Juhana Jauhiainen

OHJELMOINTIKIRJASTO MONIKOSKETUSSOVELLUSTEN KEHITTÄMISEEN

Opinnäytetyö

KESKI-POHJANMAAN AMMATTIKORKEAKOULU

Mediatekniikan koulutusohjelma

Toukokuu 2010

KESKI-POHJANMAAN AMMATTIKORKEAKOULU
MELLERSTA ÖSTERBOTTENS YRKESHÖGSKOLA

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö	**Aika**	**Tekijä/tekijät**
Ylivieska	Toukokuu 2010	Juhana Jauhiainen

Koulutusohjelma
Mediatekniikka

Työn nimi
Ohjelmointikirjasto monikosketussovellusten kehittämiseen

Työn ohjaaja	**Sivumäärä**
Mika Luimula	43

Työelämäohjaaja
Ville Autio

Monikosketustekniikka tekee mahdolliseksi tietokoneen tai mobiililaitteen käyttämisen usealla sormella yhtä aikaa. Monikosketussovellusten kehittäminen kuitenkin vaatii omat ohjelmointikirjastonsa, jotta sovelluskehitys on mahdollisiman sujuvaa. Opinnäytetyössä kehitettiin työkaluja monikosketussovellusten kehittämiseen, toteutettiin yleisimpien monikosketuseleiden algoritmit ja kehitettiin monikosketussovelluksia.

Opinnäytetön tuloksena syntyi ohjelmointikirjasto, jota voi käyttää monikosketussovellusten kehittämiseen. Kirjaston lisäksi tuloksena ovat esimerkkiohjelmat, joita sovelluskehittäjä voi käyttää omien sovellustensa pohjana. Opinnäytetyön ohjelmointikirjastoa käytetään Keski-Pohjanmaan ammattikorkeakoulun tutkimus- ja kehitysyksikön CENTRIAn monikosketussovellusten kehittämisessä.

Työssä käsitellään monikosketuksen historiaa, monikosketusnäyttöjen tekniikkaa, käyttöliittymien ja ohjelmointirajapintojen suunnittelua sekä monikosketussovellusten ohjelmointia.

Asiasanat
käyttöliittymä, monikosketus, ohjelmointikirjasto

CENTRAL OSTROBOTHNIA
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

CENTRAL OSTROBOTHNIA UNIVERSITY OF APPLIED SCIENCES Ylivieska	**Date** May 2010	**Author** Juhana Jauhiainen
Degree programme Media Technology		
Name of thesis Programming library for multitouch software development		
Instructor Mika Luimula		**Pages** 43
Supervisor Ville Autio		

Multi-touch technology enables using touchscreen computers and mobile devices with multiple fingers. In order to make multi-touch application development easy and straight forward, specialized programming libraries and tools are required. In this thesis, a multi-touch programming library was developed along with a number of multi-touch applications. The programming library includes implementations of some common multi-touch gestures.

The result of the thesis was a programming library for multi-touch software development and multi-touch applications that demonstrate using the library. Application developers can use these example applications as a basis for their own applications. The programming library. The library was used in CENTRIA research & development for developing multi-touch applications.

The thesis covers the history multi-touch, the technologies used in multi-touch screens, design of user interfaces, design of application programming interfaces and programming multi-touch applications.

Key words
multi-touch, programming, user interface

TIIVISTELMÄ

ABSTRACT

SISÄLLYS

LYHENTEET

1 JOHDANTO 1

2 MONIKOSKETUSTEKNIIKAN HISTORIA JA NYKYTILA 3
2.1 Monikosketuksen historia 3
2.2 Optinen monikosketus 6
2.3 Kapasitiivinen ja resistiivinen monikosketustekniikka 8
2.4 Yleisimmät laitetyypit 9

3 TAUSTAJÄRJESTELMÄT JA OHJELMISTOARKKITEHTUURI 10
3.1 Optisten näyttöjen taustajärjestelmät 10
3.2 Tangible User Interface Object -protokolla 12

4 MONIKOSKETUSKÄYTTÖLIITTYMÄT 15
4.1 Natural User Interface 15
4.2 Käyttöliittymän suunnittelu 17
4.3 Eleet 19
4.3.1 Eleiden jaottelu 19
4.3.2 Eleet osana käyttöliittymää 20
4.4 Näyttö tietokoneen hallintalaitteena 21

5 OHJELMOINTIRAJAPINTOJEN SUUNNITTELU 23
5.1 Komponenttien suunnittelu 24
5.2 Komponenttien rajapinnat 25

6 KIRJASTO MONIKOSKETUSOVELLUSTEN KEHITTÄMISEEN 27
6.1 Kirjaston arkkitehtuuri ja käyttö 27
6.2 Tangible User Interface Object -signaalien vastaanottaminen 29
6.3 Objektien hallinta ja tapahtumien lähettäminen 30
6.4 Yleisimpien eleiden toteutus 31
6.5 Perusluokat monikosketuselementtien kehittämiseen 34

7 ESIMERKKIOHJELMAT 36
7.1 Valokuvademo 36
7.2 Pistepilven selaaminen 37
7.3 Karttasovellukset 38

8 POHDINTA JA TULOKSET 41

LÄHTEET

LYHENTEET

CCV	Community Core Vision
CLI	Command Line Interface
DI	Diffused Illumination
FTIR	Frustrated Total Internal Reflection
GUI	Graphical User Interface
LCD	Liquid Crystal Display
NUI	Natural User Interface
OSC	Open Sound Control
TUIO	Tangible User Interface Object
UDP	User Datagram Protocol
WIMP	Windows, Icons, Menus, Pointing devices
XML	eXtensible Markup Language

1 JOHDANTO

Monikosketus on tekniikka ja käyttöliittymäajattelumalli, jossa tietokonetta tai mobiililaitetta hallitaan montaa yhtäaikaista kosketusta tukevan näytön avulla. Se on saanut viime vuosina paljon huomiota erilaisten monikosketuslaitteiden tultua myös kuluttajamarkkinoille. Itse tekniikkaa on kuitenkin kehitetty jo pidemmän aikaa.

Työni keskittyy ohjelmistokehitykseen ja sen helpottamiseen suurille optisille, eli valoon perustuville monikosketusnäytöille. Tähän mennessä CENTRIAn, Keski-Pohjanmaan ammattikorkeakoulun tutkimus- ja kehitysyksikön, ohjelmistokehitys optisille näytöille on tehty käyttäen kolmannen osapuolen kehityskirjastoja. Ohjelmistokehityksen kannalta tärkeimpiä vaikuttajia ovat avoimen lähdekoodin yhteisöt, Microsoft sekä eri monikosketusnäyttöjen valmistajat kuten helsinkiläinen MultiTouch. Kaikilla näillä on omat rajapintansa sovellusten kehittämiseen.

Työ lähti liikkeelle tarpeesta saada Nokian Qt-sovelluskehitysympäristön kanssa yhteensopiva monikosketuskirjasto CENTRIA tutkimus ja kehityksen 2D-karttamoottorin avulla toteutettavia karttasovelluksia ja muita C++-sovelluksia varten. Työtä suunnitellessani saatavillani ei ollut vielä tähän sopivia sovelluskirjastoja. Kaikki edelliset toteuttamani monikosketussovellukset, lukuun ottamatta Adoben Flash-tekniikalla toteutettuja, täytyi tehdä käyttäen pelkästään raakaa kosketusdataa ilman kehittyneempiä käyttöliittymäelementtejä. Tämä rajoitti ohjelmistokehityksen hyvin yksinkertaisiin sovelluksiin.

Työssä kehitin Qt-sovelluskehitysympäristöön perustuvan monikosketussovellusten kehittämistä helpottavan C++-kirjaston ja kirjaston ominaisuuksia esitteleviä demoja. Kirjastoa käyttäen on tehty myös asiakassovellus, jota käsitellään tarkemmin luvussa seitsemän.

Seuraavassa luvussa käsitellään monikosketuksen ja siihen liittyvien tekniikoiden historiaa ja nykytilaa. Luvussa kolme siirrytään kuvaamaan optisten monikosketusnäyttöjen

taustajärjestelmiä ja ohjelmistoarkkitehtuureja. Luvussa neljä käsitellään monikosketuskäyttöliittymiä, niiden suhdetta nykyisin käytössä oleviin käyttöliittymiin ja tietokoneen hallintalaitteisiin. Luvussa neljä käydään läpi myös asioita, jotka tulee ottaa huomioon monikosketussovellusten käyttöliittymiä suunnitellessa. Viidennessä luvussa käydään läpi sovelluskirjastojen kehitysperiaatteita.

Luvussa kuusi käsitellään opinnäytetyössä kehitettyä C++-kirjastoa, sen toimintaa ja käyttöä sovelluskehittäjän näkökulmasta. Seitsemännessä luvussa esitetään kirjaston avulla kehitettyjä ohjelmistoja. Viimeisessä luvussa pohditaan työn tuloksia ja tavoitteiden täyttymistä.

