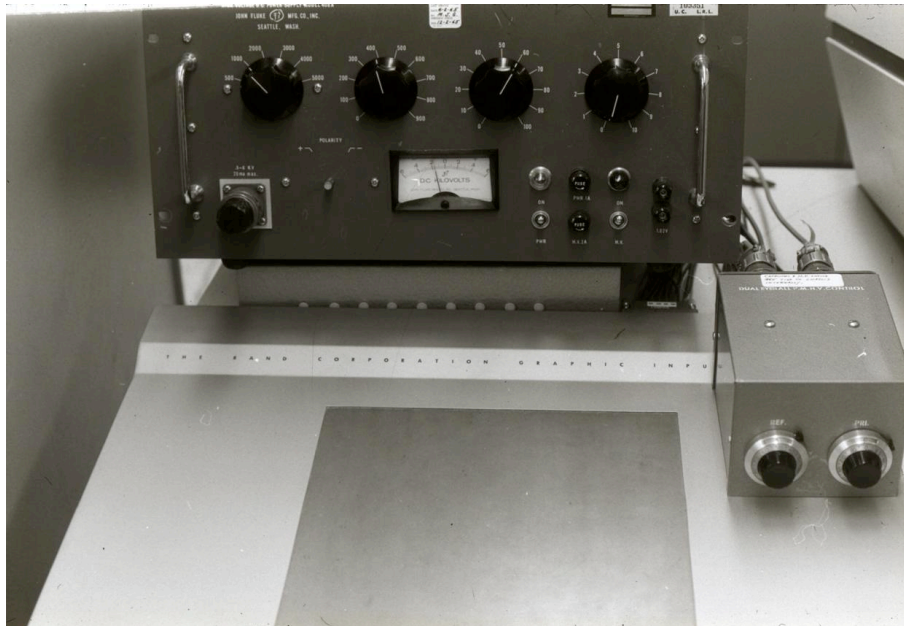
Figur 2.1: Historisk tidslinje för alternativa inmatningssystem

2.2.1 En-punktsinmatning

1954 introducerades ljuspennan som inmatningsmetod [25]. Med ljuspennan arbetade användaren direkt på skärmen med en för ändamålet utvecklad penna. Pennan hade en inbyggd ljussensor som kände av när CRT-skärmens katodstråle träffade densamma. Katodstrålens position är känd och utifrån denna information kan man härleda pennans position i två dimensioner.

THE RAND TABLET (se figur 2.2) lanserades 1961 [4] och hade en 10 gånger 10 tum stor yta vilken kunde läsa diskreta punkter i planet. Ytan bestod av en elektriskt laddad platta. På ytan användes en speciell penna (sond) för att göra

inmatningar. Pennan hade en tryckkänslig spets för att simulera en verklig penna och på så sätt mata in data i tre dimensioner. Då inmatning och utmatning var fysiskt separerade kan THE RAND TABLET främst liknas vid dagens ritbord¹ [6].



Figur 2.2: The Rand Tablet

1966, fem år efter RAND, utvecklades en in-dataenhet kallad THE LINCOLN WAND [20], där användaren kunde peka direkt på skärmen. Fortfarande användes en speciell penna för att göra kopplingen mellan pekpunkten på skärmen och datorn. Pennan var utrustad med en ultraljudsmottagare och dess position gick att utläsa i tre dimensioner, till skillnad från ljuspennans två dimensioner. Tekniken fungerar, i kort, så att skärmen har fyra stycken inbyggda ultraljudssändare som periodiskt sänder ut en ultraljudssignal. När användarens penna tar emot ljuden från sändarna kan avståndet beräknas till varje sändare. På så sätt kunde positionen för pennan bestämmas.

¹Engelska: Tablet

Elva år senare, 1977, tillverkades en transparent yta kallad AccuTouchTM[16] som kunde avläsa en punkt baserat på fingrets placering på skärmen. Systemet består av två lager av elektriskt ledande material. När ett finger trycker på ytan sluts en elektrisk krets mellan de två lagren. Fingrets position beräknas genom att mäta potentialen, horisontellt och vertikalt, i punkten [8].

Året efter AccuTouch, 1978, användes en pekskärm i en utbildningsprodukt vid namn PLATO [9] (se figur 2.3). PLATO hade en plasmaskärm om 512×512 bildpunkter² som dessutom var utrustad med möjligheten att ta emot data genom att placera ett finger på skärmen. En användare kunde på så sätt göra val i olika applikationer utan att använda tangentbordet. Pekskrmen hade en upplösning på 16×16 punkter vilket innebar att det var möjligt att känna av 256 olika områden på skärmen. Runt skärmen placerades infraröda ljuskällor och mottagare. Dessa skapade ett rutnät av ljusstrålar över skärmen. När ett finger sattes på skärmen bröts två ljusstrålar och detta gav fingrets position som vidarebefordrades till applikationen [2].

Från 1980-talet och framåt började pekskrmar leta sig in i allt fler kommersiella produkter [31]. Hewlett-Packard lanserade kommersiella arbetsstationer för personligt bruk med inbyggd pekskärm. Dessa byggde på samma teknik som den i PLATO.

Under nittiotalet, närmare bestämt 1991, presenterades Smart BOARDTM. Smart BOARD, är en interaktiv whiteborard som är menad att användas i utbildningssammanhang.

2.2.2 Fler-punktsinmatning

Redan 1982 togs en prototyp fram för att kunna känna av flera simultana inmatningspunkter från en eller flera användare. Prototypen fick namnet The Flexible

²Engelska: Pixels



Figur 2.3: PLATO 1978

Machine Interface [31] och fungerade genom bildinläsning. En yta läses av med en kamera och fingrarna uppträder som svarta punkter som kameran kan avläsa.

Tre år senare, 1985, presenterades en yta med möjlighet att ta emot flera samtidigt punkter och som dessutom talar om graden av kontakt [21]. Ytan, som var indelad i 64 gånger 32 punkter, tillät att användaren arbetade direkt med fingrarna på ytan. Avläsningen skedde genom att ytan periodiskt skannades av med hjälp av en algoritm där punkter med lägre elektrisk laddning upptäcks. Ytan rapporterade plankoordinater samt hur hårt användaren tryckte på punkten. Detta gav en tredimensionell inmatning.

The DigitalDesk Calculator [38] presenterades 1991. Enheten projicerade ett gränssnitt på toppen av ett bord. Bredvid projektorn fanns även en kamera som

registrerade användarens rörelser över bordet och dessutom objekt på bordet. Systemet kunde då ta emot flera samtidiga inmatningar och känna igen dokument placerade på bordet.

Sju år senare, 1998, presenterades projektet i-LAND [36] och som en del av projektet togs två enheter fram, DynaWall och InteracTable. DynaWall var en stor interaktiv whiteboard som hade möjligheten att ta emot information från flera samtidiga användare och InteracTable var ett bord runt vilket upp till sex personer kunde stå. Bordet kunde ta emot in-data från flera samtidiga användare.

Samma år presenterades även HoloWall [23]. HoloWall var en vägg designad för att kunna ta emot fler samtidiga inmatningar men även känna igen föremål som hölls upp framför ytan. HoloWall fungerade genom att en yta projekteras bakifrån, från samma håll belyses även ytan med ett jämt infrarött ljus. Vid projektorn fanns en kamera som filmade ytan och det infraröda ljuset som belös ytan. Kameran kunde registrera det infraröda ljuset som reflekteras och på så sätt räkna ut pekpunkter.

Under tjugohundratalet, närmare bestämt 2004, presenterades en fungerande prototyp av reactTable*[19] (se figur 2.4). reactTable* var ett interaktivt musikinstrument som hade stöd för att kunna känna igen flera fysiska objekt placerade på ytan samtidigt. Objekt identifierades optiskt med hjälp av en kamera och märkningar på varje objekt. Utifrån objekten, dess riktning och deras inbördes förhållande genererades musik [17].

De tre åren 2006, 2007 och 2008 innebar stora språng på marknaden för flerpunktspekskärmar. 2006 presenterade Jefferson Y. Han en flerpekpunktsyta som kunde konstrueras både enkelt och icke kostnadskrävande [10]. Han presenterade sina idéer, om en billig flerpunktspekskärm, på konferensen TED (Technology, Entertainment, Design)[37] vilket fick världen att öppna ögonen för vad som kan åstadkommas.



Figur 2.5: iPhone 3G

flerpunktspeksskärmar.

När det gäller flerpunktspeksskärmar levererar ett företag vid namn N-trig ytor för detta. N-trig levererar ytor till bland annat Dell och Intel [26] som har implementerat dessa i sina datorer.

Tekniken som kommer att användas för att ta fram en hårdvaruenhet i denna uppsats är liknande den som återfinns i Microsoft Surface.

2.4 Sammanfattning

Flerpunktspeksskärmar ger fler fördelar än en-punktspeksskärmar. Bland annat låter de användaren använda flera händer och fingrar när denne utforskar ett problem och dessutom låter de flera personer samtidigt interagera med ett system. Flerpunktspeksskärmar passar inte i alla situationer men kan dock med fördel an-



Figur 2.6: Microsoft Surface

vändas i flera specifika tillämpningar.

Pekskärmar är ingen nyhet på marknaden och så inte heller flerpunktspek-skärmar. Det har funnits varianter av pekskärmar sedan 1954 och redan från början har de använts för att mata in data till olika applikationer. Det har också i cirka 20 år funnits pekskärmar som kan ta emot flertalet in-datapunkter trots att det endast är på senare år de har börjat hitta in i olika kommersiella produkter.

Kapitel 3

Potentiella lösningar

I det här kapitlet kommer olika lösningsförslag ges till problemet med att skapa hård- och mjukvara för flerpunktspeksskärmar. Först presenteras vägar att konstruera hårdvaran med aspekter i metoder för både bildåtergivning och pekpunktsregistrering. Senare i kapitlet diskuteras olika lösningsförslag för mjukvaran och hur denna kan bli expanderbar.

3.1 Hårdvara

3.1.1 Pekpunktsregistrering

För att flerpunktspeksskärmen skall kunna användas måste samtliga pekpunkter som befinner sig på skärmen registreras av systemet. Det finns ett flertal olika sätt att genomföra denna registrering på.

3.1.1.1 Kapacitiv pekpunktsregistrering

Genom historien har flera olika produkter använt sig av kapacitiv avläsning av pekpunkter [21] [16]. Ett inköp eller utvecklande av en sådan metod kan rimligtvis inte ligga innanför varken budget eller tidsrymd för detta arbete varpå detta alternativ inte kommer att behandlas vidare.

3.1.1.2 Optisk pekpunktsregistrering

Ett annat sätt att läsa av aktiva pekpunkter på skärmen är genom optisk avläsning.

En kamera monteras så att den filmar ytan som skall avläsas. Avläsning av ytan kan lämpligen ske på motsatt sida användaren för att användarens händer och armar då inte är i vägen för avläsningen. Det enda som då syns för kameran är de punkter som är nära skärmens yta eller i kontakt med ytan.

För att kameran tydligt skall kunna se pekpunkter från motsatt sida användaren måste dessa framhävas, exempelvis genom belysning. För att inte störa skärmens bild bör ett, för ögat, osynligt ljus användas för detta ändamål [23] [30]. Ett exempel på osynligt ljus är infrarött ljus, vilket återfinns i exempelvis fjärrkontroller till hemelektronik.

Hur upplysningen av pekpunkterna sker kan varieras och flera metoder presenteras nedan.

Utspridd upplysning

Ett sätt att lysa upp punkterna är med en teknik som kallad utspridd upplysning¹. Ytan som skall registreras belyses underifrån med infrarött ljus [23]. När ett finger placeras på ytan kommer detta att reflektera ljuset nedåt så att detta kan registreras av kameran (se figur 3.1).

Frustrerad total intern reflektion

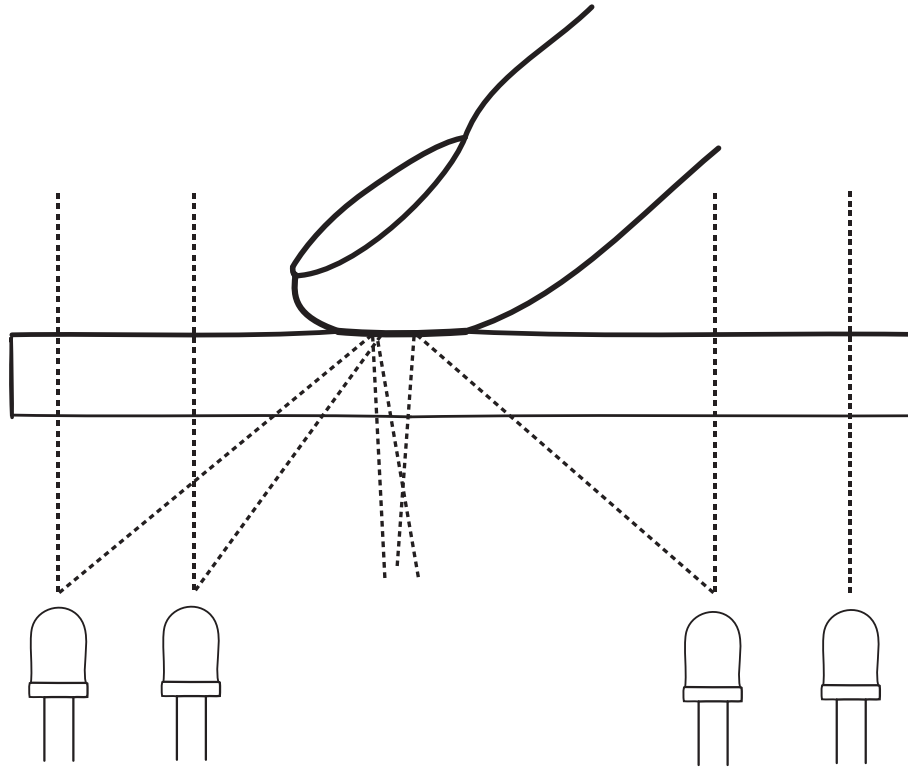
Ett annat sätt att belysa pekpunkter är via så kallas frustrerad total intern reflektion² (FTIR) [10].

Tekniken går ut på att man sänder in infrarött ljus i en plexiglasskiva³. Ljuset sänds in med en vinkel lika med eller större än den så kallade *kritiska vinkeln* för

¹Engelska: Diffused Illumination

²Engelska: Frustrated Total Internal Reflection

³Polymetylmetakrylat



Figur 3.1: Utspridd upplysning

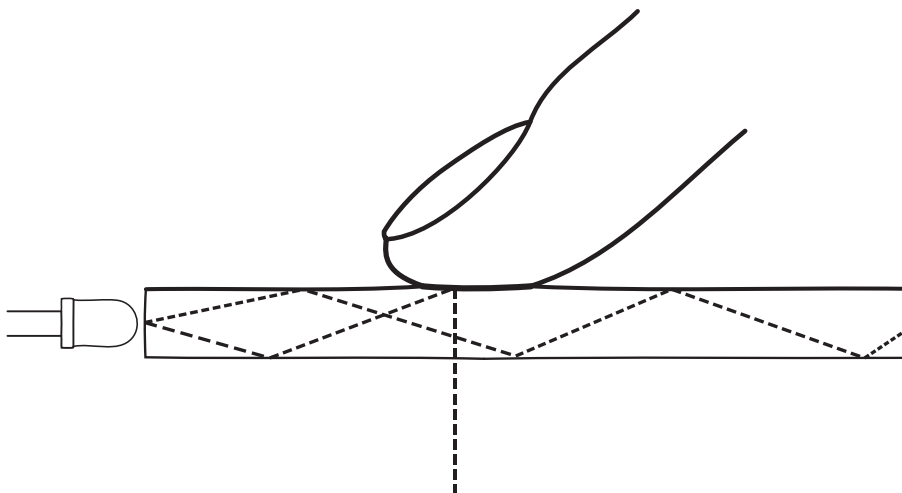
total reflektion. När ljuset försöker bryta sig ut från plexiglasskivan kommer det att träffa på luft som har ett lägre optiskt brytningsindex än plexiglas. Detta gör att allt ljus kommer att reflekteras och stanna kvar inuti plexiglasskivan.