2 MONIKOSKETUSTEKNIIKAN HISTORIA JA NYKYTILA

Monikosketuksella tarkoitetaan tietokoneen hallintatekniikkaa ja laitteistoa, joka mahdollistaa tietokoneen käytön ilman hiirtä tai näppäimistöä käyttäen niiden sijaan käsiä ja yhtä tai useampaa sormea sovelluksien hallintaan. Monikosketus koostuu näytöstä, pöydästä, seinästä tai kosketuslevystä ja ohjelmistosta, joka tunnistaa monta yhtäaikaista kosketuspistettä. Perinteiset kosketusnäytöt tunnistavat vain yhden kosketuspisteen kerrallaan. Monikosketus saattaa tulevaisuudessa olla yksi niistä tekijöistä, jotka mullistavat tavan, jolla ihmiset käyttävät tietokoneita. Se mahdollistaa suoran ja intuitiivisen tavan olla vuorovaikutuksessa graafisten käyttöliittymien kanssa. (Muller 2008.)

2.1 Monikosketuksen historia

Monikosketus on viime vuosina saanut paljon huomiota etenkin uusien sitä tukevien mobiililaitteiden takia, mutta idea ei ole kovin uusi. Ensimmäiset montaa yhtäaikaista kosketusta havaitsevat sensorit kehitettiin jo 80-luvulla. Vuonna 1983 Myron Krueger esitteli Video Place- ja Video Desk-nimellä tunnetun interaktiivisen hallintalaitteen, joka kuvasi käyttäjää ja päätteli tämän siluetin liikkeistä tehtävät toiminnot. Kruegerin Video Place esitteli ensimmäisenä nykyään paljon käytetyn nipistyseleen (katso luku 6.4) ja muita konsepteja joiden pohjalle nykyisetkin monikosketuskäyttöliittymät rakentuvat. (Buxton 2009a.)

Ensimmäinen varsinainen monikosketusnäyttö kehitettiin vuonna 1984 Bellin laboratoriossa Yhdysvalloissa. Se sisälsi läpinäkyvän kapasitiivisen (katso luku 2.3) kosketussensorin, joka asetettiin näytön päälle. Monikosketuskäyttöliittymien kehityksessä merkittävä edistysaskel otettiin vuonna 1991, kun Pierre Wellner esitteli Digital-Deskin. Se oli kosketuspöytä, joka tunnisti monta kosketusta yhtäaikaa ja sisälsi monia nykyään käytössä olevia eleitä, kuten graafisten objektien skaalaus ja siirtely kahden sormen avulla. (Buxton 2009a.)

KUVIO 1. Video Place sisälsi mm. nipistyseleen.

Monet yritykset ja yliopistot ovat kehittäneet omia monikosketukseen liittyviä tekniikoita viimeisen parin kymmenen vuoden aikana. Tärkeimpiä näistä nykyisin käytössä olevien tekniikoiden kannalta ovat olleet Fingerworks, Mitshubishi, Stantum, Microsoft, Barcelonan Pombeu Fabra yliopisto sekä New Yorkin yliopisto.

Fingerworks on kehittänyt jo vuodesta 1995 monikosketusta hyödyntäviä näppäimistöjä ja kannettavien kosketuslevyjä. Vuonna 2005 Yhdysvaltalainen Apple osti Fingerworksin. Tämän jälkeen Apple on lisännyt kannettaviin tietokoneisiinsa monikosketusta tukevat kosketuslevyt, ja tuonut markkinoille monikosketusta tukevia mobiililaitteita, kuten iPhone ja iPad. (Buxton 2009a.)

Mitsubishin vuonna 2001 esittelemä Diamond Touch pystyy tunnistamaan kosketuspisteiden lisäksi käyttäjän kädet sekä sen, mihin käteen mikäkin sormi kuuluu, ja näin erottelemaan eri käyttäjät toisistaan. Mitshubishin tutkijat ovat julkaisseet paljon tutkimuksia monikosketuskäyttöliittymien käytettävyydestä ja eleiden suunnittelusta. Erityisesti Mitsubishun tutkijat ovat tutkineet hiiren ja monikosketustekniikan eroja käytettävyyden kannalta. (Dietz & Leigh 2001.)

Ranskalainen Stantum kehitti ensimmäisen kaupallisen läpinäkyvän monikosketuspaneelin silloista päätuotettaan Lemur-musiikkikontrolleria varten, mutta on sittemmin keskittynyt valmistamaan kosketuspaneeleita muun muassa puhelimiin. Stantumin monikosketuspaneelit

käyttävät resistiivistä (katso luku 2.3) kosketustekniikkaa. Lemur on monikosketusnäytöllinen ääni- ja mediakontrolleri, jonka etuna on se, että käyttöliittymää voi muokata kuinka haluaa käyttöliittymän ollessa ohjelmistopohjainen. (JazzMutant 2009.)

Microsoftin TouchLight oli ensimmäinen taustaprojisoitu kosketusnäyttö, eli näyttö, johon kuva tulee kosketuspinnan takana olevasta projektorista. TouchLight sisälsi infrapunataajuuksisia LED-lamppuja, ja se käytti kahta kameraa käyttäjän käsistä ja sormista heijastuvan infrapunavalon seuraamiseen. TouchLightissa pintamateriaalina käytettiin akryylikalvoa, jolloin pinta oli läpinäkyvä. (Buxton 2009a.)

KUVIO 2. ReacTable monikosketusta hyödyntävä soitin.

Barcelonassa on vuodesta 2003 asti kehitetty reacTable nimistä elektronista musiikkisoitinta, joka käyttää käsin kosketeltavaa käyttöliittymää (tangible interface). ReacTablen käyttöliittymässä käyttäjä liikuttelee läpinäkyvällä pöydällä olevia esineitä, jotka edustavat esimerkiksi syntetisaattoreita tai ääniefektejä. ReacTablen kehitysyhteisö on kehittänyt työkaluja, joita voi hyödyntää käsin kosketeltavien käyttöliittymien ohjelmoinnissa, sekä monikosketuspöydille sovelluksia tehtäessä. (Reactable Systems 2010.)

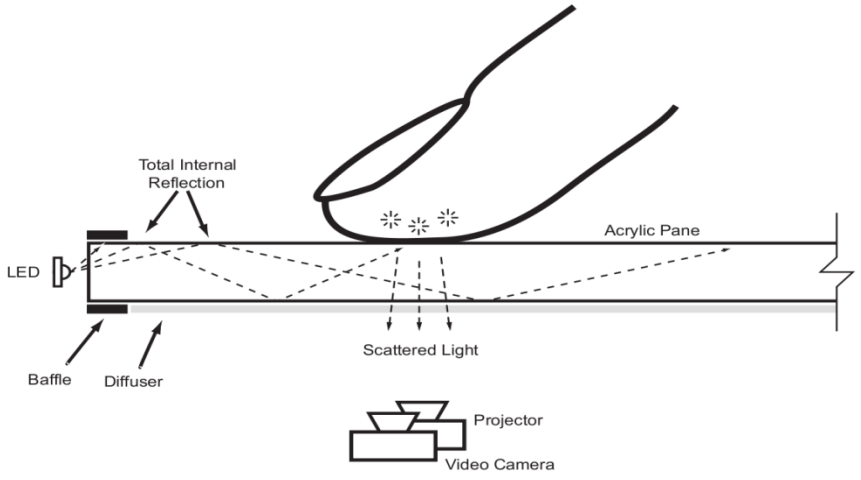
Vuonna 2005 New Yorkin yliopiston tutkija Jeff Han esitteli Frustrated Total Internal Reflection (FTIR) tekniikan, jolla voi rakentaa isoja seinällä tai pöytämuodossa olevia kosketusnäyttöjä alhaisin kustannuksin. Hänen kehittämänsä tekniikka kuuluu optisiin kosketuspöytätekniikoihin. Jeff Hanin kehittämistä tekniikoista ja muista optisista kosketuspöytätekniikoista on tullut hyvin suosittuja, sillä niiden avulla pystytään kustannustehokkaasti rakentamaan suuria monikosketuspöytiä tai -seiniä. Optisten

kosketuspöytätekniikoiden ympärille on syntynyt myös avoimen lähdekoodin yhteisöjä, jotka kehittävät työkaluja monikosketussovellusten kehittämiseen. (NuiGroup Authors 2009.)

Suuren yleisön tietoisuuteen monikosketuksen ovat viime vuosina tuoneet Applen iPhone sekä Microsoftin Surface. Monikosketusnäyttöjen tulo kuluttajien ulottuville suuressa mittakaavassa on siis kestänyt yli kaksikymmentä vuotta. Applen iPhone on älypuhelin, jossa on 3,5 tuuman kapasitiivinen monikosketusnäyttö. Microsoft Surface taas on suuri monikosketuspöytä, joka käyttää optisia tekniikoita kosketuksen tunnistamiseen. Surface esiteltiin vuonna 2008, ja sen käyttöjärjestelmänä toimi tuolloin Windows Vista. Microsoftin uusi Windows 7 käyttöjärjestelmä pystyy hyödyntämään Surfacen monikosketusominaisuuksia. (Buxton 2009a.)

2.2 Optinen monikosketus

Optisista monikosketustekniikoista on tullut suosittuja Jeff Hanin esiteltyä Frustrated Total Internal Reflection (FTIR) tekniikkaa vuonna 2006 Technology Entertainment Design -konferenssissa. Optisten ratkaisujen etuna on skaalautuvuus, edullisuus ja yksinkertainen rakenne. Optiset tekniikat muistuttavat toisiaan hyvin paljon ja ovat kaikki kamerapohjaisia monikosketusjärjestelmiä. Optisia monikosketustekniikoita ovat FTIR, Diffused Illumination (DI), Stereo Visio ja Laser Light Plane. (NuiGroup Authors 2009.)



KUVIO 3. FTIR tekniikan toiminta.

CENTRIAn kehittämä monikosketuspöytä käyttää FTIR-tekniikkaa, ja toinen CENTRIAn pöydistä on MultiTouchin tekemä DI-tekniikkaa käyttävä pöytä. FTIR ja DI ovatkin käytetyimpiä optisista tekniikoista, ja siksi tässä työssä keskitytään niiden käsittelyyn.

Sekä FTIR-tekniikan että DI-tekniikan perusosia ovat projektori, tai LCD-paneeli, kamera ja infrapunavalonlähde. DI-tekniikasta on muutamia erilaisia versioita riippuen siitä, missä infrapunavalonlähde sijaitsee. Pääpiirteittäin optiset tekniikat kuitenkin toimivat siten, että yksi tai useampi kamera kuvaa kosketuspintaa sen takaa ja infrapunavalonlähde valaisee pinnan joko sen reunoilta, takaa tai edestä. Kameran kuvan vastaanottaa ohjelmisto, joka analysoi sitä. (NuiGroup Authors 2009.)

Kun käyttäjä laittaa sormensa tai minkä tahansa valoa heijastavan esineen kosketuspinnalle, analysointiohjelmisto tunnistaa kosketuspisteen alaspäin heijastuvan infrapunavalon avulla. Analysointiohjelma käyttää muutamia algoritmeja kosketuspisteiden erottamiseen taustasta, ja häiriöiden poistamiseen kuvasta. Riippuen käytetyn kameran tarkkuudesta, voidaan analysointiohjelmistoa ajaa melko vaatimattomasti varustelluilla koneillakin. Optisten näyttöjen taustajärjestelmistä, eli ohjelmistoista, jotka hoitavat kosketuspisteiden tunnistuksen, kerrotaan tarkemmin luvussa 3.1. (Han 2005.)

Optisilla tekniikoilla toteutetuissa kosketusnäytöissä esiintyy joskus virheellisiä kosketuspisteen tunnistuksia. FTIR-tekniikalla toteutetuissa pöydissä virheellisiä tunnistuksia esiintyy vähemmän, ja FTIR-tekniikalla toteutetut pöydät ovat myös tarkempia kuin DI-tekniikalla toteutetut pöydät. Virheelliset tunnistukset saattavat ilmetä esimerkiksi siten, että käyttäjän painaessa jotain pistettä kosketuspinnalla, saattaa ohjelmisto tunnistaa useita lähes yhtäaikaisia kosketuksia tästä kohdasta. (Han 2005.)

DI- ja FTIR-tekniikoilla on myös eroja, varsinkin näyttöjen rakenteen, mutta myös ohjelmoinnin kannalta. FTIR-tekniikkaa käytettäessä ei näytön tarvitse olla ”suljettu laatikko” toisin kuin DI-tekniikassa. FTIR vaatii akryylimuovista tehdyn kosketuspinnan, kun taas DI:n kanssa voi käyttää mitä tahansa läpinäkyvää pintaa, kuten lasia. DI-tekniikkaa käytettäessä kosketuspintaa on vaikea saada valaistua tasaisesti infrapunavalolla, joten eri näytön kohdat

saattavat reagoida huonommin kosketukseen kuin toiset. DI-tekniikkaa käyttävillä näytöillä myös virheellisten kosketuspisteiden todennäköisyys on suurempi. (NuiGroup Authors 2009.)

Tällä hetkellä yksi optisten kosketuspöytien heikkouksista on niiden suuri kotelointi. Koska tekniikka vaatii kameran, täytyy kosketuspinnan takana olla paljon tilaa kameran sijoittelua varten. Optiset tekniikat ovat myös ongelmallisia tiloissa joissa valaistus muuttuu usein. (Han 2005.)

2.3 Kapasitiivinen ja resistiivinen monikosketustekniikka

Kuluttajien saatavilla olevissa laitteissa käytetään yleensä kapasitiivista tai resistiivistä kosketustekniikkaa. Kumpaakin tekniikkaa käytetään erityisesti puhelinten sekä muiden pieniä näyttöjä sisältävien laitteiden kanssa. (Wikipedia 2010a.)

Kapasitiivinen tekniikka perustuu kapasitanssiin, eli kosketusnäytön sähköisesti varautuneiden osien potentiaalieroon. Kapasitiivinen kosketusnäyttö rakentuu useista tasoista, jotka on sijoitettu LCD-näytön päälle. Näytössä on kaksi tasoa, joihin etsatut samansuuntaiset johdot muodostavat sähköisesti varatun ruudukon. Käyttäjä koskettaessa näyttöä sormellaan aiheuttaa se muutoksen kosketusnäytön sisältämään sähköiseen kenttään. Tieto tästä muutoksesta lähetetään ohjelmistolle, joka laskee kosketuspisteen paikan. (Wilson 2010; Baxter 1996.)

Resistiivistä monikosketustekniikkaa ei vielä ole kuluttajien saatavilla olevissa laitteissa. Resistiivinen kosketustekniikka on kuitenkin hyvin yleistä tavallisissa yhtä kosketuspistettä kerrallaan tukevissa kosketusnäytöissä (Wikipedia 2010b). Kapasitiivisen näytön tapaan myös resistiivinen kosketusnäyttö rakentuu tasoista, jotka on sijoitettu LCD-näytön päälle. Kosketuspisteen tunnistamiseen resistiivisessä näytössä käytetään kahta joustavaa tasoa, joista toisessa on sähkövirta. Tasojen välissä on tyhjä rako, ja käyttäjän painaessa näyttöä koskettavat tasot toisiaan, jolloin ne muodostavat suljetun sähköpiirin. (Helander, Landauer & Prabhu 1997).

Ainakin ranskalainen Stantum on kuitenkin rakentanut monikosketusnäyttöjä resistiivisellä tekniikalla. Stantumin näytöt tukevat rajatonta määrää kosketuspisteitä ja sormia käyttäessä tunnistavat myös paineen. (Miller, R. 2009.)

Kapasitiivisen ja resistiivisen tekniikan välillä on muutamia eroja. Johtuen kapasitiivisten näyttöjen tekniikasta voi niitä käyttää vain sähköä johtavilla instrumenteilla, kuten sormilla. Resistiivisiä näyttöjä voi sen sijaan käyttää myös kynää käyttäen tai käsineet kädessä. Tämä mahdollistaa tarkemman käytön ja tuntuu luonnolliselta esimerkiksi piirtosovelluksissa. Kapasitiivisten näyttöjen käyttö taas ei vaadi näytön painamista, vaan kevyt kosketus riittää. Toisaalta tämä tarkoittaa sitä, että kapasitiiviset näytöt eivät tunnista näyttöön kohdistunutta painetta toisin kuin resistiiviset. Resistiivisen tekniikan etuna ovat myös alhaiset valmistuskustannukset. (UMCPortal 2009.)

2.4 Yleisimmät laitetyypit

Monikosketusta hyödyntäviä laitteita on nykyään huomattava määrä myös kuluttajamarkkinoilla. Suurin osa näistä laitteista on puhelimia tai älypuhelimia. Ensimmäinen kosketusnäytöllinen puhelin tehtiin jo 90-luvun alussa ja kosketusnäytöt olivat käytössä mm. PDA-laitteissa. Kosketusnäytöt ja erityisesti monikosketusnäytöt tulivat laajasti puhelimiin vasta vuoden 2000-luvun ensimmäisen vuosikymmenen jälkipuoliskolla. Toinen näkyvä monikosketuslaitekategoria ovat suuret monikosketuspöydät ja -seinät. Niiden suosio on kasvanut erityisesti Jeff Hanin esiteltyä FTIR-tekniikan, ja Microsoftin tuotua markkinoille Surface monikosketuspöydän. (Buxton 2009a.)

Monikosketusnäyttöjen käyttäminen työpöytäkoneissa ei ole saavuttanut suurta suosiota monista käyttöön ja käytettävyyteen liittyvistä syistä. Kosketus on interaktiotekniikkana epätarkka, eikä se sovellu vaativaan työkäyttöön kovin hyvin. (Miller, C. 2009).