Den *kritiska vinkeln*, θ_c , då total reflektion uppstår kan härledas med hjälp av Snells lag [28] (se ekvation 3.1). Total reflektion uppstår endast när refraktionsvinkeln är större än 90 grader. I ekvationen är n_1 brytningsindex för plexiglasskivan och n_2 brytningsindex för luft.

$$n_2 \sin \theta_c = n_1 \sin 90^\circ \Leftrightarrow \theta_c = \arcsin \left(\frac{n_1}{n_2} \right) \quad (3.1)$$

När ett finger sätts mot plexiglasskivans yta medför det en förändring av den kritiska vinkeln vid det aktuella området. Detta innebär att ljuset kommer att

lämna plexiglasets och reflekteras mot fingret (se figur 3.2)



Figur 3.2: Frustrerad total intern reflektion

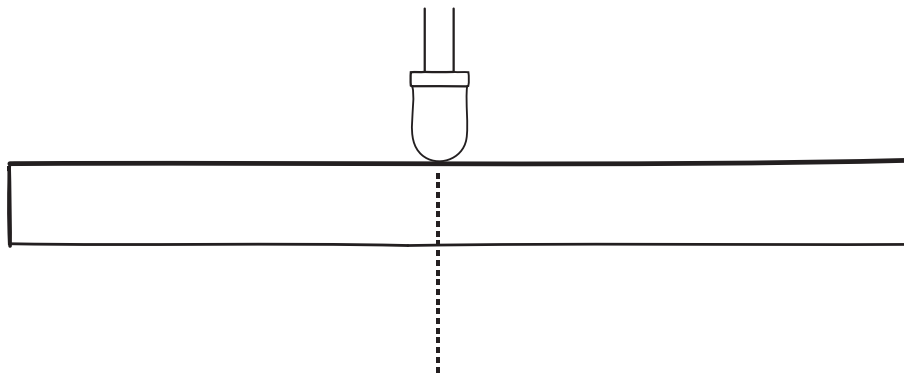
Penna

Eftersom pekpunkter registreras med hjälp av en kamera är det möjligt att belysa kameran direkt med en infraröd ljuspunkt, istället för indirekt via reflektion. Detta är en säkrare metod eftersom ljuset träffar kameran med mer kraft. Metoden går också att kombinera med de övriga metoderna för optisk avläsning utan att göra modifikationer i hårdvaran.

Genom att bygga in en infraröd lysdiod i en penna kan pennan användas för att peka på skärmen. Ljuset från pennans infraröda lysdiod kommer att uppfattas som en pekpunkt av kameran.

3.1.1.3 Sensorbaserad pekpunktsregistrering

Den sensorbaserade metoden går ut på att plattskärmen har inbyggda sensorer för att känna av ljus. När ett finger placeras på skärmen registreras detta av TFT-fotoceller inbyggda skärmens bildpunkter [7].



Figur 3.3: Lysdiodspenna

Denna metod kräver antingen att en hårdvara med tekniken köps in eller kan konstrueras. Detta alternativ kommer, på grund av dess komplexitet, inte att vidare behandlas i denna uppsats.

3.1.1.4 Val av lösning

Hårdvarulösningen har valts utifrån två aspekter. En stor aspekt är priset och det andra är tiden för att utföra jobbet. Båda parametrarna är ytterst begränsade i detta projekt.

Den lösning som valts baseras på optisk pekpunktsregistrering med hjälp av en kamera.

3.1.2 Bildåtergivning

För att flerpunktspeksskärmen skall fungera måste ut-data, i form av grafik, presenteras på den yta som registrerar pekpunkter. Det finns flera sätt att lösa detta på och det går att kombinera olika sorters bildåtergivningsmetoder med pekpunktsregistrering.

För det är arbetet finns två tänkbara alternativ och detta är digitalprojektorn⁴

⁴Engelska: Video projector

och LCD-skärmen⁵.

3.1.2.1 Digitalprojektor

Ett möjligt bildåtergivningssätt är att använda sig av en digitalprojektor. En digitalprojektor monteras så att den projicerar en bild på en duk. Digitalprojektorn kan monteras i sitt ursprungliga skick men kan behöva kompletteras med exempelvis speglar då avståndet för projicering kan bli kort. [10] Vissa digitalprojektorer kan också behöva kompletteras med ett filter för att ta bort oönskat infrarött ljus från projektorns lampa [23] vilket annars skulle störa optisk pekpunktsregistrering, se avsnitt 3.1.1.2.

Fördelar

- Ger bättre bildkvalité än LCD-skärm, då LCD-skärmen måste modifieras
- Ytan som projiceras kan vara mindre IR-ljushindrade

Nackdelar

- Dyrare än LCD-skärm i inköp
- Värmeutvecklande, extra ventilation kan krävas
- Kan kräva långt projiceringsavstånd

3.1.2.2 Plattskärm

Ett alternativ till digitalprojektor är LCD-skärmar, som i mindre skala är billigare. Ett problem med LCD-skärmar, vidare kallat plattskärmar, är att de kan behöva modifieras för att användas tillsammans med olika typer av pekpunktsregistrering.

⁵Med eller utan TFT

Då optisk pekpunktsregistrering (se avsnitt 3.1.1.2) används måste skärmen kunna släppa genom ljus. Plattskärmen måste monteras isär så att bakstycket kan avlägsnas och kretskort flyttas för att göra det möjligt att sända ljus rakt genom skärmen[24] (se avsnitt 3.1.1.2). Skärmen kan innehålla optiska filter vilka kan behöva avlägsnas för att kunna registrera pekpunkter korrekt.

Fördelar

- Billigare än digitalprojektor
- Oftast högre upplösning än digitalprojektor

Nackdelar

- Plattskärmens konstruktion kan förhindra att optisk pekpunktsregistrering används
- Kräver modifiering

3.1.2.3 Val av lösning

För bildåtergivning har en plattskärm valts eftersom det är det billigaste alternativet, i valet mellan videoprojektor och plattskärm och ekonomi är i detta fall styrande faktor.

3.2 Mjukvara

Mjukvaran är en stor del av den pekpunktsbaserade inmatning. Mjukvarudelen består av ett grafiskt användargränssnitt, en spåraren⁶ samt logik som tolkar uppgifter från spåraren.

⁶Engelska: Tracker

3.2.1 Optisk pekpunktsinformation

När man använder optisk avläsning är den data man erhåller en sekvens av stillbilder. För tillämpningen är de erhållna bilderna oanvändbara i sitt ursprungliga format. Användbar information utläses ur bilden genom att behandla och tolka bilden, vilken är spårarens uppgift. Spårarens ska åtminstone lämna i från sig information om koordinater för pekpunkter.

3.2.1.1 Olika spårare

Det finns flera spårare som kan användas. Spårarna implementerar olika sätt att lämna i från sig data men gemensamt för de som studerats är att de implementerar TUIO (se avsnitt 3.2.3.2). I tabell 3.1 finns exempel på ett antal spårare som implementerar TUIO.

<i>Spårare</i>
BBTouch
Touché
TouchLib
Reactivision

Tabell 3.1: Olika spårare som implementerar TUIO

3.2.1.2 Val av lösning

Då optisk pekpunktsregistrering används har spåraren *Touché* valts. Dock kan vilken spårare som helst, som implementerar TUIO, användas. Touché har valts på grund av att den är öppen källkod samt fungerar på Apple MacOS® X som även applikationen skall köras under (se avsnitt 3.2.2.3).

3.2.2 Applikation

Applikationens uppgift är att ta emot pekpunktsinformation och tillämpa informationen. Applikationen ritar upp gränssnittet på skärmen och innehåller logik för att reagera på pekpunktsinformationen.

3.2.2.1 Grafisk uppbyggnad

En möjlig lösning på den grafiska uppbyggnaden är att ett antal lager ritas på skärmen. Varje lager kan exempelvis innehålla information om vad som skall ritas, dess position samt metoder för att tala om för lagret om omgivande händelser.

På detta sätt kan ett lager skapas för varje grafiskt objekt som skall ritas, vilket är bra ur ett expansions- och separeringsperspektiv.

3.2.2.2 Datahantering

Applikationen tar emot pekpunkter från hårdvaran och översätter dessa till tre stycken olika meddelande-typer (se tabell 3.2). Meddelandetyperna används oavsett vilket protokoll eller API som använts mot hårdvaran. Dessa tre meddelandetyper räcker fullgott för att utföra operationer på objekt i planet eftersom de innehåller information om var en punkt skapas, flyttas och tas bort.

<i>Typ</i>	<i>Argument</i>
Aktivera pekpunkt	Pekpunktens identitet, koordinat
Flytta pekpunkt	Pekpunktens identitet, koordinat
Avaktivera en pekpunkt	Pekpunktens identitet

Tabell 3.2: Meddelandetyper, internt i applikation

Ett *Aktivera pekpunkt*-meddelande kommer att orsaka en kollisionsdetektering mellan pekpunktens koordinat samt de olika lager som finns i systemet. Om en kollision mellan pekpunkten och lagret uppstår kan meddelandet skickas vidare till detta lager.

När det gäller meddelandena *Flytta pekpunkt* samt *Avaktivera pekpunkt* så vet systemet redan vilket lager som är associerat med den aktuella pekpunktsidentiteten. Meddelanden skickas då vidare direkt till detta lager.

Det finns en stor fördel med att låta varje lager ta hand om inkomna meddelanden. Om en generell funktion skall ta emot meddelanden, detektera operationen och sedan tala om för lagret vad det skall utföras begränsas applikationen till dessa förbestämda operationstyper. Om lagret själv i stället hanterar meddelandena kan nya lager-typer introduceras utan att omgivande struktur måste modifieras.

Exempel

Låt ett lager ta emot meddelandet *Rotera X grader*. För att detta skall fungera måste applikationen först ta emot någon av de tre meddelandetyperna i tabell 3.2. Efter detta skall en beräkning utföras för att räkna ut hur många grader som lagret skall roteras och slutligen skall meddelandet *Rotera X grader* skickas till lagret.

Introducera sedan ett nytt lager som inte skall kunna roteras men spela upp ett ljud. Ett sådant lager skulle kunna ta emot meddelandena *Spela* och *Stoppa*. Nu måste en del av applikationen skrivas om för att kunna veta vilka lager som svarar på vilka meddelanden samt lägga till kunskap för att veta om ett *Spela*- eller *Stoppa*-meddelande skall skickas.

Om all denna funktionalitet istället finns i lagret behöver applikationen enbart ta reda på vilket lager som skall ha ett visst meddelande och därefter vidarebefordra detsamma. Varje lager kan själv bestämma vad som skall hända när meddelandet tas emot.

3.2.2.3 Val av programspråk och plattform

Mjukvaruprototypen skrivs i Objective-C® för Apple MacOS X. Anledningen till detta val är att det finns bibliotek med lämplig funktionalitet och dessutom att var

en plattform som fanns tillgänglig.

3.2.3 Spårare och applikationskommunikation

Applikationen måste få information om pekpunkter från spåraren. Hur spåraren tillhandahåller applikationen denna information varierar beroende på implementation.

3.2.3.1 API

Ett sätt att låta applikationen kommunicera med spåraren är via ett API. APIets utformning varierar mellan olika spårare och måste därför anpassas efter vilken spårare som används. Detta innebär att applikationen måste skrivas om för varje spårare som skall användas.

3.2.3.2 TUIO

Ett annat kommunikationssätt är att använda sig av ett protokoll. TUIO [18] är ett protokoll framtaget specifikt i syfte att förmedla information om optiskt identifierade objekt. TUIO arbetar över valfritt transportprotokoll, men UDP används med fördel i ett IP-nätverk. Eftersom TUIO är ett protokoll är det enkelt att byta mellan spårare som implementerar TUIO helt utan att applikationen måste skrivas om.

Exempel på spårare som implementerar TUIO finns i tabell 3.1.

3.2.3.3 Val av lösning

Spårare och applikationskommunikation via TUIO har valts på grund av dess fördelar mot ett API-baserat. TUIO möjliggör ett byta av spårare utan att applikationen behöver skrivas om.

3.3 Begränsningar och krav

3.3.1 Hårdvara

Hårdvarans uppgift är att visa det grafiska gränssnittet. Hårdvaran skall kunna visa en fullt grafisk bild från en videokälla.

Hårdvaran skall också ge ifrån sig en stillbildsserie över pekytan. Pekpunkterna skall vara tillräckligt framträdande i bilderna så att spåraren kan detektera dessa.

Hårdvaran får inte kosta över 4000 SEK i tillverkning.

3.3.2 Mjukvara

Mjukvaran skall kunna visa minst två objekt samtidigt. Ett objekt avser i det här fallet en bitmappsbild. Med hjälp av pekpunkter skall operationer kunna utföras på objekten. I det här fallet är operationerna rotation, proportionell skalning och förflyttning.

Mjukvaran skall vara designad på ett sådant sätt att det underlättar introduktion av nya objekt och operationer.

Kapitel 4

Hårdvaruprototyp

En prototyp (se figur 4.1) har tagits fram för detta projekt enligt de val som gjorts i kapitel 3. Hårdvaruprototypen kommer att användas för att testa mjukvaruprototypen. I det här kapitlet kommer utförliga beskrivningar att ges över hur hårdvaruprototypen är konstruerad. Först kommer konstruktionen för pekpunktsregistrering att tas upp och därefter metoden för att mata in pekpunkter. Efter detta ges förklaringar kring bildåtergivningen och sist i kapitlet tas chassits konstruktion upp samt en sammanfattning över detta kapitlet ges.

4.1 Pekpunktsregistrering

Denna lösning baseras på optisk pekpunktsregistrering med hjälp av en kamera (se avsnitt 3.1.1.2).

4.1.1 Kamera

För att läsa in pekpunkter optiskt används en kamera. En webbkamera har inhandlats för ändamålet och specifikationerna kan ses i tabell 4.1. Man kan med fördel använda en kamera av bättre kvalitet för att få snabbare överföring av bilder samt bättre kvalitet på bilderna. Detta kan ge bättre resultat eftersom ursprungsbilderna då är av en högre kvalitet. Kameran (se figur 4.2) i detta projekt är konstruerad för 640×480 bildpunkters upplösning i 30 bilder per sekund.



Figur 4.1: Hårdvaran

4.1.1.1 Modifiering för infrarött ljus

För pekning används infrarött ljus (se avsnitt 4.2.1). Detta gör att kameran med fördel kan modifieras för att enbart registrera infrarött ljus, vilket i sin tur innebär att inget annat ljus kommer att störa kameran.

Filter

Ett filter, som enbart släpper igenom infrarött ljus, har monterats in i kameran. Bäst resultat fås genom att använda ett filter som enbart släpper genom ljus av den våglängd som används i ljuskällan (se avsnitt 4.2.1). Ett sådant filter är dock

<i>Tillverkare</i>	<i>Modell</i>	<i>Inköpsställe</i>
Microsoft	Xbox LIVE Vision	GameStop Sweden AB

Tabell 4.1: Kameraspecifikationer



Figur 4.2: XBOX Live Vision

relativt dyrt och kan kosta över \$100 [22].

I stället används fullt exponerad och framkallat negativ [29]. Kameran monterar isär och filtret placeras framför kamerans bildsensor. Efter testning framkom att *tre lager* av sådant negativ behövdes för att blockera så mycket synligt ljus som möjligt. Fler lager innebar ingen märkbar skillnad för varken synligt eller infrarött ljus.

Bilder som nu tas med kameran kommer enbart att registrera infrarött ljus eller ljus som ligger mycket nära det infraröda ljusspektrumet. Allt annat ljus bör släckas ut av filtret.

4.1.2 Beräkningar

Kameran måste monteras inuti chassit om den skall kunna registrera pekpunkter underifrån, enligt 3.1.1.2. Ett lämpligt avstånd mellan kameran och bildskärmens yta måste bestämmas och detta genom att kamerans synfält har beräknats. Detta avstånd har använts för att beräkna chassits höjd.