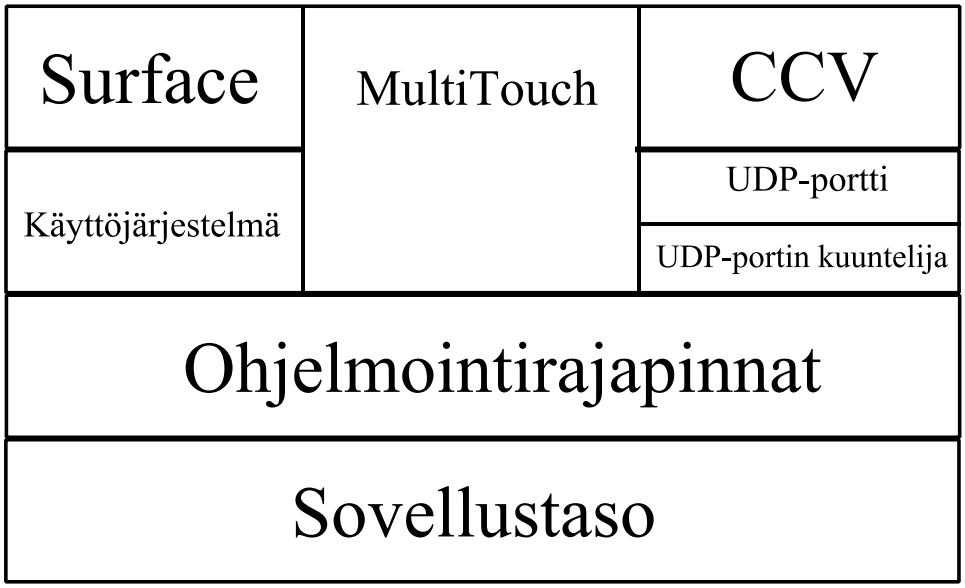
Monikosketustekniikkaa sisältäviä kannettavia tietokoneita ovat julkaisseet mm. Dell, Hewlett-Packard, Lenovo ja Acer. Niiden suosio on kuitenkin ollut vaatimatonta. (Wikipedia 2010a.)

3 TAUSTAJÄRJESTELMÄT JA OHJELMISTOARKKITEHTUURI

Optisissa monikosketustekniikoissa kosketuspisteiden tunnistus tapahtuu analysoimalla laitteiston kameran antamaa kuvaa. Kuvan analysointi on paljon prosessointitehoa vaativa toimenpide, joka tehdään sitä varten suunnitellussa taustajärjestelmässä. Taustajärjestelmässä kuvaa muokataan erilaisilla suotimilla, jonka jälkeen kaikki pisteistä saatu data on sovellusten käytettävissä. Kaupallisten optisten monikosketusnäyttöjen tekijöillä, kuten Microsoftilla ja MultiTouchilla, on omat taustajärjestelmänsä. Näiden lisäksi saatavilla on useita avoimen lähdekoodin taustajärjestelmiä.

3.1 Optisten näyttöjen taustajärjestelmät

Taustajärjestelmät vastaanottavat kameran tai kameroiden kuvaa ja käyttävät erilaisia suotimia erotellakseen kuvasta vain kirkkaimmat kohteet, käyttäjän sormet, joista infrapunavalo heijastuu. Pisteiden tunnistamiseksi kuvaa käsitellään erilaisilla suotimilla, kuten häiriönpoisto- ja taustanpoistosuotimilla. (Han 2009.) Tämän jälkeen taustajärjestelmä antaa tunnistetuille pisteille omat tunnisteet ja alkaa lähettää tietoa niistä eri menetelmillä riippuen taustajärjestelmästä.

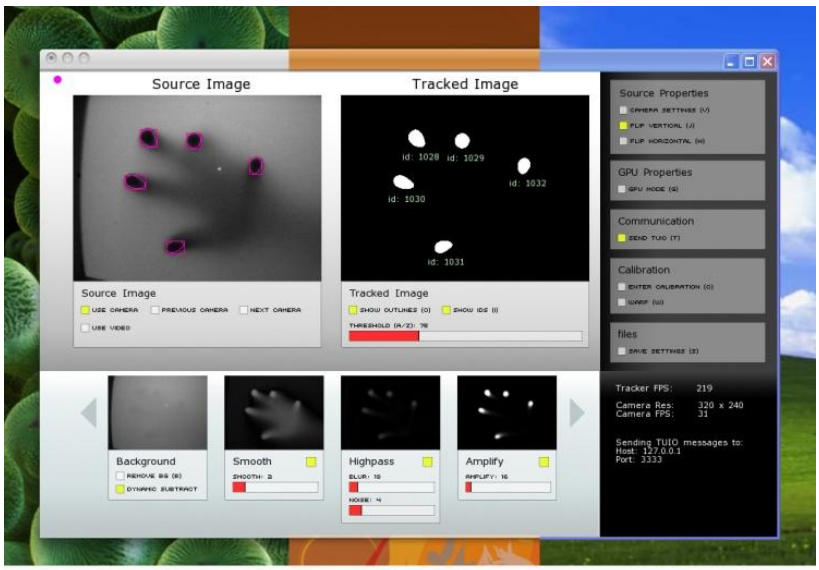


KUVIO 4. Optisten näyttöjen taustajärjestelmien arkkitehtuurit.

Kuviosta 4 näkyy kolmen eri taustajärjestelmän arkkitehtuurit. Microsoft Surface toimii samaan tapaan kuin mikä tahansa tietokoneen lisälaite. Taustajärjestelmä on eräänlainen ajuri, joka lähettää viestit kosketuspisteistä käyttöjärjestelmälle, ja sovelluskehittäjä voi tehdä monikosketussovelluksia Windowsin ohjelmointirajapintojen avulla. Windowsin ohjemointirajapintoja käytettäessä sovellus toimii vain Microsoftin Surfacella, ja niillä monikosketuslaitteilla, joille on Windows-ajurit. Surfacen tapauksessa monikosketus on selvästi integroituna käyttöjärjestelmään verrattuna muihin järjestelmiin. (Microsoft 2010.)

MultiTouchin ja avoimen lähdekoodin Touchlib järjestelmät toimivat siten, että sovelluskehittäjä käyttää MultiTouchin tai Touchlibin ohjelmointirajapintoja, jotka liittävät analysointijärjestelmän sovelluskehittäjän ohjelmaan. Analysointijärjestelmä ajetaan samalla, kun sovelluskehittäjän oma ohjelma käynnistetään. Tällöin kosketusdata tulee suoraan analysointia tekevältä taustaprosessilta. Tässä tapauksessa on tarkoitus, että vain yksi monikosketussovellus on auki kerrallaan, sillä jokainen niistä käynnistää oman analysointiprosessinsa. MultiTouchin taustajärjestelmä pystyy sormien lisäksi tunnistamaan käyttäjän kädet ja sen mihin käteen mikäkin sormi kuuluu. (MultiTouch 2010.)

Community Core Vision(CCV) on avoimen lähdekoodin taustajärjestelmistä suosituin. CCV tukee kosketusdatan lähetystä epäsuorasti Open Sound Control (OSC) -muodossa, eXtensible Markup Langueage (XML) -muodossa ja Tangible User Interface Object (TUIO) -signaaleina. TUIO-muotoisia viestejä käsitellään tarkemmin seuraavassa luvussa. CCV on saatavissa Windowsille, Mac OS X:lle ja Linuxille. CCV tunnistaa kosketuspisteiden paikan ja koon sekä laskee niiden nopeuden ja kiihtyvyyden. Viestien lähetys tehdään yleensä johonkin tiettyyn verkkoporttiin, jota ohjelmointirajapinnat tai loppukäyttäjän käyttämä sovellus kuuntelee. (NuiGroup 2010.)



KUVIO 5. CCV tunnistaa ja näyttää kosketuspisteet reaaliaikaisesti.

Tässä järjestelmässä kuvan analysointi voidaan erottaa omalle tietokoneelleen, jolloin tehoja säästyy monikosketussovellusten käyttöön. CCV:n etuna on myös se, että sen kanssa on helppo käyttää monia eri ohjelmointirajapintoja ja -kieliä, koska taustajärjestelmä on täysin erotettu ohjelmointirajapinnoista. (Kaltenbrunner 2005.) Sovelluskehittäjän näkökulmasta kosketusdata on siis saatavissa kahdella eri tavalla: joko suoraan järjestelmän ajureilta, sovelluksen taustalla ajettavilta analysointiprosesseilta tai epäsuorasti jonkin verkkoportin kautta lähetettävinä viesteinä.

3.2 Tangible User Interface Object -protokolla

Tangible User Interface Object (TUIO) -protokolla on Pompeu Fabran yliopistossa Barcelonassa kehitetty protokolla kosketuspisteiden tiedon lähettämiseen pisteitä tunnistavalta taustajärjestelmältä monikosketussovelluksille. TUIO-protokollan tavoite on tarjota yleinen ja joustava kommunikaatiorajapinta kosketuspisteitä tunnistavan taustajärjestelmän ja monikosketussovellusten välille. Kosketuspisteiden, yleensä sormien, lisäksi TUIO-protokollalla voi välittää tietoa pöydälle laitetuista pöydän kanssa interaktioon tarkoitetuista esineistä. (Kaltenbrunner 2005.)

TUIO-signaalit lähetetään yleensä Open Sound Control (OSC) -muodossa User Datagram Protocol (UDP) paketteina verkkoporttiin 3333. UDP:n käyttö mahdollistaa pienen viiveen, sillä UDP-protokollassa ei käytetä pakettien virheentarkistusta, vaan pyritään suureen toimitusnopeuteen. OSC on MIDI:n korvaajaksi suunniteltu joustava viestimuoto, joka on suunniteltu tuomaan modernien verkkoteknologioiden hyödyt elektronisten musiikkilaitteistojen käyttöön. UDP:n ja OSC:n käyttö mahdollistaa joustavuuden ja nopeuden, johon esimerkiksi XML-muotoisten viestien lähettämisessä ei pystytä. (Wright 1997.)

TUIO-protokollalle on olemassa kolme spesifikaatiota, versio 1.0, versio 1.1 ja versio 2.0. Version 2.0 spesifikaatiota ei ole vielä viimeistelty, joten 1.1 on uusin käytössä oleva spesifikaatio. TUIO-protokolla määrittelee kaksi pääviestiä: SET-viestit sekä ALIVE-viestit. SET-viestit sisältävät tietoa kosketuspisteiden paikasta, nopeudesta ja orientaatiosta. ALIVE-viestit sisältävät listan kaikista tällä hetkellä kosketuspinnalla olevien kosketuspisteiden tunnuksista. Näiden lisäksi protokolla sisältää FSEQ-viestejä, joiden tarkoitus on ryhmitellä SET- ja ALIVE-viestit ajallistesti oikein. Kuviossa 6 on esitettynä TUIO-viestien muoto ja esimerkkejä kaikista viesteistä. (Kaltenbrunner 2005.)

```
TUIO-viestien muoto:
/tuio/2Dcur alive id0 ... idn
/tuio/2Dcur set id x y x_nopeus y_nopeus kiihtyvyys
/tuio/2Dcur fseq id

Esimerkki TUIO-viesteistä:
/tuio/2Dcur alive 23 24
/tuio/2Dcur set 23 102 453 3 -6 2.403
/tuio/2Dcur fseq 256
```

KUVIO 6. TUIO-signaalien muoto.

Kuvion 6 esittämässä tilanteessa käyttäjällä on kaksi sormea näytöllä. Näitä sormia vastaaville kosketuspisteille on taustajärjestelmä antanut tunnuksiksi numerot 23 ja 24. Ensimmäisenä on ALIVE-viesti, jossa lähetetään tieto näytöllä olevista sormista näiden tunnusnumeroitten avulla. Käyttäjän siirtäessä toista sormeaan taustajärjestelmä lähettää SET-viestin. SET-viesti sisältää kosketuspisteen tunnuksen, x-koordinaatin, y-koordinaatin, x-nopeuden, y-nopeuden ja kiihtyvyyden. Käyttäjä on tässä tapauksessa siis siirtänyt toista sormeaan pisteeseen (102,

453) 3-pikseliä oikealle ja 6-pikseliä ylöspäin kiihtyvyydellä 2.403. Viimeisenä esimerkissä on FSEQ-viesti, joka kertoo, että tämä on kahdessadasviideskymmeneskuudes paketti, joka on taustajärjestelmästä lähetetty tämän istunnon aikana.

TUIO-protokollan avulla voidaan ohjelmistoista tehdä käyttöjärjestelmästä, kosketuspöydän tekniikasta ja taustajärjestelmästä riippumattomia. TUIO-signaalit toimivat tällöin laitteiston ja ohjelmiston välisenä abstraktiona. (Kaltenbrunner 2009.)

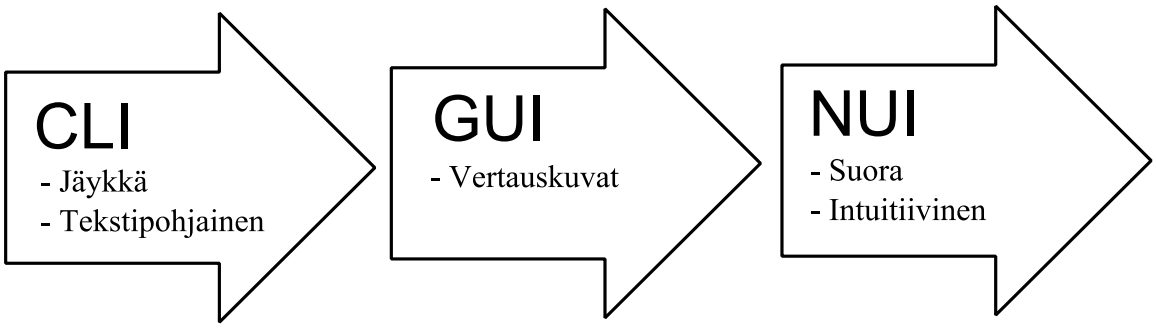
4 MONIKOSKETUSKÄYTTÖLIITTYMÄT

”Käyttöliittymien varsinainen ongelma on se että ne ovat olemassa. Käyttöliittymät ovat vain tiellä. En halua käyttää energiaa käyttöliittymään vaan haluan keskittyä työhön” (Norman 1990.). Hiiren keksimisestä kesti noin 30 vuotta sen käyttöön pohjautuvien graafisten käyttöliittymien popularisointiin, Microsoftin julkaistessa Windows 95 -käyttöjärjestelmän. Hiiri toi mukanaan uuden vuorovaikutustavan ja graafisen käyttöliittymän(GUI). Monikosketus, ääniohjaus ja eleet tuovat mukanaan luonnollisen käyttöliittymän(NUI). Käyttöliittymänkehitys on vielä monin tavoin alkuvaiheessa tekniikan ollessa vielä vaikeasti saatavilla. (Buxton 2009b.)

4.1 Natural User Interface

Natural User Interface (NUI) tarkoittaa ihminen-tietokone-vuorovaikutuksen ajattelumallia, joka keskittyy kehittämään vuorovaikutteisuutta ihmiselle luontaiseen suuntaan, ja käyttämään ihmisen ominaisuuksia kuten kosketusta, näköä, puhetta, liikettä ja jopa ajattelua sekä tietoisuutta vuorovaikutteisuuden lähtökohtina. Tavoitteena on tarkempi ja optimoidumpi vuorovaikutteisuus fyysisen ja digitaalisen todellisuuden välillä. (NUIGroup 2010.)

Yleensä termillä NUI tarkoitetaan käyttöliittymää, joka on mahdollisimman helppokäyttöinen, saumaton, huomaamaton ja luonnollinen. (Raisamo 1999.)



KUVIO 7. Käyttöliittymien kehitys.

Kuviossa 7 on esitetty NUI:n suhde sen edeltäjiin. NUI:ta pidetään graafisen käyttöliittymän (GUI) seuraajana, samaan tapaan kuin GUI oli komentorivikäyttöliittymien (CLI) seuraaja. Siitä mitä NUI sisältää on monia käsityksiä. Verrattaessa aikaisempiin käyttöliittymiin NUI:n tavoitteena on olla suorempi ja luonnollisempi. (Seow, Wixon, MacKenzie, Jacucci, Morrison, & Wilson 2009).

Tavoitteena kuitenkin on tehdä ihminen-tietokone-vuorovaikutuksesta samankaltaista kuin ihminen-ihminen-vuorovaikutus sekä ihminen-ympäristö-vuorovaikutus. NUI-käyttöliittymiä voidaan ajatella digitaalisen muuttamisena fyysiseksi. Toinen seuraavan sukupolven käyttöliittymistä puhuttaessa esiin tuleva termi onkin Tangible User Interface (TUI) eli käsinkosketeltava käyttöliittymä. (NUIGroup 2010.)

Monikosketus on vain yksi osa NUI:ta ja monissa monikosketuskäyttöliittymissä ei NUI:n ideoita toteuteta. Monikosketuskäyttöliittymän tavoite kuitenkin tulisi olla fyysisen ja digitaalisen välisen abstraktion poistaminen, suoran manipulaation mahdollistaminen sekä objekti- ja kontekstipohjaisen käyttöliittymän tarjoaminen. Parhaiten monikosketuksen ominaisuuksia ja mahdollisuuksia hyödynnetään käyttöliittymissä, jotka on rakennettu sitä varten. Muutaman eleen lisääminen osaksi käyttöliittymää rajoitettuihin kohtiin ei vielä toteuta NUI:n periaatteita, eikä tuo käyttäjälle parempaa tai luonnollisempaa vuorovaikutustapaa, vaan pelkän käyttöliittymätempun. (NuiGroup Authors 2009.)

Pidemmälle vietyjä ajatuksia NUI:sta tarjoaa mm. Jakob Nielsen. Hän kirjoitti jo 1993 GUI:n seuraajan olevan ei-komentopohjainen käyttöliittymä, joka suoran vuorovaikutuksen ja komentojen vastaanottamisen sijaan tarkkailee ihmistä ja tekee päätöksiä tämän mukaan. Tulevaisuuden NUI-käyttöliittymät ympäröivät meitä joka paikkaan sijoitettujen sensorien ja tietokoneiden avulla. (Nielsen 1993.)

Käyttöliittymäajattelumallien vaihtuminen on läheisessä yhteydessä tietokoneiden tekniikkaan ja yleisyyteen. Komentorivikäyttöliittymien aikakautena yhtä tietokonetta käytti monta ihmistä, graafisten käyttöliittymien aikakautena yleistyivät henkilökohtaiset tietokoneet ja

seuraavan sukupolven käyttöliittymien ja läsnä-älyn aikakautena yhtä ihmistä varten on monia tietokoneita. (NUIGroup 2010.)