4.1.2.1 Kamerans synfält

Det *exakta* avståndet mellan kameran och ytan som registreras är inte intressant, utan det viktiga är att man får ett avstånd så att åtminstone hela ytan registreras. I prototypen är det bra att ha lite utrymme för justeringar i efterhand samt möjligheten att korrigera för fel.

Eftersom bildskärmen (se avsnitt 4.3) är i bredbildaformat¹ och kameran registrerar en nästan kvadratisk yta så är bredden på skärmen det intressanta.

Låt A vara avståndet från kamerans lins till ytan som skall registrera. Låt vidare B vara bredden på det område som kan registreras av kameran vid avståndet A . För att kunna ge chassits höjd är det nu intressant att veta förhållandet mellan A och B (se figur 4.3).

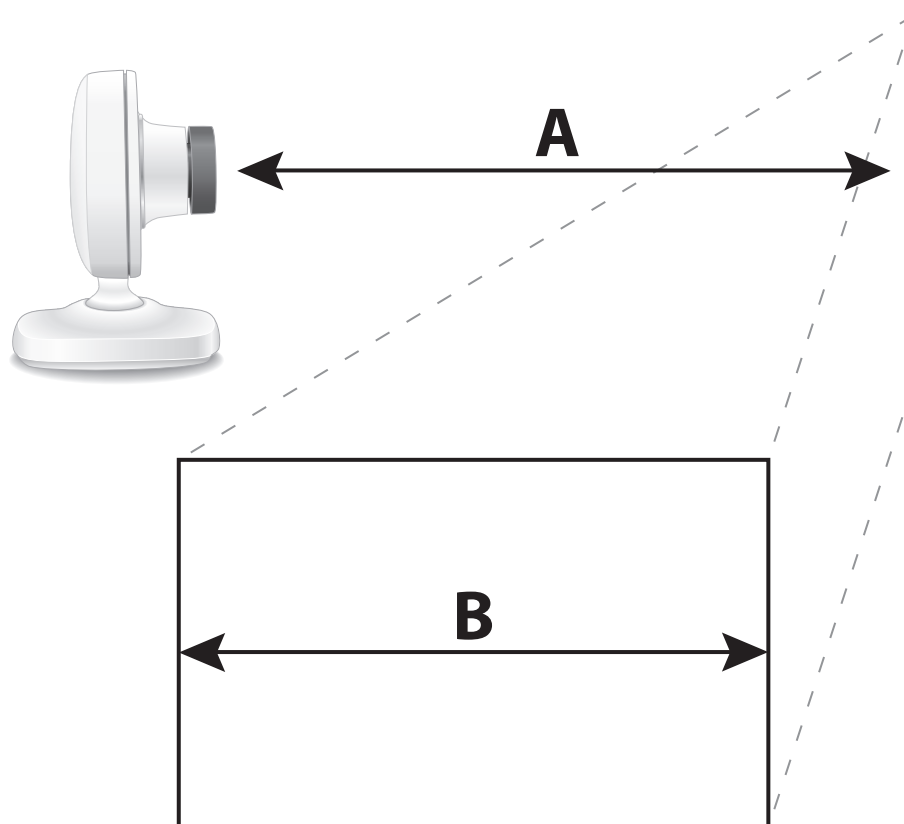
Mätvärden

Tabell 4.2 visar olika mätpunkter samt förhållandet mellan A och B (se figur 4.3).

Som synes i tabell 4.2 kretsar alla värden kring $\frac{4}{3}$. Formel 4.1 erhålls från dessa värden.

$$A = \frac{4}{3}B \quad (4.1)$$

¹Engelska: Wide screen



Figur 4.3: Kamerans synfält

$A \text{ (mm)}$	$B \text{ (mm)}$	A/B
100	75	$4/3$
90	65	$18/13$
80	60	$4/3$
70	53	$70/53$
60	45	$4/3$
50	40	$4/3$
40	30	$4/3$
30	22	$15/11$

Tabell 4.2: Mätvärden för kamerans synfält

4.1.2.2 Chassits höjd

Det är intressant att veta hur långt från ytan som kameran måste sitta för att kunna registrera hela ytan.

Utifrån formel 4.1 kan vi härleda kamerans avstånd, A , eftersom B är känd från skärmens specifikationer.

$$\begin{cases} A = \frac{4}{3}B \\ B = 485 \text{ mm} \end{cases} \Rightarrow A = \frac{485 \cdot 4}{3} = \frac{1940}{3} \approx 647 \text{ mm.}$$

Detta ger att chassits höjd måste som allra minst vara 647 mm högt. Utöver detta tillkommer kamerans egen höjd samt den höjd som en monteringsanordning upptar.

Minska chassits höjd

För att reducera chassits höjd används en spegel. Spegeln reflekterar inkommande ljus från pekpunkter och förlänger på så sätt avståndet mellan ytan och kameran.

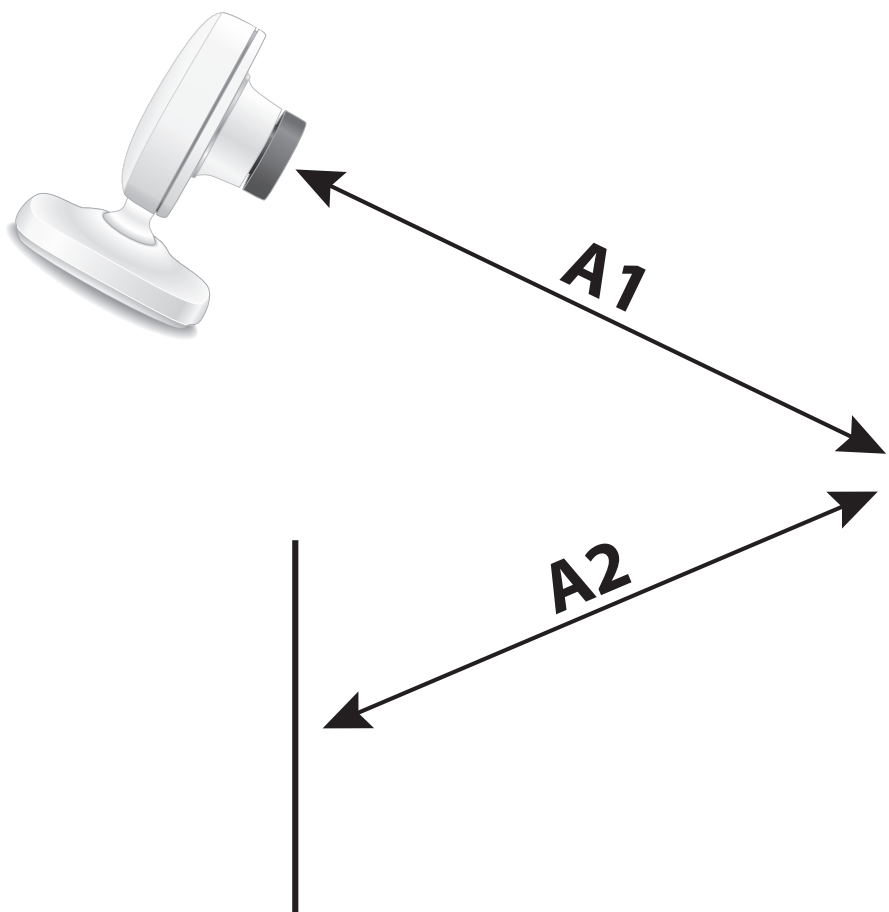
Teoretiskt kan höjden på chassit halveras med hjälp av en plan spegel. Dock är detta inte praktiskt möjligt eftersom

- 1 – inget får skymma ytan och
- 2 – kameran inte kan monteras exakt i toppen på chassit.

Kameran kan dock monteras på sidan av chassit och vinklas för att titta neråt. Om spegeln också vinklas kan en vy över toppen på lådan åstadkommas enligt figur 4.4. Kameran kan antingen placeras på kort eller långsidan. Placeras den på kortsidan finns det fritt utrymme att justera den i höjddled. Placeras den däremot på långsidan så begränsas placeringen av ett kretskort tillhörande bildskärmen.

Att montera kameran på kortsidan kräver att en monteringsanordning byggs eftersom kameran måste monteras på högkant och kan då inte stå på foten (se figur över kamera 4.2).

Monteringskonstruktionen för kretskortet skapar en “naturlig hylla” där kameran kan monteras. Överkanten på denna “hylla” är 70 mm från toppen då den



Figur 4.4: Kamerans synfält utökas genom en spegel

måste placeras intill skärmen. Kameran placerades på denna “hylla” för att inte behöva tillverka ovannämnda monteringsanordning för kameran.

647 mm är det minsta avståndet mellan kameran och ytan som skall registreras. Hälften av detta avstånd är $\frac{647 \text{ mm}}{2} = 324 \text{ mm}$. Minsta möjliga höjd är därmed 324 mm. Höjden på chassit beslutades till 400 mm för att ge möjlighet att montera kameran en bit nedanför toppen av chassit samt ge utrymme för justeringar i efterhand.

4.1.3 Bildinhämtning

Kameran anslöts till en dator med hjälp av USB. En mjukvara kan då, via en kompatibel drivrutin, använda sig av kameran för att inhämta bilder.

Spåraren hämtar in bilder och sedan bearbeta dessa för att utläsa relevant information. Mer om denna procedur presenteras i avsnitt 5.2.

4.2 Pekning

Då optisk pekpuntsregistrering används måste någon av de metoder som beskrivs i avsnitt 3.1.1.2 används för att skapa pekpunkter.

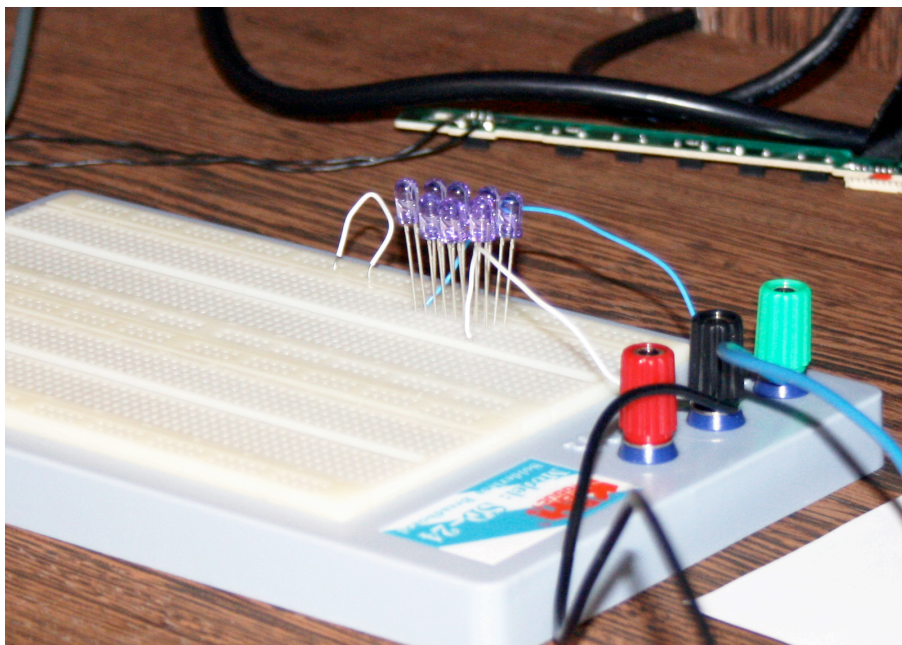
4.2.1 Ljuskälla

Enligt avsnitt 3.1.1 är infrarött ljus en lämplig ljuskälla att använda för pekpuntsregistrering. För denna prototyp har infraröda lysdioder använts med en våglängd av 880 nm [29].

4.2.1.1 Utspridd upplysning

Bland beskrivna metoder i avsnitt 3.1.1.2 är tekniken med *utspridd upplysning* den teknik som är teoretiskt sett enklast att genomföra. Detta eftersom ett antal lysdioder enbart behöver monteras upp i ett jämt mönster.

Ett antal infraröda lysdioder monterades upp på en laborationsplatta som sedan placerades inuti chassit. Då undersidan av plattskärmens panel är blank uppstår reflektioner från ljuskällan och sådana reflektioner är svåröverröstade av reflektioner från pekpunkter (exempelvis fingrar). För att reducera reflektionen och ge en jämnare spridd belysning placeras ett tunt skisspapper under den blanka ytan. Reflektionerna reduceras därmed marginellt.



Figur 4.5: Utspridd belysning uppmonterad

De infraröda lysdioderna måste placeras ut med ett jämnt avstånd mellan varje lysdiod för att kunna ge en jämn belysning över hela ytan som skall registreras.

4.2.2 Problem

4.2.2.1 Utspridd upplysning

Ett problem med den utspridda upplysningen var att det inte gick att med enbart fingrar överrösta de reflektioner som kvarstod efter att skisspappret hade monterats. Det gör det omöjligt att använda denna teknik i den aktuella konstruktionen. För att komma vidare måste en annan teknik användas.

4.2.3 Infraröd ljuspenna

Som lösning på de problem som uppstått med pekpunktsinmatningen (se avsnitt 4.2.2) har två infraröda ljuspennor tillverkats. Pennorna har inbyggda infraröda lysdioder och emitterar själva infrarött ljus i stället för att reflektera, detta gör ljuspunkterna mycket starkare. En fördel med att använda dessa pennor är att de fungerar med alla metoder som beskrivits i avsnitt 3.1.1.2.

4.2.3.1 Uppbyggnad

Pennorna använder en whiteborard-penna som grundstomme. Inuti pennan finns ett AAA-batteri, en infraröd lysdiod, en brytare, ett motstånd, en plexiglasstav samt en metallbricka. Samtliga komponenter finns i tabell 4.3.

Lysdioden har placerats i pennans spets (se figur 4.6). När lysdioden trycks in sluts brytaren, med hjälp av plexiglasstaven, och dioden tänds. Detta medför att pennan kan användas för pekning genom att den enbart sätts ned mot ytan. När pennans spets inte är tryckt mot någon yta är lysdioden släckt vilket medför att pennan då inte registreras som en pekpunkt.

<i>Komponent</i>	<i>Tillverkare och modell</i>	<i>Inköpsställe</i>
OH-penna	Edding 363	
Infraröd lysdiod	Osram SFH485-2	ELFA AB
Tangentbordsswitch		Kjell & Company AB
Batteri 1,5 Volt AAA	Duracell	Clas Ohlson
Motstånd, 48 Ohm		ELFA
Plexiglasstav		
Metallbricka		

Tabell 4.3: Komponenter för konstruktion av infraröda ljuspennor



Figur 4.6: Pekpennas utseende och uppbyggnad

4.3 Bildåtergivning

4.3.1 Val av lösning

För bildåtergivning har en plattskärm (LCD) använts (se avsnitt 3.1.2.3).

4.3.1.1 Plattskärm

Plattskärmen har ett antal icke önskvärda egenskaper, som dessutom kan variera från modell till modell. Eftersom skärmen måste kunna släppa genom infrarött ljus måste bakstycket avlägsnas och så även de kretskort som återfinns där.

Plattskärmen är uppbyggd av en bildpanel ytterst, ett antal filtrerande lager samt en bakgrundsbelysning. En del av de filtrerande lagern har sådana egenskaper att de påverkar ljusets genomflöde på ett sådant sätt att det är omöjligt att detektera pekpunkter. Dessa lager måste tas bort.

I figur 4.7 syns plattskärmens bildpanel utan filtrerande lager men det vita

lagret (se figur 4.8) lämnas kvar. Utan detta får bilden på skärmen avsevärt sänkt kvalité. Detta lager släpper dessutom genom infrarött ljus och utgör därför inget större problem.