Nykyisiä graafisia käyttöliittymiä nimitetään joskus niiden peruselementtien mukaan WIMP-käyttöliittymiksi (windows, icons, menus, pointing devices). Nämä elementit ovat graafisten käyttöliittymien oleellisimpia osia. NUI-käyttöliittymien peruselementeiksi on ehdotettu objekteja, säiliöitä, eleitä ja manipulaatiota. NUI-käyttöliittymissä pyritään siis vuorovaikutuksen luonnollisuuden lisäksi poistamaan käyttöliittymistä kielikuvat, kuten ikkunat ja valikot. Ne korvataan data- ja objektilähtöisillä käsitteillä, jotka ovat lähempänä fyysistä maailmaa. (Blake 2009.)

4.2 Käyttöliittymän suunnittelu

Monikosketuskäyttöliittymien suunnitteluun ei ole vielä muodostunut standardeja tai yleisiä käytäntöjä, lukuunottamatta muutamia eleitä kuten useasti skaalaukseen käytetty nipistysele. Standardeja kuitenkin suunnitellaan yritysten ja tutkimuslaitosten toimesta koko ajan (Widgor, Fletcher & Morrison 2009).

Monet periaatteet, joita käytetään tavallisille kosketusnäytöille sovelluksia suunnitellessa pätevät myös monikosketussovellusten suunnittelussa. Etenkin pienten näyttöjen, kuten puhelimissa olevien kosketusnäyttöjen, tapauksessa eroavaisuudet ovat vähäisiä. Isommilla näytöillä, joilla on enemmän tilaa eleille, useamman sormen ja käden käytölle tai jopa usealle yhtäaikaiselle käyttäjälle, eroavaisuudet ovat kuitenkin huomattavia. (Saffer 2009.)

Tärkeimmät huomioitavat asiat monikosketuskäyttöliittymiä suunniteltaessa ovat selkeys, vuorovaikutteisten elementtien erottuminen, ikonien selkeä yhteys niihin liittyviin toimintoihin, eleiden löydettävyys ja johdonmukaisuus, käyttöliittymän vasteajat sekä selkeästi merkityt paluu-, lopetus- ja peruutustoiminnot. Suunniteltaessa sovelluksia, joissa käyttäjät viipyvät vain hetken aikaa, täytyy käyttöliittymästä tehdä visuaalisesti puoleensavetävä ja kontrollien käyttötavoista selkeitä. Tällaisia sovelluksia ovat esimerkiksi visuaaliset demonstraatiot sekä informaatio- ja opastussovellukset. (Cooper & Reimann 2003.)

Käyttöliittymien tulisi olla mahdollisimman yksinkertaistettuja, koska se vähentää käyttäjän muistitaakkaa ja väärinymmärryksiä. Parhaassa tapauksessa käyttäjälle näytetään vain se informaatio, jota hän tarvitsee, eikä yhtään enempää. (Nielsen 1993.)

Isoillekin monikosketusnäytöille käyttöliittymiä suunniteltaessa täytyy ottaa huomioon käyttöliittymän selkeys ja yksinkertaisuus. Turhia elementtejä, eli elementtejä, jotka eivät ole oleellisia tämän hetkisten toimintojen kannalta, tulisi välttää. Ne vievät käyttäjän huomion tärkeistä elementeistä, jotka välittävät tietoa käyttöliittymästä, sen toiminnasta ja tilasta. (Cooper & Reimann 2003.)

Tilan käytöllä on suuri merkitys käyttöliittymän selkeyden kannalta. Toisiinsa liittyvät elementit ja kontrollit tulee visuaalisesti ryhmittää sijoittamalla ne lähekkäin, antamalla niille yhtenäinen visuaalinen ilme tai erottamalla ne muusta käyttöliittymästä siihen tarkoitetuilla elementeillä. (Nielsen 1993). Selkeyttä edistävät myös maltillinen värien käyttö, niin eri värien määrän kuin värikylläisyydenkin suhteen, tyhjän tilan jättäminen kontrollien ympärille ja yhtenäinen fontti. (Cooper & Reimann 2003.)

Vuorovaikutteisten elementtien tulisi erottua sovelluksen taustasta ja ei-vuorovaikutteisista elementeistä selkeästi ja muistettavasti. Tämä voidaan saavuttaa esimerkiksi ulottuvuus-, väri- tai tilakontrastin avulla. (Cooper & Reimann 2003.)

Vuorovaikutteisten elementtien erottumisen lisäksi täytyy elementtien sisältää visuaalisia vihjeitä siitä, miten niiden kanssa ollaan vuorovaikutuksessa. Näitä visuaalisia vihjeitä kutsutaan affordansseiksi. Affordansseilla voidaan välittää käyttäjälle jotain tietoa siitä miten elementtiä tulisi koskea. Esimerkiksi painettavan napin tapauksessa liukuvärin tai kolmiulotteista muotoa viestivän reunuksen lisääminen nappiin auttaa käyttäjää erottamaan sen ei-vuorovaikutteisista elementeistä. Affordansseja ei välttämättä tarvita, jos kontekstista käy selväksi miten elementtien kanssa ollaan vuorovaikutuksessa. Napeista puhuttaessa näin voidaan ajatella olevan esimerkiksi virtuaalisen näppäimistön tapauksessa. (Norman 1990.)

Kosketuskäyttöliittymissä tulisi ottaa huomioon myös näyttöjen epätarkkuus. Elementtien tulisi olla sormilla käytettäväksi tarpeeksi suuria sekä ottaa huomioon sormien epätarkkuus verrattuna osoitinlaitteisiin. Isoilla näytöillä käyttäjän kädet ja käsivarret ovat suuren osan ajasta käyttäjän edessä ja estävät käyttäjää näkemästä koko käyttöliittymää kerralla. Tästä aiheutuvia ongelmia voidaan välttää sijoittamalla tärkeät elementit alueille joita käyttäjän kädet eivät yleensä peitä. (Albinsson & Zhai 2003.)

Jotta käyttäjä tuntisi olonsa turvallisemmaksi ja vapaammaksi käyttäessään sovellusta, olisi lopetus-, paluu- ja peruutanappien oltava selvästi merkittyjä. Näin käyttäjä pääsee aina takaisin alkuun eikä hänen tarvitse pelätä ”rikkovansa” ohjelmaa. (Nielsen 1993.)

4.3 Eleet

Laajasti määritettynä käsitteellä ele tarkoitetaan ihmisen tekemää liikettä, jonka tarkoitus on viestiä jotain toiselle ihmiselle. Monikosketuksen kontekstissa eleellä tarkoitetaan yhden tai useamman sormen liikuttamista kosketuspinnalla niin että liike on tulkittavissa joksikin ennalta määritetyksi viestiksi. (Nespolous, Perron & Levours 1986.)

Monikosketuseleet mahdollistavat paljon vapaamman ja laajemman tavan kommunikoida kuin hiirellä tai tavallisella kosketusnäytöllä tehtävät eleet. Eleet voivat olla yksinkertaisia symboleja, jotka suorittavat jonkin toiminnon, tai niillä voidaan määrittää jonkin toiminnon parametrit. (Kourakis 2004.)

4.3.1 Eleiden jaottelu

Monikosketuseleiden jaotteluun ja kategorisointiin ei ole muodostunut vielä mitään selvää standardia. Monikosketuspinnan ja käyttäjän vuorovaikutuksesta on kuitenkin eroteltavissa kaksi vuorovaikutustapaa, epäsuorat ja symboliset eleet sekä suora manipulaatio. Suoraa

manipulaatiota ei välttämättä luokitella eleeksi, mutta yleisesti joistain suoran manipulaation tekniikoista puhutaan eleinä. (George 2009.)

Tässä työssä eleet on jaettu manipulaatioeleisiin ja symbolisiin eleisiin. Manipulaatioeleet ovat usein luonnollisen tuntuisia eleitä, joilla muokataan sovelluksen tilaa tai objekteja. Hyviä esimerkkejä manipulaatioeleistä ovat kahden sormen avulla skaalaus, pyöritys ja siirto. (Nardi 2008, 11.)

Symboliset eleet ovat eleitä joissa käyttäjän tekemät liikkeet muodostavat symbolin tai kuvion, jonka tunnistaessaan ohjelmisto suorittaa kyseiseen symboliin liitetyn toiminnon. Symbolisia eleitä voidaan käyttää erilaisten valikkojen avaamiseen, tai esimerkiksi toimintojen hyväksymiseen tai hylkäämiseen. Symboliset eleet eivät välttämättä vaikuta sovelluksen toimintaan ennen kuin ele on suoritettu loppuun jolloin ohjelmisto sen tunnistaa. (Westerman, Elias & Hedge 2001.)

4.3.2 Eleet osana käyttöliittymää

Aikaisemmin graafisten sovellusten vuorovaikutteisuus on rajoittunut yksinkertaiseen hiirellä osoittamiseen, nappien painamiseen ja raahaamiseen. Monikosketuseleet kuitenkin mahdollistavat luonnollisemman ja monipuolisemman tavan olla vuorovaikutuksessa sovellusten kanssa. (Buxton 2009b.)