Figur 4.7: Plattskärmens bildpanel

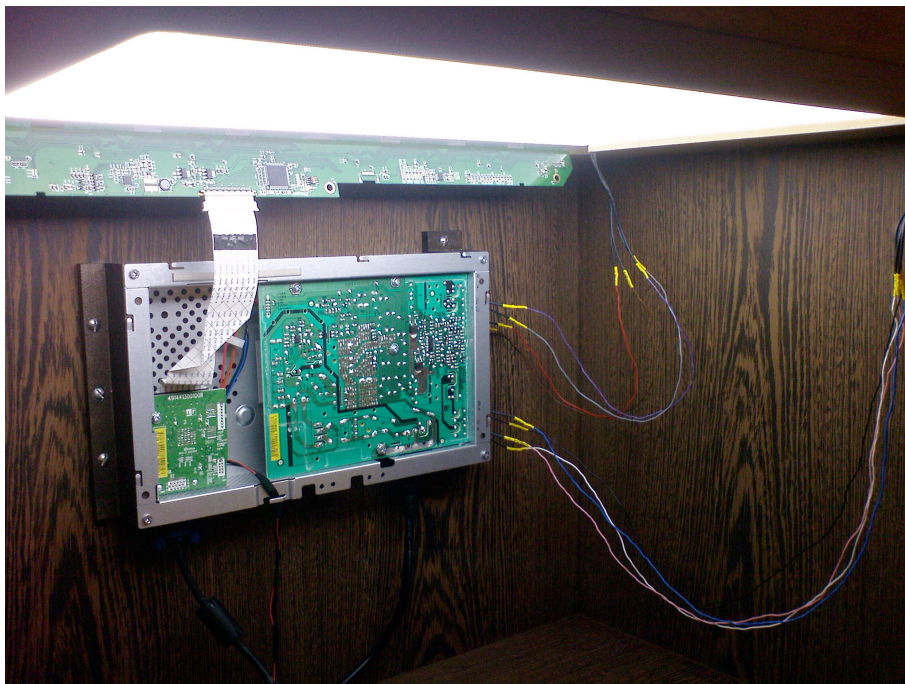
Plattskärmen har monterats i chassit, se avsnitt 4.4, tillsammans med bakgrundsbelysningen (se figur 4.9). Plattskärmens kretskort har nu monterats på chassits vägg. Detta eftersom de inte får vara i vägen för den optiska bildinläsningen, se avsnitt 4.1.1.

Märke, modell och storlek

För ändamålet valdes den billigaste plattskärmen i storlek 22 tum som var lättåtkomlig. Specifikationerna över vilken skärm som används finns i tabell 4.4. Storleken 22 tum valdes utifrån att det är en vanligt förekommande skärmstorlek vilket gör det möjligt att få ett bra pris samt att det är en relativt stor skärm. Priset



Figur 4.8: Lager för att bland annat ge vit färg



Figur 4.9: Plattskärm monterad i chassi. På bilden syns undersidan av skärmen.

söktes med hjälp av webbsidan <http://www.prisjakt.se>.

<i>Tillverkare</i>	<i>Modell</i>	<i>Inköpsställe</i>
LG	W2234S	Nya Internetworking Data i Gbg AB (Inet)

Tabell 4.4: Plattskärmsspecifikationer

4.4 Chassi

Ett chassi har konstruerats i trä. Chassit rymmer all hårdvara utom den dator som tar emot pekinformation och visar gränssnittet.

Ritningar över chassit finns som bilaga B.

4.4.1 Mått

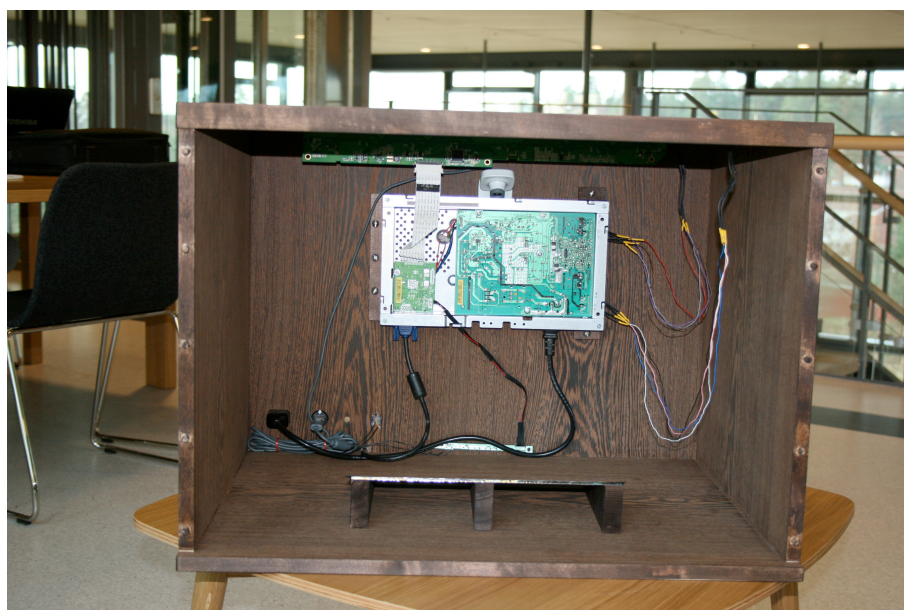
Chassits inre höjd är 400 mm, denna höjd beslutades utifrån de beräkningar som gjordes i avsnitt 4.1.2.2. Bredden och djupet på chassit ges av plattskärmens storlek. Djupet på chassit måste dessutom utökas med bredden på kretskorten och kameran som skall monteras i bakkant. Chassits bredd är 645 mm och djupet är 475 mm.

4.4.2 Intern uppbyggnad

Inuti chassit sitter kontrollerkort för plattskärmen, en kamera, en spegel samt kablar för anslutning. En översiktsbild av chassits interna uppbyggnad finns i figur 4.10

4.4.2.1 Plattskärm

Som tidigare nämnts används en plattskärm (se avsnitt 4.3) för bildåtergivning. Plattskärmen är monterad i chassits topp med sin bildpanel uppåt. Plattskärmen



Figur 4.10: Hårdvarans uppbyggnad internt

har ett antal kretskort som inte får vara monterade får inte blockera kamerans synfält. Dessa kretskort har därför monterats mot chassits bakre vägg (se figur 4.9).

4.4.2.2 Glasskiva

På toppen av chassit har en glasskiva monterats. Syftet med denna skiva är att förhindra att fingrar eller pennor används direkt på plattskärmens bildpanel och på så sätt skadar denna.

4.4.2.3 Kamera

Kameran har monterats på “hyllan” ovan plattskärmen kretskort (se figur 4.11). Detta ger den ett lämpligt avstånd till spegeln och därmed ytan som skall registreras.



Figur 4.11: Kameran är monterad ovan plattskärmens kretskort

4.4.2.4 Spegel

En plan spegel har monterats i botten av chassit. Dess uppgift är att förlänga avståndet mellan kameran och ytan som skall registreras.

<i>Typ</i>	<i>Inköpsställe</i>
3 millimeter tjock plan spegel	Karlstads Nya Glasmästeri AB

Tabell 4.5: Spegelspecifikation

4.4.3 Anslutningar

Chassit har anslutningar för elektricitet, USB samt en D-Sub-15. Electricitetskontakten används för att ge skärmen ström (se figur 4.12). Chassit har även ett intag för USB där en USB-kabel (Typ A till Typ B) kan anslutas mellan datorn och chassit. Internt är kameran ansluten till denna kontakt via en USB Typ A (se figur 4.12).



Figur 4.12: T.v intag till chassit och t.h. intern koppling

För att minimera risken för störningar i signalen till plattskärmen finns inget intag för videosignal monterad på chassit. I stället har videosignalskabeln dragits genom chassit. Detta ger möjligheten att utan skarvning ansluta skärmen direkt till den dator som skall visa användargränssnittet.

4.4.4 Start och avstängning

För att kunna starta plattskärmen måste det vara möjligt att nå den inbyggda start- och avstängningsknappen. För att göra detta möjligt har en knapp monterats vid anslutningarna på chassits baksida, knappen syns i figur 4.12. Denna knapp har internt kopplats över den befintliga start- och avstängningsknappen på plattskärmen.

4.5 Sammanfattning

En hårdvaruprototyp har konstruerats utifrån en plattskärm, en kamera samt infraröda lysdioder. Initialt var planen att använda utspridd upplysning men detta misslyckades och istället skapades ett antal pennor med infraröda lysdioder som kan användas för pekning.

Bildåtergivningen sker med hjälp av en plattskärm vilken har monterats, till-

sammans med kameror och speglar, inuti ett chassi. På toppen av chassit finns en glasskiva som skall förhindra användaren från att skada plattskärmens bildpanel.

Kapitel 5

Mjukvaruprototyp

I detta kapitel kommer detaljerade specifikationer om mjukvaruprototypens konstruktion att förklaras.

Först i det här kapitlet presenteras allmän information om spårare och hur dessa fungerar. Efter detta beskrivs kommunikationen mellan spåraren och applikationen. Näst sist i detta kapitlet presenteras applikationen som har implementerats och sist i kapitlet finns en sammanfattning.

5.1 Översikt

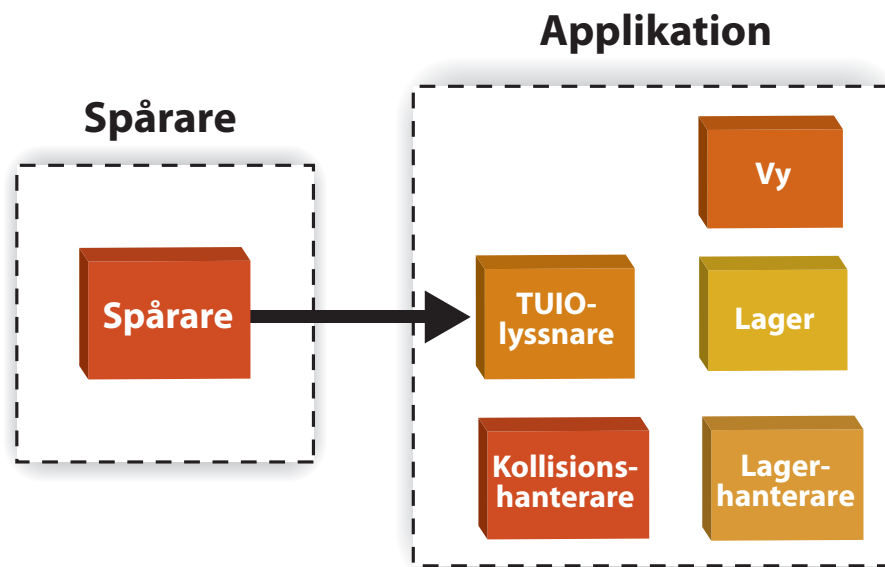
Mjukvaruprototypen består av en spårare samt en applikationsdel och använder en befintlig spårare.

I figur 5.1 syns en översikt över mjukvaruprototypen och som kan ses består mjukvaruprototypen av två isolerade delar, *Spårare* och *Applikation*.

Till vänster syns spåraren (avsnitt 5.2) som skickar information till den applikation som har implementerats (avsnitt 5.4), till höger i figuren.

5.2 Spårare

Hårdvaruprototypen som presenterats i denna uppsats använder optisk pekpunktsregistrering, se avsnitt 4.1. Den information som erhålls är en sekvens av stillbilder. För att tolka dessa bilder och utifrån dessa generera pekpunkter behövs en så



Figur 5.1: Översikt över mjukvaruprototyp

kallad spårare.

5.2.1 Funktionalitet

Bilden innehåller en mängd information men enbart en delmängd av denna information är intressant för tillämpningen, exempelvis är färginformation inte intressant. Genom att applicera lämpliga filter på bilden gallras irrelevant information bort. Den intressanta information har olika utseende beroende på vilka förhållanden som råder när bilden tas. På grund av detta behövs olika filter vid olika förhållanden.

Efter filtrering skall spåraren tolka och analysera den kvarvarande informationen. Spåraren måste identifiera pekpunkter och följa dessa mellan olika bilder i sekvensen. Till sist lämnar spåraren ifrån sig information om de aktiva pekpunkterna och deras position.

5.2.1.1 Filtrering

Filter är matematiska funktioner som appliceras på bilden och ger den ett nytt utseende. Nedan beskrivs några exempel på filter som kan användas och utav dessa användes bakgrundssubtraktion, oksärpa samt kontrast i mjukvaruprototypen.

Bakgrundssubtraktion används för att ta bort huvuddelen av den i bilden icke intressanta informationen. Först sparas en referensbild som inte innehåller några pekpunkter. Från varje bild som skall filtreras subtraheras referensbilden och kvar blir bara de delar av bilden som har ändrats sedan referensbilden togs. Det vill säga i stort sett blir enbart pekpunkterna kvar [30].

Oskärpa är ett filter som används för att reducera bruset i bilden.

Invertering används för att byta ut alla färger mot dess invers. Enkelt sett kan man säga att en pixel med byte-färgvärde 1 får värdet 255, 2 får 254 och så vidare. På så sätt blir vitt svart och svart blir vitt och kan användas då pekpunkter exempelvis uppträder som skuggor istället för ljuspunkter.

Kontrast är ett filter som används för att öka skillnaden mellan de ljusa och mörka områdena i bilden. Detta filter används för att göra svaga ljuspunkter i en mörk bild tydligare så att spåraren senare kan upptäcka dem.

5.2.1.2 Bildsegmentering

Med bildsegmentering skapas en bild där varje bildpunkt representeras av en bit. Det här gör att varje bildpunkt antingen kan anta värdet färgad eller ofärgad (svart eller vit) [32]. Den här tekniken används för att skapa en bild där ljusa områden blir vita och resten blir svart. Ur en sådan här bild är det enklare att finna sammanhängande områden än i en bild med fler färgbiter per bildpunkt.

5.2.1.3 Tolkning och analys

Efter lyckad filtrering och segmentering har spårare en bild med noll eller fler sammanhängande pekområden (se figur 5.2).



Figur 5.2: Bildsegmentering med tre sammanhängande områden

För varje sammanhängande pekområde måste spåraren avgöra om området är tillräckligt omfattande för att klassas som en pekpunkt. Dessutom måste spåraren avgöra om det är en tidigare punkt som flyttats eller en ny punkt.

5.3 Spårare och applikationskommunikation

Den här prototypen använder sig av protokollet TUIO för att hämta in information om pekpunkter. För mer information varför TUIO används, se avsnitt 3.2.3.2.

5.3.1 Om TUIO

TUIO [18] utvecklades som en del av reactTable* [19] (se avsnitt 2.2.2) och bygger på OSC-protokollet. Detta gör det möjligt att använda TUIO på alla system som har stöd för OSC, exempelvis Flash [18].

Innan ytterligare information om TUIO ges kommer en kort presentation om OSC-protokollet och hur det är uppbyggt.

5.3.1.1 OSC

OSC [40], OpenSound Control, är ett protokoll för kommunikation mellan datorer och multimediaenheter som exempelvis synthar. OSC är optimerat för nätverkskommunikation och är konstruerat för klient/server-kommunikation. OSC är dock oberoende av vilket transportprotokoll som används.

OSCs datablock kallas för paket och innehåller ett eller flera meddelanden. Ett meddelande består av en adress, en typsträng samt noll eller fler argument. De datatyper som stöds enligt OSC är ASCII-strängar, 32-bitars flyttal, 32-bitars heltal (integer) samt 64-bitars tidstämplar och binär data [40] [39].

Adressen i meddelandet ger destinationen [39], adresserna är inte förbestämda i protokollet vilket gör att man kan skapa nya sorters adresser. Adressen är uppbyggd av noder som är separerade med ett “/”, exempelvis skulle en svensk postadress kunna se ut enligt “/sverige/karlstad/universitetsgatan/2”. Sista noden i adressen är mottagaren av meddelandet.

Typsträngen är en ASCII-sträng som inleds med ett “,”. Varje tecken i sekvensen motsvarar exakt datatypen för sekvensen av argument i meddelandet. I tabell 5.1 återfinns en lista över alla datatyper som stöds hos argumenten.

Antalet argument i meddelandet ges av antalet tecken i typsträngen och som tidigare nämnt ges även datatypen för argumentet av typsträngen.

OSC Typ	Motsvarande typ
i	Heltal, 32 bitar
f	Flyttal, 32 bitar
s	OSC-sträng
b	OSC-blobdata

Tabell 5.1: OSC Argumenttyper

5.3.1.2 TUIO över OSC

TUIO definierar en uppsättning meddelanden i OSC (se figur 5.3) med uppgift att förmedla information om objekt och pekpunkter. Meddelandena är definierade på så sätt att första argumentet i meddelandet anger vilken typ meddelandet har.

I huvudsak finns två meddelandetyper. En meddelandetyper, *alive*, talar om vilka objekt och pekpunkter som befinner sig på ytan. En annan meddelandetyper, *set*, talar om information om de objekt och pekpunkter som är placerade på ytan. Utöver dessa definierar TUIO ett meddelande som talar om sekvensnummer, *fseq*, samt ett meddelande med namnet på servern, *source*. [18]

```
/tuio/[profileName] source name  
/tuio/[profileName] set [sessionID parameterList]  
/tuio/[profileName] alive [list of active sessionIDs]  
/tuio/[profileName] fseq [int32]
```

Figur 5.3: TUIO, Meddelandetyper

TUIO-meddelanden är uppdelade i profiler (se figur 5.4). Profilnamnet finns lagrad i OSC-adressen och är där ändnoden. Profilens namn föregås av “/tuio/” i adressen [18].

Varje meddelande typ har ett antal argument som varierar beroende på vilken profil som används. Argumenten som används för de olika meddelandena syns i se figur 5.4. Semantiken för dessa argument syns i tabell 5.2 där även datatyper för de olika argumenten finns [18]. Observera att dessa argumenttyper inte skall blandas ihop med OSC-datatyper!

2D Interactive Surface

```
/tuio/2Dobj set s i x y a X Y A m r
/tuio/2Dcur set s x y X Y m
```

2.5D Interactive Surface

```
/tuio/25Dobj set s i x y z a X Y A m r
/tuio/25Dcur set s x y z X Y Z m
```

3D Interactive Surfaces

```
/tuio/3Dobj set s i x y z a X Y Z A B C m r
/tuio/3Dcur set s x y z X Y Z m
```

raw profile

```
/tuio/raw_[profileName]
/tuio/raw_dtouch set i x y a
```

custom profile

```
/tuio/_[formatString]
/tuio/_sixyP set s i x y 0.5
```

Figur 5.4: TUIO, Profiler

5.4 Applikation

Den applikation som har implementerats för att demonstrera några av de möjligheter som finns med flerpunktspeksskärm har kallats *Lime - A Slight Touch of Lime*.

5.4.1 Uvecklingsmiljö

Applikationen använder sig av fyra huvudbibliotek. Dessa tillhandahåller funktionalitet för grafik, kommunikation och datahantering. Applikationen har skrivits i Objective-C för Apple MacOS X 10.5.

Cocoa [13] är en objektorienterad miljö som används för programutveckling i Apple MacOS X. Cocoa består främst av två ramverk; Foundation Kit och Application Kit. Dessa ramverk tillhandahåller bland annat strängar, värde-manipulering, containers samt grafiska element.

<i>Argumenttyp</i>	<i>Semantisk betydelse</i>	<i>OSC-datatyp</i>
s	Sessionsidentitet, temporärt ID	i
i	Klassidentitet	i
x, y, z	Position, mellan 0..1,	f
a, b, c	Vinkel, mellan 0..2PI	f
X, Y, Z	Förflyttningsvektor	f
A, B, C	Rotationsvektor	f
m	Acceleration	f
r	Roteringsacceleration	f
P	Fri parameter	i,f,s,b

Tabell 5.2: TUIO, Meddelandeargument

Quartz 2D [14] är ett två-dimensionellt API för att rita grafik i Apple MacOS X.

Quartz 2D tillhandahåller Portable Document Format (PDF)s ritmodell.

Core Animation [15] är ett API för att hantera och animera grafiklager vilka kan innehålla olika sorters information, ex. vyer, bilder med mera.

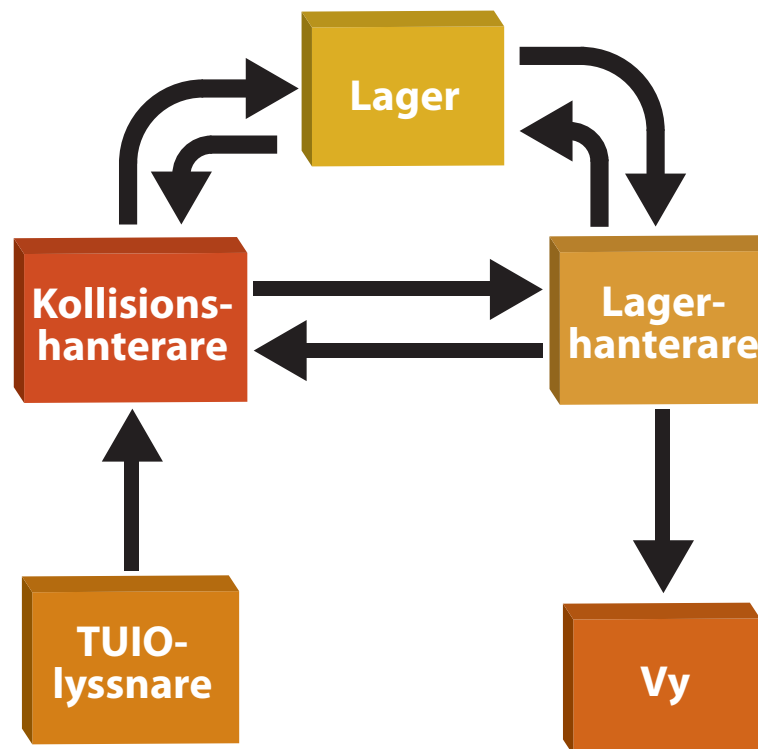
BBOSC [35] används för att ta emot data från spåraren, ytterligare information finns i avsnitt 5.4.2.1.

5.4.2 Implementation

Mjukvaran består av fem delar (se figur 5.5) och bygger på tekniken med lager där varje lager representerar en viss mängd information och där lagren är separerade från varandra. Dessa delars funktionalitet beskrivs av nedan och kan ses i figur 5.5.

TUIO-lyssnare är den komponent som lyssnar efter meddelanden från spåraren.

Lager är en sammansättning av operationer och innehåll. Innehållet kan vara grafik som ritas till vyn.



Figur 5.5: Mjukvaruprototypens delar

Lagerhanterarens uppgift är att ha hand om alla de lager som finns i systemet och kontrollera ordningen de skrivs ut.

Kollisionshanteraren har till uppgift att upptäcka kollisioner mellan pekpunkter och lager.

Vyn är den komponent som visas på skärmen.

Utöver dessa komponenter har tre stycken elementära operationer skapats - rotation, förflyttning och skalning. Denna funktionalitet ligger i *Lager* och kan kombineras fritt för att få önskad funktionalitet. Utöver detta är det också möjligt att introducera nya operationer för *Lager*.

5.4.2.1 TUIO-lyssnare

Applikationen använder sig enbart av en profil från TUIO. Profilen som används är `"/tuio/2Dcur"` och denna representerar punkter i ett tvådimensionellt plan. I samma klass som `"/tuio/2Dcur"` finns även `"/tuio/2Dobj"` vilken kan användas för att unikt identifiera olika sorters objekt men eftersom unikt identifierbara objekt inte är intressant i denna implementation så kommer profil att utelämnas.

I mjukvaruprototypen har en klass, vid namn `TUIOListener`, implementerats. Klassens uppgift är att ta emot TUIO-meddelanden från nätverket och översätta dessa till de meddelanden som återfinns i programkod 5.4. Dessa meddelanden skickas sedan vidare till kollisionshanteraren.

BBOSC

För mjukvaruprototypen har biblioteket BBOSC används för att ta emot och behandla OSC-meddelanden över nätverk. Biblioteket tar emot alla slags OSC-meddelanden och en filtrering måste göras för att se till att endast relevanta meddelanden behandlas.

Implementation

En lyssnare upprättas med hjälp av BBOSC och denna lyssnar på en specifik nätverksport. När data tas emot behandlas detta för att först och främst avgöra om det är ett TUIO-meddelande och dessutom ett TUIO-meddelande i rätt adress (`"/tuio/2Dcur"`).

Data tas emot i form av paket vilka kan innehålla flera TUIO-meddelanden. BBOSC kallar dessa paket för `BBOSCBundle` och meddelanden för `BBOSCMessage`. Programkod 5.1 nedan visar hur `BBOSCMessage` extraheras ur ett `BBOSCBundle` och sedan kontrolleras för att innehålla rätt adress.

När alla meddelanden i ett `BBOSCBundle` har samlats in kommer var och ett av

```
NSArray* messages = [aBBOSCBundle attachedObjects];

for(BBOSCMessages* aMessage in messages)
{
    //Accept only 2D "cursor" information
    if([[[aMessage address] address]
        isEqualToString:@"tuio/2Dcur"])
    {
        //Handle message, check for and
        // handle set, alive, fseq and source
    }
}
```

Programkod 5.1: Hantering av meddelanden

dem att utföras. De meddelanden som är intressantast att utföra är “set” och “alive” eftersom de innehåller verklig pekpunktsinformation.

Meddelanden av typ “set” kontrolleras mot en intern lista över aktiva pekpunkter (se programkod 5.2). Om det inkomna meddelandets pekpunktsidentitet inte tidigare existerar i listan skickas ett `touch:at:` till kollisionshanteraren och i annat fall skickas ett `touch:moveTo:` till kollisionshanteraren.

När det gäller meddelanden av typen “alive” jämförs de aktiva pekpunkterna i meddelandet mot de som finns i den interna listan (se programkod 5.3). Om de punkter som finns med i den interna listan men inte i “alive”-meddelandet kommer ett `unTouch:` skickas till kollisionshanteraren för varje sådan punkt.

5.4.2.2 Lager

Ett lager är ett objekt som det är möjligt att interagera med via pekpunkter och det kan även innehålla visuell information som visas i gränssnittet. Lagret har en utgångsposition, en storlek samt en ordning i djupled. Ordningen talar om vilket lager, om två lager befinner sig på samma position, som skall visas överst och vilket som skall döljas. Lägre ordningstal i djupled gör att lagret visas framför ett lager av högre ordningstal.

```
//touchDelegate is a delegate to the collisionhandler
- (void)processSetMessage:(BBOSCMessage*)message
{
    NSArray* arguments = [message attachedObjects];
    int touchID;
    TUIOPoint touchPoint;

    touchID = [[arguments objectAtIndex:1] intValue];
    touchPoint.x = [[arguments objectAtIndex:2] floatValue];
    touchPoint.y = [[arguments objectAtIndex:3] floatValue];

    if([activeTouchIds containsObject: [NSNumber numberWithInt:
        touchID]])
    {
        //Send "touch-point" move
        if([touchDelegate respondsToSelector:
            @selector(tuioTouch:moveTo:)])
            [touchDelegate tuioTouch:touchID moveTo:touchPoint];
    }
    else
    {
        //Send "touch-point" down
        if([touchDelegate respondsToSelector:
            @selector(tuioTouch:at:)])
            [touchDelegate tuioTouch:touchID at:touchPoint];
    }
}
```

Programkod 5.2: Hantering av "set"-meddelande

Lagerinnehåll

Den grafiska representationen av lagret beskrivs med hjälp av ett CALayer som har placerats i en vy vilken är synlig på skärmen. CALayer är en klass som tillhandahålls av Core Animation och kan innehålla olika sorters media, så som texter, bilder och film.

Lagrets innehåll är det som ritas inuti CALayer som finns i lagret.

5.4.2.3 Lagerhanterare

Lagerhanteraren har en central roll i mjukvaran eftersom den har ansvar för samtliga lager. Lagerhanteraren bestämmer lagrens ordning i djupled, tillhandahåller dem till kollisionshanteraren samt lägger dem i vyn.

Lagerhanteraren har en intern lista i form av en NSMutableArray, en rediger-

```

//touchDelegate is a deltegate to the collisionhandler
- (void)processAliveMessage:(BBOSCMessages*)message
{
    NSArray* arguments = [message attachedObjects];
    NSMutableSet* tempTouchIds = [[NSMutableSet alloc] init];

    for(BBOSCMessages* aArgument in arguments)
    {
        [tempTouchIds addObject:[NSNumber numberWithInt:
                                [aArgument intValue]]];
    }

    for(NSNumber* aTouchID in activeTouchIds)
    {
        if(![tempTouchIds containsObject:aTouchID])
        {
            //Send up
            if([touchDelegate respondsToSelector:
                @selector(tuioUnTouch:)])
            {
                [touchDelegate tuioUnTouch: [aTouchID integerValue]];
            }
        }
    }

    [activeTouchIds release];
    activeTouchIds = tempTouchIds;
}

```

Programkod 5.3: Hantering av "alive"-meddelande

bar vektor tillhålls av Cocoa, som håller samtliga lager i den ordning de presenteras. Lager med djupled 0 ligger också överst i listan. Denna lista tillhåller lagerhanteraren även till kollisionshanteraren som behöver information om vilka lager som finns i systemet (se avsnitt 5.4.2.4).

Utöver detta är det lagerhanterarens uppgift att placera alla lager i vyn. Lagerhanteraren använder vyns `CALayer` där den placerar lagren.

5.4.2.4 Kollisionshanterare

Kollisionshanterarens uppgift är att finna de lager som är intresserade av de punkter som kommer in i systemet. Kollisionshanteraren gör detta genom att hämta en lista över alla lager i systemet från lagerhanteraren, se avsnitt 5.4.2.3, och därefter ställa ett antal fördefinierade frågor.

För att ett lager skall fungera i systemet måste det svara på ett antal fördefinierade frågor. Frågorna används för kommunikation mellan lagren och kollisionshanteraren. Frågorna är definierade via ett informellt protokoll, se programkod 5.4, och ställs då en pekpunkt uppträder första gången.

```
- (bool)interestedInPoint:(NSPoint)point  
    withAZOrderOf:(NSInteger)order;  
  
- (bool)containsPoint:(NSPoint)point;  
  
- (bool)transparentToTouch;
```

Programkod 5.4: Kollision- och lagerkommunikation, frågor

Frågan `containsPoint` syftar till att ge kollisionshanteraren information om ifall ett lager innehåller den specifika punkten eller inte. Trots att lagret alltid är rektangulärt ger denna funktionalitet lagret möjlighet till att rita upp valfri form och enbart tala om vilka punkter som ligger i formen.

Frågan `interestedInPoint` ska ge svar på ifall ett lager är intresserad av en specifik pekpunkt då lagret ligger på ett visst ordningsnummer i djupled. Detta ger lagret möjligheter att själv bestämma vilka pekpunkter som är intressanta. För att illustrera detta kan två lager läggas ovanpå varandra. Det översta lagret kommer få frågan `interestedInPoint:aPoint withAZOrderOf:0;` som denna kan svara på. Lagret under kommer däremot att få frågan `interestedInPoint:aPoint withAZOrderOf:1;`, eftersom denna ligger dolt bakom det första lager. Det viktiga här är att lagret fortfarande har möjlighet att vara intresserad av punkter trots att det inte är överst.

Den sista frågan, `transparentToTouch`, är till för att indikera att lagret är genomskinlig för pekningar. Om lagret svarar YES på denna fråga kan det fortfarande acceptera en pekpunkt men när pekpunkten skickas till nästa lager har inte ordningstalet för djupled räknats upp. Ett exempel för att illustrera detta kan vara två lager som ligger ovanpå varandra. Det översta lagret har satt sin `transparentToTouch`

till YES och tar emot en frågan `interestedInPoint:aPoint withAZOrderOf:0;`. Lagret under detta lager kommer också att få frågan `interestedInPoint:aPoint withAZOrderOf:0;` eftersom ordningstalet inte har räknats upp.

Pekpunktsdata

Efter att Kollisionshanteraren har gjort klart för sig, via ovanstående frågor, vilka lager som är intresserade av pekpunkten sparas denna information lokalt för att användas senare. Pekpunkten har, så länge den existerar, en unik identitet och denna identitet lagras tillsammans med vilka lager som är intresserade av just denna pekpunkt. På så sätt behöver inte frågan om pekpunkten är intressant för lagret ställas flera gånger.