Hyvin suunnitellut eleet mahdollistavat sulavan ja joustavan vuorovaikutuksen erityisesti isoilla kosketusnäytöillä, joilla näppäimistön ja hiiren käyttäminen ei ole käytännöllistä. Elepohjaisia käyttöliittymien suunnittelu on kuitenkin haastavaa. Eleiden täytyy olla helposti opittavia, toisistaan erottuvia ja kontekstiin sopivia. (Wu, Shen, Ryall, Forlines & Balakrishnan 2006.)

Elettä tehdessään käyttäjän tulisi saada välitöntä visuaalista palautetta toimistaan. Manipulaatioeleiden tapauksessa tämä yksinkertaisesti tarkoittaa sitä, että objektien manipulaatio tehdään reaaliaikaisesti. Symbolisten eleiden tapauksessa mahdollisuus antaa käyttäjälle visuaalista pa-

lautetta riippuu siitä, miten symbolisten eleiden tunnistus on toteutettu. Jotkin järjestelmät yrittävät tunnistaa eleen heti kun käyttäjä asettaa sormensa kosketuspinnalle ja jotkin vasta sen jälkeen kun ele on suoritettu loppuun. (Wu ym. 2006.)

Eleen monimutkaisuuden tulisi vastata sen avulla suoritettavan toiminnon monimutkaisuutta. Tämä ei tarkoita sitä, että monimutkaiset toiminnot tarvitsivat monimutkaisemmat eleet, vaan sitä, että yksikertaisimmat eleet tulisi olla varattuna yksinkertaisille toiminnoille, joita käytetään eniten. Hyvin suunnitellussa järjestelmässä eleet vastaavat niitä eleitä, joita käyttäjä saattaisi muutenkin käyttää kyseistä toimintoa tehdessään. (Saffer 2009.)

Eleiden sisällyttäminen käyttöliittymään tuo mukanaan myös ongelmia. Yksi haaste elepohjaista käyttöliittymää tehtäessä on eleiden löydettävyys ja se, että kosketettavista käyttöliittymäelementeistä ei usein näe, kuinka niiden kanssa ollaan vuorovaikutuksessa. (Saffer 2009.)

4.4 Näyttö tietokoneen hallintalaitteena

Sormien ja käsien käyttäminen on ihmiselle luonnollinen tapa hallita ympäristöään. Kosketusnäyttöjen etuna onkin niiden luonnollisuus. Ne ovat välitön tapa kommunikoida tietokoneiden ja niiden käyttöliittymien kanssa lähes samoin kuin tavallistenkin esineiden tai laitteiden kanssa. (Kim, Park, Kim & Lee 2007.) Sen sijaan, että käyttäjä käyttäisi osoitinlaitetta, käyttäjä voi olla käyttöliittymän kanssa suoraan kosketuksissa (Ka 2008).

Monikosketuskäyttöliittymistä on mahdollista tehdä niin intuitiivisia ja luonnollisia, että ohjeistusta niiden käyttöön ei tarvita. Käyttöliittymä voi esimerkiksi muistuttaa papereiden tai esineiden liikuttelua pöydällä. Monikosketusnäytöt ja -pöydät mahdollistavat sovellukset joita voi käyttää useampi ihminen yhtä aikaa. Tämä pätee erityisesti suuriin monikosketusnäyttöihin. (Thörnlund 2007.)

Monikosketuskäyttöliittymiin ei ole vielä muotoutunut selviä standardeja tai käytäntöjä siitä, miten käyttöliittymän tulisi toimia ja millaisilla eleillä käyttöliittymiä käytetään. Eleiden standardointia tapahtuu kuitenkin koko ajan monikosketustekniikan kehittyessä ja käytäntöjen vakiintuessa.

Yksi käyttöliittymän ongelmallisista kohdista ovat eleet. Monikosketuseleet ovat hyvä keino hallita visuaalista sovellusta luonnollisella tavalla, mutta niiden sisällyttäminen käyttöliittymään tuo mukanaan myös ongelmia. Useimmat eleet eivät ole luonnollisia eivätkä helposti muistettavia. Monimutkaisia eleitä on vaikea ymmärtää ennen kuin on nähnyt jonkun muun niitä käyttävän. Jos sovelluksessa ei ole näkyvää ohjeistusta, ovat eleet näkymätön osa käyttöliittymää, ja ne on selitettävä käyttäjälle etukäteen. Usein käyttäjä ei saa eleistä minkäänlaista visuaalista palautetta, jolloin ei voi olla varma, tekikö käyttäjä eleen oikein. (Norman 2010.)

Koska kosketusnäyttöjen hallinnassa käytetään aktiivisesti käsiä, vaikuttaa se siihen minkälaisiksi näytöt pitäisi rakentaa. 1980-luvulla tuli tunnetuksi käsite Gorillakäsivarsi pystysuuntaisten kosketusnäyttöjen yleistyessä. Jatkuvassa käytössä käyttäjien kädet väsyivät ja kosketusnäyttöjen käyttö oli työlästä. (Wikipedia 2010a.)

Yksi suurista ongelmista monikosketusnäyttöjen käyttämisessä työpöytäkoneissa onkin ergonomia. Riippuen näytön asettelusta käyttäjät joutuvat joko pitämään käsiään ylhäällä tai katsomaan alaspäin näin rasittaen niskaansa. (Miller, C. 2009.)

Hallintalaitteet, jotka eivät perustu kytkimiin, näppäimistöihin ja hiiriin, ovat kuitenkin hyvin tervetullut lisä tietokoneiden hallintalaitteisiin. Luonnolliset käyttöliittymät sopivat erityisesti tilanteisiin, joissa perinteiset hallintalaitteet ovat epäkäytännöllisiä. Monikosketusnäytöt ja muutkaan luonnolliset käyttöliittymät eivät kuitenkaan ole ongelmattomia ratkaisuja, vaan niissä on omat haasteensa ja heikkoutensa. Monikosketusta ei kannatakaan ajatella perinteisten vuorovaikutustekniikoiden korvaajana vaan uutena mahdollisuutena. Ajan myötä tekniikoiden ja käytäntöjen kehittyessä monikosketusnäytöt löytävä oman paikkansa. (Norman 2010.)

5 OHJELMOINTIRAJAPINTOJEN SUUNNITTELU

Koska opinnäytetyön tuloksena on ohjelmointikirjasto, käsitellään tässä luvussa ohjelmointikirjastojen ja -rajapintojen suunnittelua. Tavoitteena työssä oli kehittää ohjelmointikirjasto, joka on helppokäyttöinen, helposti laajennettavissa ja minimaalinen.

Ohjelmointirajapinnan ja ohjelmoijan suhde on verrattavissa käyttäjän ja graafisen käyttöliittymän suhteeseen. Rajapinnat suunnitellaan ohjelmoijaa eikä ohjelmaa varten. Tärkeimpiä huomioitavia asioita kirjastoja ja rajapintoja suunniteltaessa on kirjaston komponenttien keskinäinen riippumattomuus ja komponenttien uudelleenkäytettävyys. Nokian Qt-kehitysympäristön kehittäjien mukaan hyvällä rajapinta on minimaalinen, toiminnallisuudeltaan valmis, yksinkertainen, intuitiivinen, muistettava ja rajapinnan käyttäminen johtaa helposti ymmärrettävään ohjelmakoodiin. (Ettrich 2005).

Minimaalisuus tarkoittaa sitä, että rajapinnan luokilla on vain muutama julkinen jäsen, ja rajapinta sisältää mahdollisimman vähän luokkia. Tämä tekee ohjelmointirajapinnasta helpommin ymmärrettävän ja muistettavan. Muistettavuutta lisäävät myös johdonmukaiset nimeämiskäytännöt. (Blanchette 2008).

Toiminnallisuudeltaan valmis rajapinta tarjoaa sen toiminnallisuuden, jonka siltä odottaisikin. Vähintään tulisi kuitenkin tarjota mahdollisuus rajapinnan laajentamiseen ja lisätoimintojen tekemiseen periytyvyyden kautta. (Blanchette 2008).

Yksinkertaisuudella tarkoitetaan rajapinnan käytön yksinkertaisuutta. Usein käytettyjen toimintojen käyttäminen tulisi olla mahdollisimman helppoa. On parempi keskittyä tarkkaan määriteltyjen ongelmien ratkaisemiseen kuin yrittää tehdä yleisratkaisuja, sillä niiden käyttäminen on hankalaa. (Ettrich 2005).

Ohjelmointirajapintaa voidaan pitää intuitiivisena, jos keskitason kokemuksen omaava ohjelmoija pystyy käyttämään sitä lukematta ohjeistusta. Ohjelmointirajapinnan käytön tulisi

johtaa helposti luettavaan ohjelmakoodiin. Tämä tarkoittaa sitä, että rajapinta tulisi suunnitella siten, että se ohjaa ohjelmoijan tekemään ratkaisuja, jotka myöhemmin ohjelmakoodia luettaessa tekevät ohjelmakoodista helposti ymmärrettävän. (Ettrich 2005.)