Det finns tre olika meddelanden (se programkod 5.5), hämtade från avsnitt 3.2.2.2, vilka representerar tre olika händelser i en pekpunkts livstid. Dessa meddelanden översätts av `TUIOListener` innan de vidarebefordras till kollisionshanteraren. Kollisionshanteraren använder tre stycken metoder, beskrivna av i programkod 5.5 på varje lager för att delge information om pekpunkter.

```
- (void) touch: (NSInteger) touchID  
          at: (NSPoint) touchPoint;  
  
- (void) touch: (NSInteger) touchID  
          moveTo: (NSPoint) touchPoint;  
  
- (void) unTouch: (NSInteger) touchID;
```

Programkod 5.5: Meddelanden för pekpunter

När en pekpunkt har inkommit (föds) skickas ett `touch:at:-`meddelande till varje lager som tidigare klargjort att densamma är intresserad av pekpunkten. Detta meddelande innehåller information om pekpunktens identitet och placering i planet.

Meddelandet `touch:moveTo:` skickas till intresserade lager för att tala om att en

pekpunkt med ett viss identitet har flyttat sig.

Det sista meddelandet som skickas under en pekpunkts livstid är `unTouch`: vilket indikerar för ett lager att pekpunkten inte längre existerar (död).

5.4.2.5 Vyn

Vyn är implementerad som en `NSView`, vilket är vyklass tillhandahållen av Cocoa. Denna har en `CALayer` i vilken lagerhanteraren placerar sina registrerade lager. Vyn är i sin tur placerad i ett fönster på skärmen och detta medför att de lager som lagerhanteraren placerar i vynes `CALayer` visas på skärmen.

5.4.2.6 Standardlager

Applikationen definierar ett `LimeBasicContentLayer` som innehåller den grundläggande funktionaliteten som ett lager kan tänkas behöva. Denna lagertyp kan ärvas och ger då funktionalitet som besvarar frågorna om vilka pekpunkter som är intressanta (se programkod 5.4) med standardsvar, vilket i praktiken innebär att lagret är intresserad av alla pekpunkter inom dess område när lagret är överst i djupled.

Vad det gäller de meddelanden som levererar information om pekpunkter (se programkod 5.5) håller `LimeBasicContentLayer` ordning på vilka aktiva pekpunkter som är placerade på lagret och samt vilken ordning dessa uppkom. Utöver detta finns även funktionalitet för att hantera innehåll (se avsnitt 5.4.2.2).

Detta lager har grundstommen för lagerinnehåll, det vill säga att den har ett `CALayer` vilket går att tilldela innehåll.

De operationer som presenteras nedan finns implementerade i `LimeBasicManipulatableLayer` vilken ärver från `LimeBasicContentLayer`. `LimeBasicManipulatableLayer` kan användas som den är då den tar emot information om sitt innehåll vid konstruktion, det är också möjligt att ärvä från `LimeBasicManipulatableLayer` för att vidare utveckla

funktionaliteten.

Flyttningsoperation - flytta lager mellan två punkter

Flyttningsoperationen är en relativt simpel operation som är till för att kunna flytta ett lager från en position till en annan. Operationen kräver enbart en pekpunkt för att fungera och förflyttningen sker genom att beräkna skillnaden i x- och y-led när pekpunkten flyttats. Därefter adderas skillnaden till positionen på CALayer. Detta medför att CALayer får en ny position och representerar där med förflyttningen.

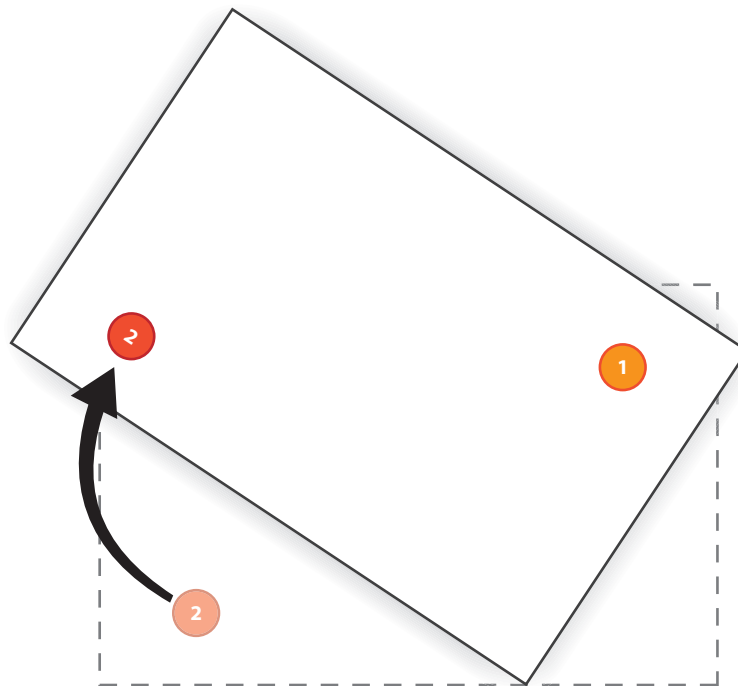
Flyttningsoperationen finns i LimeBasicManipulatableLayer implementerad som `-(void)moveFromPoint:(NSPoint)from toPoint:(NSPoint)to`.

Rotationsoperation - rotera kring ankarpunkt

Rotationsoperationen är en relativt avancerad operation. Rotationesgårdären syftar till att kunna rotera ett lager kring en valbar ankarpunkt i lagret (se figur 5.6) och finns i LimeBasicManipulatableLayer implementerad som `-(void)rotateFromPoint:(NSPoint)from toPoint:(NSPoint)to roundPoint:(NSPoint)anchor`.

För att utföra en rotation krävs två pekpunkter, där en av pekpunkterna agerar ankarpunkt och den andra pekpunkten ger rotationsvinkeln vid förflyttning. Den pekpunkt som inte flyttas kommer att agera ankarpunkt och benämns nedan med ankarpunkt. Den andra pekpunkten, som kommer att utföra en förflyttning, kallas nedan rotationspunkt. För att en rotation skall uppstå måste denna punkt göra en förflyttning.

När rotationspunkten har gjort en förflyttning finns två koordinater tillgängliga för den. En koordinat där punkten flyttats från och en dit punkten flyttats. Först skapas ett nytt koordinatsystem för CALayer med origo i ankarpunkten (se figur 5.7 och programkod 5.6).

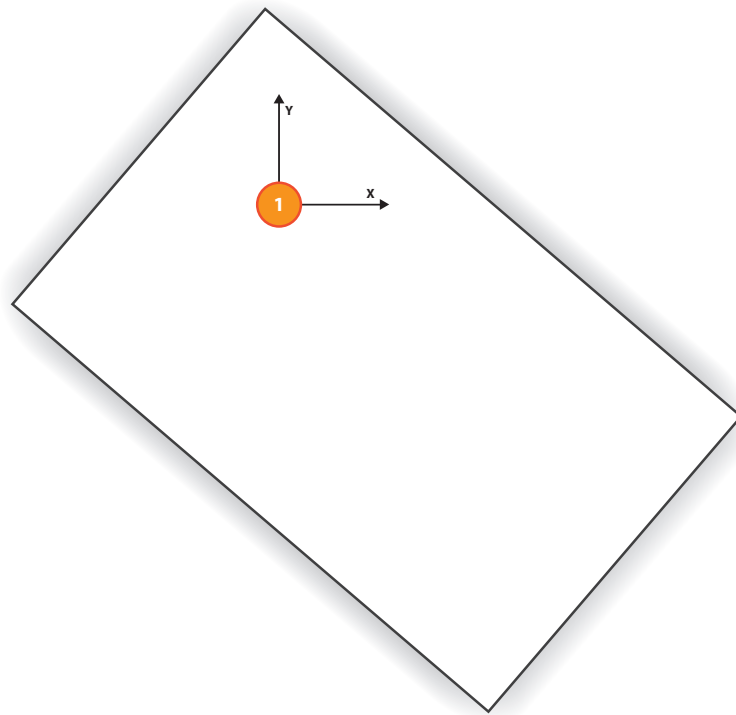


Figur 5.6: Rotation kring ankarpunkt

Vinkeln mellan start och stop på rotationspunkten, här kallat α , beräknas med formel 5.1 [1].

$$\vec{A} \cdot \vec{B} = |\vec{A}||\vec{B}| \cos(\alpha) \quad (5.1)$$

När vinkeln har beräknats är det viktigt att avgöra vilket håll rotationen skall ske åt. Formeln 5.1 ger endast en minsta vinkeln mellan två punkter och därmed ingen riktning. För att bestämma riktningen roteras alla punkter i koordinatsystemet på ett sådant sätt att start för rotationspunkten hamnar på y-axeln. Därefter kontrolleras tecknet på x-värdet för stop-rotationspunkten och i vilken kvadrat som start-rotationspunkten ligger i. Två fall uppstår, ett vid kvadrant *I* och *II* samt ett i kvadrant *III* och *IV*. I kvadrant *I* och *II* ger ett positivt värde på stop-rotationspunktens x-värde en rotation åt höger och ett negativt värde en rotation



Figur 5.7: Nytt koordinatsystem vid ankarpunkten

åt vänster. I kvadrant *III* och *IV* är detta det motsatta, alltså ett positivt x-värde ger vänsterrotation och ett negativt x-värde ger högerrotation.

Rotationen appliceras sedan direkt på `CALayer`.

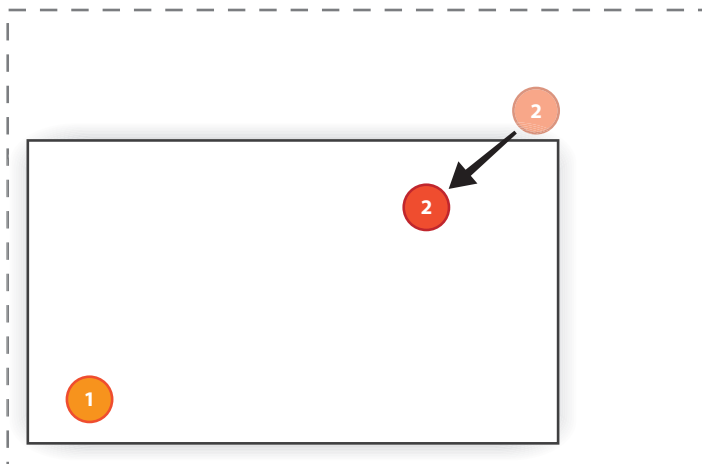
Skalningsoperation - skala lagret

Skalning av lager innebär att lagrets storlek ändras proportionellt och för detta behövs två pekpunkter. En av pekpunkterna kommer här att användas som referenspunkt och den andra bestämmer skalfaktorn. Skalningsoperationen finns i `LimeBasicManipulatableLayer` implementerad som `-(void)scaleFromPoint:(NSPoint)from toPoint:(NSPoint)to withReferencePoint:(NSPoint)reference`.

När pekpunkten som bestämmer skalfaktorn flyttas så registreras koordinaten som punkten flyttas från samt koordinaten som punkten flyttas till. Detta ger två

```
//Make coordinateSystem with origo in anchor.  
// anchor = anchor point  
// from = rotation point start coordinate  
// to = rotation point end coordinate  
  
CGAffineTransform translateToAnchorPoint =  
    CGAffineTransformMakeTranslation(-1.0 * anchor.x, -1.0 *  
    anchor.y);  
  
CGPoint fromTranslated = CGPointApplyAffineTransform(  
    NSPointToCGPoint(from), translateToAnchorPoint);  
  
CGPoint toTranslated = CGPointApplyAffineTransform(  
    NSPointToCGPoint(to), translateToAnchorPoint);
```

Programkod 5.6: Skapa nytt koordinatsystem för rotation



Figur 5.8: Skalning av lager

sträckor och den procentuella skillnaden mellan dessa två sträckor ger skalningen av lagret (se figur 5.8).

Det kan vara så att `CALayer` har skalats tidigare och detta innebär att det redan finns en skalfaktor applicerad. Skalfaktorn måste därför beräknas utifrån att det redan kan finnas en skalning, vilket syns i programkod 5.7. Efter detta kan den nya skalfaktorn (`newScaleFactor`) appliceras direkt på `CALayer`.

```
CGFloat originalDistance =
    distanceBetweenPoints(NSPointToCGPoint(reference),
        NSPointToCGPoint(from));

CGFloat newDistance =
    distanceBetweenPoints(NSPointToCGPoint(reference),
        NSPointToCGPoint(to));

CGFloat currentScaleFactor = [[[self primaryLayer]
    valueForKeyPath:@"transform.scale"]
    floatValue];

CGFloat calculatedScale = currentScaleFactor *
    (newDistance / originalDistance) -
    (1.0f * currentScaleFactor);

CGFloat newScaleFactor = currentScaleFactor + calculatedScale;
```

Programkod 5.7: Beräkning av ny skalfaktor

Egendefinierade operationer

Genom att introducera en ny lagertyp i systemet finns det möjlighet att här definiera nya operationer utan att behöva ändra i omgivande programkod.

Genom att skapa ett nytt lager som implementerar det tidigare nämnda protokollet, kallat *Touching*, (se programkod 5.8) kan valfritt beteende skapas. Det är också möjligt att ärva från `LimeBasicContentLayer` (eller `LimeBasicManipulatableLayer`) för att erhålla den grundläggande funktionaliteten. Det är i så fall möjligt att åsidosätta¹ den funktionalitet som implementeras av `touch: at:`, `touch moveTo:` och `unTouch:` och därefter använda sig av metoderna för flytt, skalning och rotation.

5.4.2.7 Problem

Ett antal problem uppstod under programmeringen av logik för olika lager. Nedan beskrivs problematiken samt lösningen.

¹Engelska: override

```
@interface NSObject (Touching)

- (void)touch:(NSInteger)touchID
      at:(NSPoint)touchPoint;

- (void)touch:(NSInteger)touchID
      moveTo:(NSPoint)touchPoint;

- (void)unTouch:(NSInteger)touchID;

- (bool)interestedInPoint:(NSPoint)point
      withAZOrderOf:(NSInteger)order;

- (bool)containsPoint:(NSPoint)point;

- (bool)transparentToTouch;

@end
```

Programkod 5.8: Det informella protokollet Touching

Insättning av ankarpunkt

Ett layers position bestäms av två egenskaper, det ena är koordinaten för lagret själv och det andra är var denna koordinat skall appliceras på lagret (exempelvis i mitten). `CALayer` är implementerat så att koordinaten för lagret appliceras i ankarpunkten för densamma.

När en godtycklig punkt sätts som en ankarpunkt på ett `CALayer` uppstår en förflyttning av lagret. Detta är en logiskt beteende eftersom ankarpunkten har flyttats. För att lagret inte visuellt skall flyttas måste positionen ändras för att kompensera. Positionen måste ändras lika mycket som ankarpunkten har flyttats, detta gör att lagret visuellt ligger kvar.

Roteringsalgoritm

Någon algoritm för att bestämma hur mycket ett lager skall roteras utifrån ett antal punkter var inte tillgänglig. I stället fick en algoritm utarbetas med en matematisk formel som grund. Utifrån formel 5.1 kunde en vinkel mellan två vektorer erhållas och problemet var bestämma ifall en höger- eller vänsterrotation skulle

utföras.

Flera olika lösningsmetoder togs fram utan lyckat resultat i samtliga fall som uppstod. Utifrån de olika lösningsmetoderna kunde gemensamma nämnare härledas vilket gav en funktionsduglig algoritm. Denna algoritm implementerades med framgång.

Skalningsalgoritmen

När skalningsalgoritmen implementerades uppstod stora problem på olika ställen som kunde härledas till insättning av ankarpunkten. Exempel på problem som uppstod var att rotation inte längre skedde kring ankapunkten.

Anledningen till problemet med insättningen av ankarpunkten var att ingen hänsyn togs till skalfaktorn när lagrets position kompenserades för ankarpunktens förflyttning. Problemet avhjälpes genom att ta hänsyn till lagrets skalfaktor.

5.5 Sammanfattning

Mjukvaruprototypen använder sig av en spårare för att läsa av den bildström som kan erhållas från hårdvaruprototypen. Spåraren överlämnar information om pekpunkter via TUIO-protokollet till applikationen. I applikationen översätts meddelandena till ett protokoll.

Applikationen har ett tre inbyggda operationer; skalning, rotation och förflyttning. Dessa typer kan användas via ett lager vilket också kan ta emot ett valfritt innehåll. Egendefinierade operationer kan införas i applikationen genom att implementera en ny lagertyp, på detta sätt blir applikationen utbyggbar.

Kapitel 6

Utvärdering av prototyp

I det här kapitlet kommer en utvärdering av både hårdvaru- och mjukvaruprototypen att ges. Utvärderingen syftar först och främst till att kontrollera prototyperna mot de krav som finns. Först utvärderas hårdvaran och därefter mjukvaran. Sist i detta kapitel sammanfattas utvärderingen.

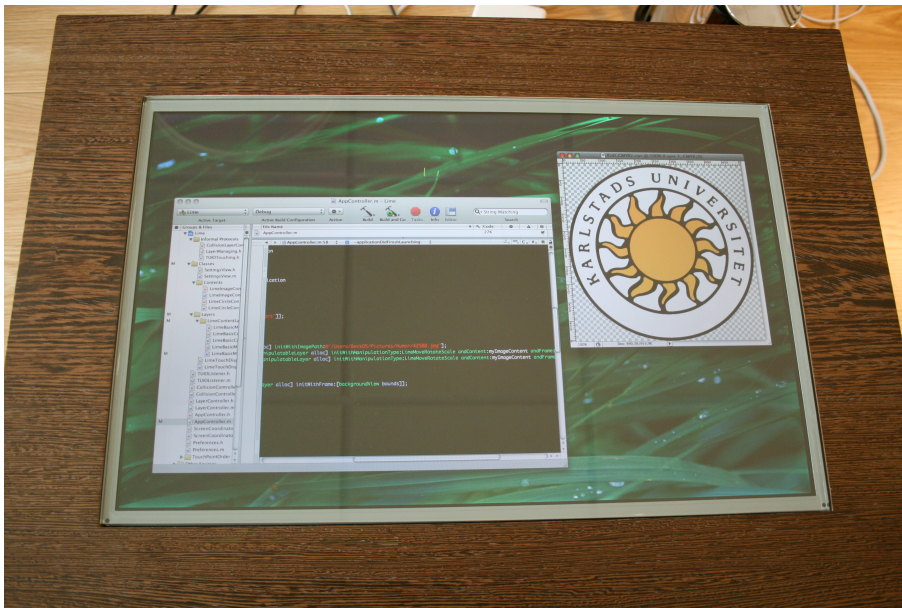
6.1 Utvärdering av krav

6.1.1 Hårdvara

Det fanns totalt tre krav ställda på hårdvaran enligt avsnitt 3.3.

6.1.1.1 Grafiskt gränssnitt

Hårdvaran skall enligt första kravet visa det grafiska gränssnittet för applikationen. För att uppfylla detta krav använder sig hårdvaran av en plattskärm som är ansluten till en dator via en D-Sub-15-kontakt, se avsnitt 4.4.3. Kravet kontrolleras genom att se att plattskärmen kan visa det gränssnitt som programmerats. Figur 6.1 som visar gränssnittet på skärmen. Plattskärmen är dessutom på bilden monterad i chassit.



Figur 6.1: Bildåtergivningen fungerar

6.1.1.2 Registrering av pekpunkter

Det andra kravet för hårdvaran var att den skall ge i från sig en stillbildsserie över pekytan. För att skapa denna stillbildsserie används en kamera som registrerar pekytan (se avsnitt 4.1.1). Kameran ger i från sig en kontinuerlig sekvens av bilder som kan läsas av tolkas av kameran.

Kravet bestod också i att att pekpunkterna skall vara tillräckligt framträdande för att kunna detekteras av spåraren. För att skapa pekpunkter som är detekterbara av spåraren används pennor med inbyggda infraröda lysdioder (se avsnitt 4.2.3).

För att testa så att hårdvaran uppfyller kravet om att pekpunkterna skall vara tillräckligt framträdande aktiveras kameran och spåraren. En testapplikation lyssnar på TUIO-meddelanden från spåraren. En pekpena (se avsnitt 4.2.3) trycks ned mot ytan på hårdvaran och testapplikationen tar emot meddelanden från spåraren att en pekpunkt är aktiv. När pennan lyfts erhåller testapplikationen ett

meddelande om att pekpunkten är borta.

6.1.1.3 Kostnad

Det sista kraven för hårdvaruprototypen var att den inte får kosta mer än 4000 SEK kr att tillverka. Nedan finns (i tabell 6.1) en sammanställning på inköpskostnader för de komponenter som ingår i hårdvaruprototypen.

<i>Komponent</i>	<i>Pris</i>
LCD-skärm	1890,00 SEK
Kamera	149,00 SEK
Glas & spegel	170,00 SEK
Trä till chassi	500,00 SEK
IR-dioder	22,40 SEK
Tangentbordsswitch	15,00 SEK
Batterier	30,00 SEK
Plexiglasstav	30,00 SEK
USB-adapter	67,80 SEK
Apparatintag	31,90 SEK
Isolationshuv	7,99 SEK
Total	2914,09 SEK

Tabell 6.1: Prislista

Som synes i tabell 6.1 överstiger *inte* den totala konstruktionskostnaden det övre taket på 4000 SEK.

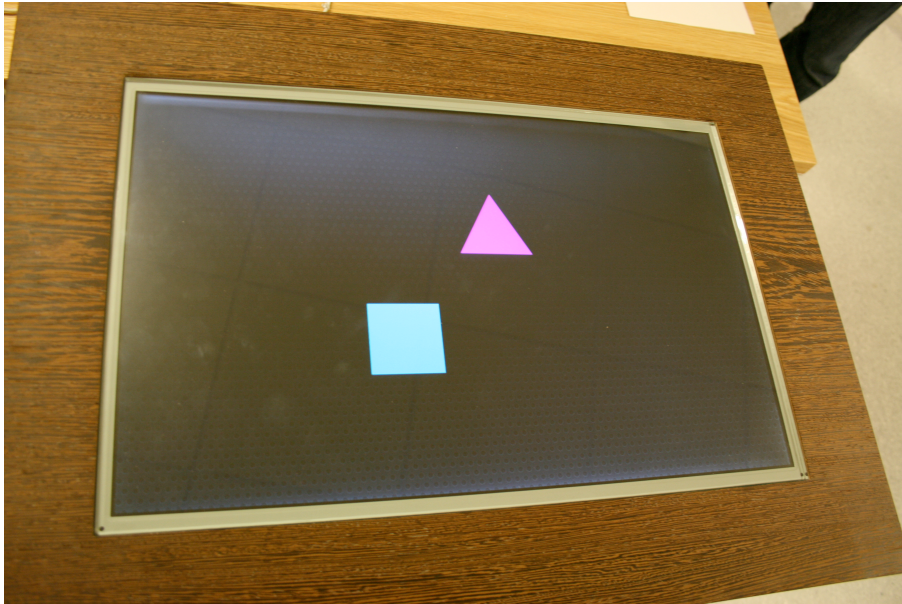
6.1.2 Mjukvara

Enligt avsnitt 3.3 fanns två krav på mjukvaran.

6.1.2.1 Operationer

I figur 6.2 syns två separata objekt, en triangel och en rektangel. I programkod 6.1 syns hur två lager med figurer läggs in, först läggs `mySquareLayer` till och därefter

`myTriangleLayer`. Detta innebär att de figurer som syns i figur 6.2 verkligen är två separata objekt.



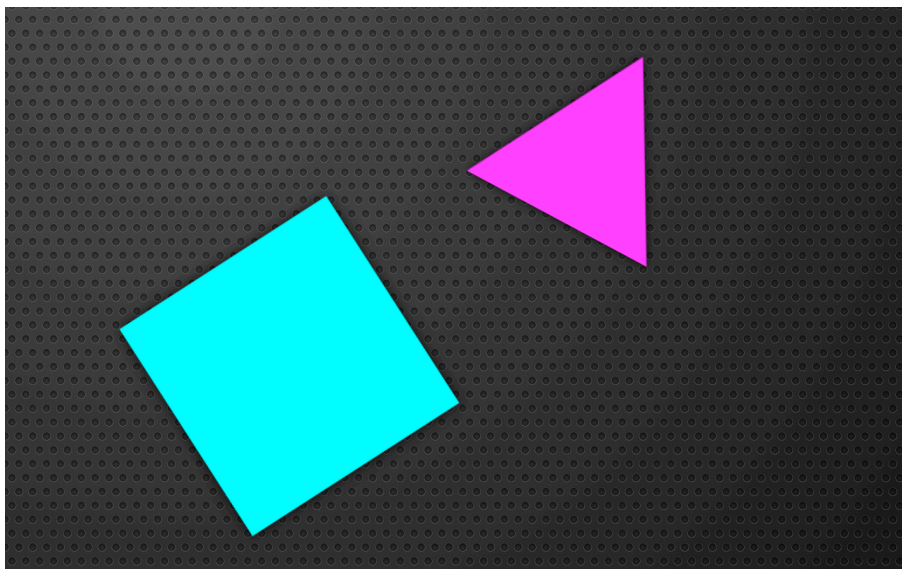
Figur 6.2: En ögonblicksbild av gränssnittet

I samma krav på mjukvaran står det att det skall vara möjligt att rotera, skala och flytta kvadraten och triangeln ovan. I och med att `LimeBasicManipulatableLayer` används för att representera dessa figurer går detta att utföra. I figur 6.3 kan man se att både rotation och skalningen fungerar som önskat.

6.1.2.2 Utbyggbarhet

Det andra kravet för mjukvaran beskriver att densamma skall vara implementerade på ett sådant sätt att det enkelt skall gå införa nya typer av operationer. För att visa detta har ett “piano” implementerats och kan ta emot flera tangentryckningar på en och samma gång och

Figur 6.4 visar en skärmbild av hur användargränssnitt är uppbyggt. En ny lagertyp har introducerats till *Lime* och den kallas för `LimePiano`. `LimePiano`



Figur 6.3: Rotation och skalning

ärver från `LimeBasicContentLayer` för att inte behöva implementera hela protokollet `Touching`. Gränssnittet för `LimePiano` kan ses i programkod 6.2.

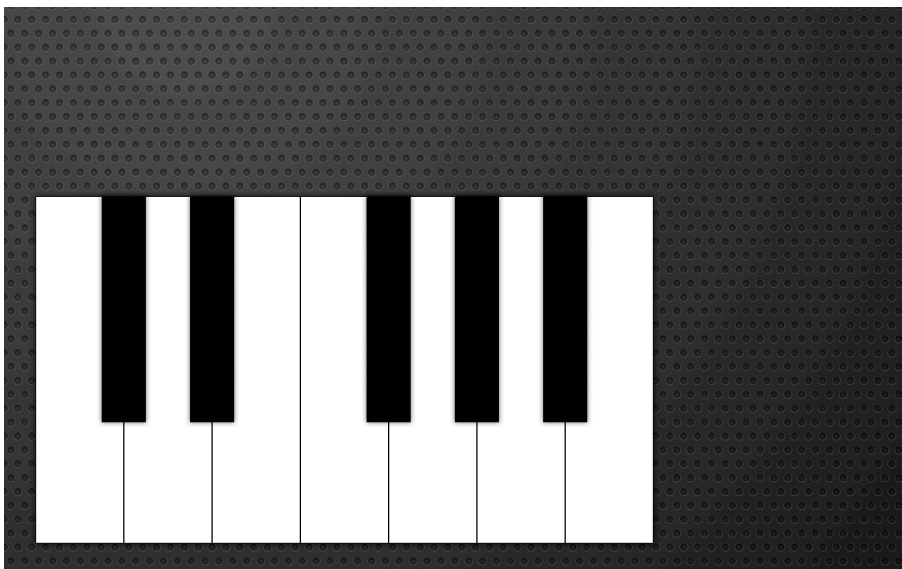
Pianot spelar upp olika toner oberoende av varandra vid “anslag” av tangenter men endast en pekpunkt kan slå an en tangent vid ett och samma tillfälle. Däremot är det möjligt för godtyckligt antal pekpunkter att slå an olika tangenter samtidigt.

6.2 Iakttagelser

6.2.1 Fördröjningar i interaktionen

Interaktionen med applikationen via hårdvaruenheten kan upplevas ge seg respons. Orsaken till detta är inte fullt utredd, men tänkbara anledningar kan vara:

Kameran har en bildhastighet på 30 bilder per sekund, detta borde vara fullt tillräckligt. Om kameran kopplas mot en applikation som visar kamerans bild i realtid kan en viss fördröjning mellan verkligheten och den visade



Figur 6.4: Skärmbild av pianot

bilden upplevas. Denna fördröjning skulle kunna bero på följande orsaker: tiden det tar från det att bilden tas tills den skickas ut på kabeln, drivrutinen som används för att kommunicera med kameran är långsam, mjukvara som används för att testa kameran är långsam.

Spåraren gör en stor mängd filtreringar på varje stillbild som kommer från kameran. Dessa filtreringar är beräkningsintensiva och tar därför en viss mängd tid i anspråk.

Applikationen gör även den en del beräkningar för varje pekpunkt som spåraren tillhandahåller den och även dessa beräkningar tar tid i anspråk.

Nätverket kan vara en flaskhals i sammanhanget. Vid testerna av systemet används dock *loopback*-gränssnittet och transportprotokollet *UDP*, med dessa förutsättningar bör inte nätverket vara en bidragande faktor till fördröjningarna.

Tester har utförts genom att simulera pekpunkter via en applikation som sänder TUIO-meddelanden över *loopback*-gränssnittet för att manipulera objekt i applikationen. Testerna är inte vetenskapligt utförda eftersom svarstider i form av data inte är relevant i detta sammanhang och det finns inga andra data att jämföra med. Trots detta kunde man med ögat se en markant skillnad från när hårdvaruenheten används, vilket tyder på att problemen med fördröjningen ligger i hårdvaruprototypen eller spåraren.

6.2.2 Grafikproblem

Den dator som hårdvaruprototypen var kopplad till hade stundvis problem med att rendera grafik. Problemen till detta tros ligga i hårdvaran och inte mjukvaran då samma problem inte uppstår på andra datorer. Då datorn som används för att visa gränssnittet är en bärbar Macintosh har ett grafikkortsutbyte uteslutits.

6.3 Sammanfattning

De krav som initalt var ställda på prototyperna har uppfyllts. Priset för hårdvaruprototypen var under de uppsatta 4000 SEK. Hårdvaruprototypen visar en bild av gränssnittet och tar emot pekpunkter som den vidarebefordrar till mjukvaruprototypen.

Mjukvaruprototypen är utvecklad så att det ska vara möjligt att introducera nya typer av innehåll men även nya operationer. För att visa att detta fungerar har ett "piano" implementerats vilket fungerar tillfredställande.

```

/*
 * Create a square content.
 * This object is responsible for drawing a square in the CALayer
 */
LimeRectangleContent *mySquare;
mySquare = [[LimeRectangleContent alloc]
             initWithColor:[NSColor cyanColor]
             andHeight:180
             andWidth:180];

/*
 * Create a triangle content.
 * This object is responsible for drawing a
 * triangle in the CALayer.
 */
LimeTriangleContent *myTriangle;
myTriangle = [[LimeTriangleContent alloc]
              initWithColor:[NSColor magentaColor]
              andHeight:180
              andBase:200];

/*
 * Create a layer with the basic manipulation operations.
 * Let mySquare be the content of this layer.
 */
LimeBasicManipulatableLayer *mySquareLayer;
mySquareLayer = [[LimeBasicManipulatableLayer alloc]
                 initWithManipulationType:LimeMoveRotateScale
                 andContent:mySquare
                 andFrame:NSMakeRect(100, 100,
                                     180, 180)];

/*
 * Create a layer with the basic manipulation operations.
 * Let myTriangle be the content of this layer.
 */
LimeBasicManipulatableLayer *myTriangleLayer;
myTriangleLayer = [[LimeBasicManipulatableLayer alloc]
                  initWithManipulationType:LimeMoveRotateScale
                  andContent:myTriangle
                  andFrame:NSMakeRect(180, 200,
                                      200, 200)];

/*
 * Register the layers with the layercontroller.
 * This will make them appear on screen.
 */
[layerController addLayer:mySquareLayer];
[layerController addLayer:myTriangleLayer];

[mySquare release];
[myTriangle release];
[mySquareLayer release];
[myTriangleLayer release];

```

Programkod 6.1: Två objekt placeras


```

@interface LimePiano : LimeBasicContentLayer
{
    QTMovie* keyC;
    QTMovie* keyD;
    QTMovie* keyE;
    QTMovie* keyF;
    QTMovie* keyG;
    QTMovie* keyA;
    QTMovie* keyB;
    QTMovie* keyCSharp;
    QTMovie* keyDSharp;
    QTMovie* keyFSharp;
    QTMovie* keyGSharp;
    QTMovie* keyASharp;
}

- (NSPoint)translatePointFromParentCoordinates:(NSPoint)point;
- (void)handleTouch:(NSPoint)touchPoint;
- (void)playKey:(NSString*)key;

@end

```

Programkod 6.2: LimePianos gränssnitt

Kapitel 7

Slutsats

I detta kapitel kommer en slutsats för den här uppsatsen att presenteras. Först ges en översikt över arbetet som utfört och reflektioner kring detta.

Efter det presenteras framtida vidareutveckling av detta projekt med aspekter i vad som behöver tas hänsyn till när användargränssnitt utvecklas till pekskärmar/flerpunktspekskärmar.

7.1 Detta projekt

De prototyper som har konstruerats fungerar och uppfyller de krav som är ställda på dem. Det finns dock, enligt vår åsikt, vissa aspekter som kunde arbetas ytterligare med.

7.1.1 Hårdvara

Hårdvaruprototypen byggdes inte helt efter initiala planer då pekning med fingrarna misslyckades, i stället introducerades pennor som används för pekning. Vår åsikt, kring pennorna, är att de inte är lika bra som fingrarna eftersom man förlorar känslan av att arbeta direkt med händerna.

Pennorna är dock en bra kompromiss då ingen ytterligare tid fanns för att implementera någon annan metod. Det är möjligt att utveckla prototypen för att använda sig av frustrerad total intern reflektion och på så sätt möjliggöra att

finger kan användas för att navigera på skärmen.

7.1.2 Mjukvara

Mjukvaran fungerar som det var planerat från början och det går att utöka den med nya lagertyper. Att skapa en ny lagertyp kräver tyvärr lite kunskaper om hur systemet fungerar i övrigt, detta tycker vi är lite synd men det borde gå att göra modellen lite mer separerad än vad den är idag.

I övrigt är applikationen i vissa hänseenden lite långsam och kan behöva optimeras. Det finns säkerligen en del programkod i applikationen som kan reduceras eller bytas ut mot kod som är mindre processorkrävande.

7.1.3 Summering

På det stora hela är, enligt oss, systemet funktionsdugligt. Dock är fördröjningen för lång mellan det att en gest utförts till att den uppdaterar objekten på skärmen. Denna fördröjning går säkerligen minska genom att optimera alla led enligt tidigare resonemang, men det är inget som det funnits tid till.

7.2 Framtida vidareutveckling

Det finns, enligt oss, gott stöd för att vidareutveckla applikationen in till olika projekt.

7.2.1 Användargränssnittsanpassning

Applikationen som utvecklats till denna rapport kan anpassas för att fylla olika sorters behov när det gäller mjukvara till flerpunktspeksskärmar. Det är därför möjligt att vidareutveckla mjukvaran till att passa de behov som för tillfället finns.

När ett nytt användargränssnitt utvecklas till applikationen finns några viktiga aspekter att ta hänsyn till.

7.2.1.1 Storlek och position

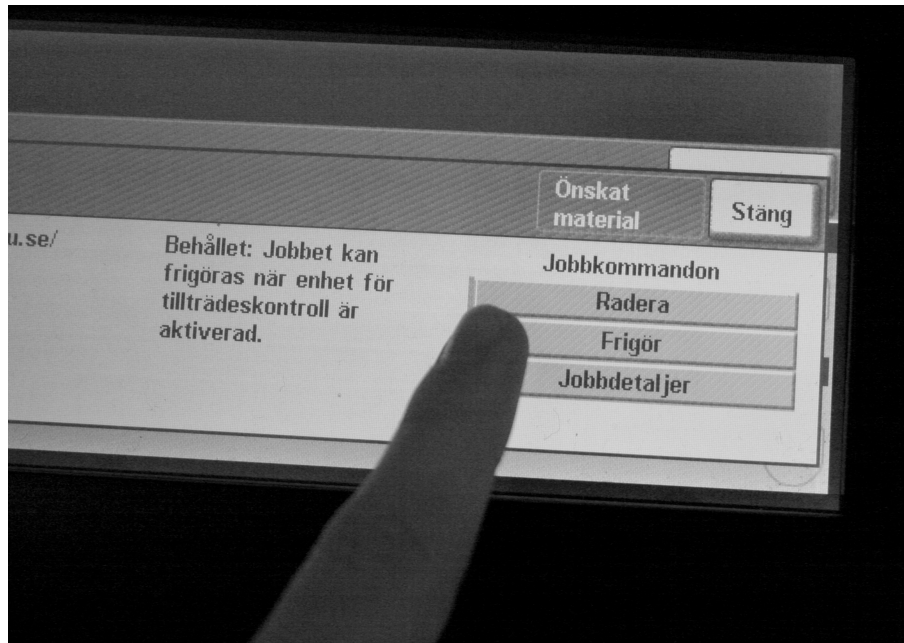
Pekskärmen eller flerpunktspeksskärmen kan inte ersätta musen rakt av eftersom en pekare på skärmen har andra egenskaper än mänskliga fingrar eller pekpennor. När ett användargränssnitt utformas till pekskärmar eller flerpunktspeksskärmar finns det flera fysiska begränsningar man måste ta hänsyn till. En vuxen människas fingertoppar är mellan 10 - 14 millimeter breda [31]. Detta medför att hänsyn måste tas när objekt storleksbestäms och positioneras.

Figur 7.1 visar ett exempel på en pekskärm där tre knappar ligger för nära varandra och dessutom är för små. En användare kan ha problem med att vara säker på att denne trycker på rätt knapp och i detta fall är skillnaden mellan "Radera" och "Frigör" mycket stor, där det ena alternativet radera dokumentet medan det andra skriver ut.

Det är också relevant att ta hänsyn till var objekt placeras på skärmen eftersom användarens hand kan dölja för tillfället intressanta objekt. Ett bra exempel på ett system med pekskärm är SJ ABs biljettkiosker där knappen som tar användaren vidare i köpprocessen är placerad längst ned på skärmen vilket innebär att användaren ser när skärmen uppdateras. I annat fall kan användaren missa att skärmen uppdateras och på så sätt tappa greppet om var i processen man är.

7.2.1.2 Fördröjning

För att en användare inte skall uppleva användandet av mjukvaran som långsamt eller "störande" måste hastigheten på systemet ökas. Det skulle kunna vara nog med att byta hårdvaruprototypen mot något som en tillverkare av flerpunktspeksskärmar kan leverera och därefter implementera en tolk för att delge applikatio-



Figur 7.1: Ett dåligt användargränssnitt för pekning

nen pekpunkter från denna.

Om det är så att det enbart räcker med en ny hårdvaruprototyp för att lösa problemet med hastigheten är inget som vi kan säga då enbart tester med simulator har utförts. Det finns dock stor anledning att tro att en sådan substitution skulle ge effekt eftersom en kommersiell produkt oftast är optimerade för sitt ändamål.

7.3 Lärdomar

Under arbetet har djupare kunskap erhållits inom följande områden:

Visuell spårning används för att identifiera pekpunkter på hårdvaruenheten och för att få dessa att fungera har vi behövt lära oss om hur spårare fungerar.

Objective-C är det programspråk som är använt för att implementera applikatio-

nen. Här har vi lärt oss om hur Objective-C är uppbyggt samt hur Apples fönsterhantering, via Cocoa, fungerar.

Historia kring inmatningssystem har gett oss kunskap om hur människor och datorer förr interagerade med varandra.

Dator-människa-interaktion är ett område som denna uppsats har nosat lite på och vi har lärt oss några viktiga egenskaper kring hur människor interagerar med datorer. Dessutom har vi lärt oss om hur anv

Litteraturförteckning

- [1] Robert A. Adams. *Calculus, A Complete Course*. Pearson Education LTD., 2003.
- [2] Bill Buxton. Internetsida, Sep 2008. Multi-Touch Systems that I Have Known and Loved.
- [3] Microsoft Corporation. Microsoft Surface, Sep 2008. <http://www.microsoft.com/surface>, läst Sept. 2008.
- [4] RAND CORPORATION. Rand 2007 annual report. Årsrapport, 2007.
- [5] Anders Lotsson DATATERMSGROUPEN KTH. E-post mellan anders lots-son och henrik bäck. Sep 2008.
- [6] M.R Davis and T.O Ellis. The rand tablet: A man-machine communication device. pages 1–21, May 1964.
- [7] W den Boer, A Abileah, P Green, and T Larsson. 56. 3: Active matrix lcd with integrated optical touch screen. *SID Symposium Digest of Technical Papers*, Jan 2003.
- [8] Morrison et al. Passive touch system and method of detecting user input. Patent 20 070 075 982, United States Patent and Trademark Office, Apr 2007.
- [9] M Grossman and D Walter. Teaching with interactive computer capabilities (plato: Computer-based education for animal breeding). *Dairy Sci*, (61):1308 – 1311, Sep 1978.
- [10] Jefferson Y Han. Low-cost multi-touch sensing through frustrated total internal reflection.
- [11] Apple Inc. Pressmedelände, SAN FRANCISCO, MACWORLD, Jan 2007. <http://www.apple.com/pr/library/2007/01/09iphone.html>, läst Sept. 2008.

- [12] Apple Inc. Pressmedelande, California, CUPERTINO, Jul 2008. <http://www.apple.com/pr/library/2008/07/14iphone.html>, läst Sept. 2008.
- [13] Apple Inc. Leopard Guides, Cocoa, Aug 2008. <http://developer.apple.com/documentation/Cocoa/index.html>, läst Nov. 2008.
- [14] Apple Inc. Leopard Reference: Graphics & Imaging, Quartz, Aug 2008. <http://developer.apple.com/reference/GraphicsImaging/idxQuartz-date.html>, läst Nov. 2008.
- [15] Apple Inc. What Is Core Animation?, Maj 2008. http://developer.apple.com/documentation/Cocoa/Conceptual/CoreAnimation_guide/Articles/WhatisCoreAnimation.html, läst Nov. 2008.
- [16] Tyco Electronics Inc. Internetsida, Sep 2008. <http://www.elotouch.com/AboutElo/History>, läst Sept. 2008.
- [17] S Jordà. Sonigraphical instruments: from fmol to the reactable. *Proceedings of the 2003 conference on New interfaces for ...*, Jan 2003.
- [18] M Kaltenbrunner, T Bovermann, and R Bencina. Tuio: A protocol for table-top tangible user interfaces. *Proc. of the The 6th Int'l Workshop on Gesture in Human- ...*, Jan 2005.
- [19] M Kaltenbrunner, G Geiger, and S Jordà. Dynamic patches for live musical performance. *Proceedings of the 2004 conference on New interfaces for ...*, Jan 2004.
- [20] G Roberts Lawrence. The lincoln wand. *Proceedings of the AFIPS Fall Joint Computer Conference*, pages 223–227, Sep 1966.
- [21] S.K Lee, W Buxton, and K.C Smith. A multi-touch three dimensional touch-sensitive tablet. *CHI '85 PROCEEDINGS*, pages 21–25, 1985.
- [22] Asahi Spectra Co. Ltd. Produkt, California, Torrance, Jan 2007. http://www.asahi-spectra.com/opticalfilters/syousaik2_ver2.asp?key=YBPA880, läst Nov. 2008.
- [23] Nobuyuki Matsushita and Jun Rekimoto. Holowall: designing a finger, hand, body, and object sensitive wall. *UIST 97*, pages 209–210.
- [24] Nima Motamedi. Hd touch: multi-touch and object sensing on a high definition lcd tv. *CHI '08: CHI '08 extended abstracts on Human factors in computing systems*, Apr 2008.

- [25] Brad A Myers. A brief history of human computer interaction technology. *ACM interactions*, 5(2):44–54, Mar 1998.
- [26] N-trig. Pressmedelände, N-trig's DuoSense™ Digitizer Embedded in Intel's New UrbanMax Mobile Computer, Aug 2008. <http://www.n-trig.com/Content.aspx?Page=PressReleases&PressReleaseId=315>, läst Sept. 2008.
- [27] Russell Owen, Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. When it gets more difficult, use both hands – exploring bimanual curve manipulation. page 8, Mar 2005.
- [28] Frank L. Pedrotti and Leno D. Pedrotti. *Introduction to Optics*. Pearson Education LTD., 1993.
- [29] Robert Lubkoll Björn Bollensdorff Prof. Marc Alexa, Karl Wessel. LEDs and IR Filters, Project Multi-touch Display, Jan 2008. <http://user.cs.tu-berlin.de/~karlw/LEDSpdf2.pdf>, läst Nov. 2008.
- [30] J Rekimoto and N Matsushita. Perceptual surfaces: Towards a human and object sensitive interactive display. *Workshop on Perceptual User Interfaces (PUI97)*, Jan 1997.
- [31] Dan Saffer. *Interactive Gestures: Designing Gestural Interfaces*. O'REILLY, 2008.
- [32] George C. Shapiro, Linda G. Stockman. *Computer Vision*. Prentice Hall, 2002.
- [33] B Shneiderman. Touch screens now offer compelling uses. *IEEE Software*, 8(2):93–94, Mar 1991.
- [34] Ben Shneiderman and Cathrene Plaisant. *Designing the user interface*. Addison-Wesley, fourth edition.
- [35] Ben Britten Smith. [benbritten.com](http://benbritten.com/blog/?s=BBOSC), Nov 2008. <http://benbritten.com/blog/?s=BBOSC>, läst Nov. 2008.
- [36] Norbert A Streitz, Jörg Geißler, Torsten Holmer, Shinichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-land: an interactive landscape for creativity and innovation. *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 120–127, 1999.

- [37] Entertainment Design TED, Technology. Speakers Jeff Han: Human-computer interface designer, Sep 2008. http://www.ted.com/index.php/speakers/jeff_han.html, läst Sept. 2008.
- [38] P Wellner. The digitaldesk calculator: Tangible manipulation on a desk top display. *ACM UIST*, pages 27–33, Nov 1991.
- [39] Matthew Wright. Open Sound Control 1.0 Specification, Sep 2002. http://opensoundcontrol.org/spec-1_0, läst Okt. 2008.
- [40] Matthew Wright, Adrian Freed, and Ali Momeni. Opensoundcontrol: State of the art 2003. *Proceedings of the 2003 Conference on New Interfaces for Musical Expression*, pages 154–159, May 2003.

Bilaga A

Figurreferenser

Figurer nämnda i denna bilaga har ägs av respektive upphovsman nedan, övriga figurer är framtagna för denna uppsats och får användas fritt.

Karlstads universitets sol: Karlstads universitet Figur 2.2: RAND Corporation

Figur 2.3: Wikipedia, GNU Free Documentation License

Figur 2.4: Xavier Sivecas, ReactTable*

Figur 2.5: Apple Inc.

Figur 2.6: Microsoft Corporation

Figur 4.2: Microsoft Corporation

Bilaga B

Ritningar för chassi