5.1 Komponenttien suunnittelu

Komponentit ovat ohjelmistoarkkitehtuurien pääyksiköitä. Ne ovat itsenäisiä ohjelmistoyksiköitä, jotka tarjoavat palvelujaan hyvin määritettyjen rajapintojen kautta. Ne voivat olla pieniä, muutamia yksinkertaisia palveluja tarjoavia ohjelmayksikköjä, mutta ne voivat olla myös sovelluksen kokoisia yksiköitä. Pääsääntöisesti komponentin tulisi olla sen kokoinen, että se on yhden henkilön hallittavissa. (Koskimies & Mikkonen 2005.)

Komponentin ideaan kuuluu, että sen sisäinen rakenne ei näy käyttäjille eikä riipu käyttöympäristöstä. Yksittäisen komponentin arkkitehtuurin ei pitäisi olla sidoksissa sitä käyttävän järjestelmän arkkitehtuuriin. Tällä pyritään vähentämään komponentin riippuvuutta muista komponenteista, ja ohjelmistojen riippuvuutta joistain tietyistä komponenteista. (Koskimies & Mikkonen 2005.)

Komponenttien tulisi olla itsenäisiä, suljettuja ja niillä tulisi olla yksi hyvin määritelty tehtävä. Jos komponentit on erotettu toisistaan hyvin, voi niitä muuttaa myöhemmin huolehtimatta siitä miten se vaikuttaa muihin komponentteihin. Jos komponenteilla on hyvin määritetyt tehtävät ja vastuut niitä voidaan myöhemmin käyttää uudelleen ja yhdistellä tavoilla joihin niitä ei ehkä alun perin tarkoitettu. (Hunt & Thomas 2009.)

Komponenteilla voidaan myös hallita järjestelmän muunneltavuutta. Tällöin komponentti esimerkiksi eristää järjestelmän jonkin tietyn osan, joka halutaan helposti vaihdettavaksi, lisättäväksi tai poistettavaksi. Yleensä järjestelmää ei kuitenkaan ole mahdollista jakaa komponentteihin siten, että kukin toiminnallisuus on täsmälleen yhdessä komponentissa. Tämä johtuu siitä, että järjestelmän rakennetta voidaan tarkastella erilaisten huolenaiheiden kannalta. Järjestelmän ja komponenttien suunnittelu onkin usein kompromissien tekemistä haluttujen

toiminnallisuuksien ja komponenttien uudelleenkäytettävyyden sekä muunneltavuuden välillä. (Koskimies & Mikkonen 2005.)

Kun komponenteista tehdään toisistaan riippumattomia, ja koko järjestelmä suunnitellaan ortogonaaliseksi eli kohtisouraksi, seuraa se, että järjestelmää on helpompi kehittää. Tämä johtuu siitä, että mahdolliset ongelmat on helpompi selvittää, kun komponentit eivät aiheuta sivuvaikutuksia toistensa toimintaa. Ortogonaalista järjestelmää on myös helpompi testata ja dokumentoida. (Hunt & Thomas 2009.)

5.2 Komponenttien rajapinnat

Rajapinnat ovat keskeinen osa ohjelmistoarkkitehtuuria. Ne määrittelevät miten komponentteja käytetään ja miten komponentit kommunikoivat keskenään. Komponenteilla voi olla sekä tarjottuja että vaadittuja rajapintoja. Komponentti voi siis tarjota ohjelmoijalle tai toiselle komponentille jonkin rajapinnan, jolloin se toteuttaa rajapinnan mukaisia palveluja, tai se voi vaatia palveluja joltain toiselta komponentilta. (Koskimies & Mikkonen 2005.)

Komponenttien tilaa ja toimintaa voidaan muuttaa rajapintojen kautta. Tämä voidaan mahdollistaa esimerkiksi asetusoperaatioiden (setter) avulla. Settereillä asetetaan arvoja komponentin muuttujille ja täten vaikutetaan komponentin toimintaan. Tietoa muuttujien arvoista saadaan hakuoperaatioiden (getter) avulla. Yleensä getterit ja setterit ovat luokkafunktioita, jotka piilottavat luokan muuttujien arvojen muokkaamiseen käytettävän logiikan. Getterit ja setterit ovat helppo tapa räätälöidä komponentteja ja niitä käytetään paljon, juuri siksi niiden nimeämiseen tulee kiinnittää erityishuomiota. (Koskimies & Mikkonen 2005.)

Sen, miten toiminnot on komponenteissa toteutettu, ei tulisi vaikuttaa rajapintaan. Toteutus tulee piilottaa siten, että sitä voidaan myöhemmin muuttaa ilman, että rajapintaa käyttävää ohjelmakoodia tarvitsee muuttaa. Luokkien ja niiden muuttujien pitäisi olla mahdillisimman suljettuja, siten että niiden muuttujat eivät näy ulospäin. (Bloch 2006.)

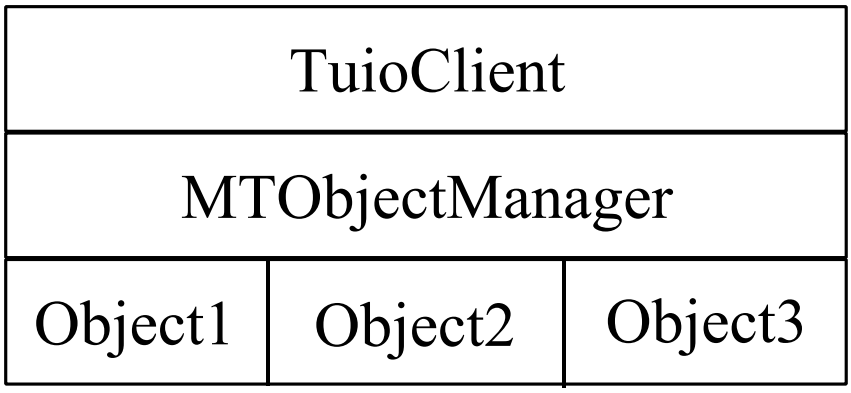
C++-kielen tapauksessa toinen, paljon monipuolisempi tapa räätälöidä komponentteja on periytyminen. Komponentin luokan perii joku toinen luokka, joka määrittelee uudelleen jotkin sen operaatiot. Näin syntyy uusi komponentti, jolla on samat tarjotut rajapinnat, mutta muokattu toteutus. Abstraktien luokkien avulla voidaan luoda helposti esimerkiksi toisistaan graafisesti eroavia komponentteja. Abstrakteissa luokissa on vain komponentin runko, mutta ei toteutusta. Näin ollen samasta abstraktista luokasta periytyvät luokat tarjoavat samat rajapinnat. (Koskimies & Mikkonen 2005.)

6 KIRJASTO MONIKOSKETUSOVELLUSTEN KEHITTÄMISEEN

Työn tavoitteena oli tehdä monikosketussovellusten kehittämistä helpottava ohjelmointikirjasto, jota voi käyttää yhdessä Qt-kirjastojen kanssa. Nokian Qt-kirjastot ja kehitysympäristö ovat alustariippumaton ohjelmointiympäristö, jota käytetään graafisten sovellusten kehittämiseen useille käyttöjärjestelmille ja alustoille. Qt-kirjastoja käyttäen sovellusten kehittäminen on helppoa, sillä ne tuovat C++:aan modernien kielten ominaisuuksia. Qt-kirjastojen kehittyneiden grafiikka-, piirto- ja multimedia-ominaisuuksien ansiosta ne sopivat hyvin monikosketussovellusten kehittämiseen.

6.1 Kirjaston arkkitehtuuri ja käyttö

Kirjaston arkkitehtuuri on hyvin yksinkertainen, kuten kuviossa 8 näkyy. Se muodostuu kahdesta pääosasta, TUIO-viestejä vastaanottavasta TuioClient-luokasta ja kosketustapahtumia hallinnoivasta MTObjectManager-luokasta. Näiden luokkien tehtävänä on tuoda tieto kosketuspisteistä sovelluskehittäjän saataville. TuioClient kuuntelee TUIO-viestejä ja lähettää niistä sen jälkeen tiedon MTObjectManagerille Qt:n signaali-järjestelmällä. MTObjectManagerilla on lista kaikista monikosketusobjekteista, ja se tarkistaa mitä objektia käyttäjä koskettaa ja lähettää sille viestin.



KUVIO 8. Kirjaston rakenne.

Monikosketusobjektit voivat olla tämän kirjaston luokkien objekteja tai Qt-kirjaston omien luokkien objekteja, kuten nappeja tai tekstilaatikoita. Kirjaston avulla monikosketusominaisuuksia pystyy siis lisäämään jo olemassa oleviin Qt:llä tehtyihin sovelluksiin pienin muutoksin. Kirjasto on myös suunniteltu modulaariseksi, eli kirjaston osat on erotettu toisistaan riippumattomiksi itsenäisiksi osiksi, jolloin esimerkiksi kosketusdatan lähteitä voi lisätä ja kirjastoa voi näin käyttää myös ilman TUIO-protokollaa. Modulaarisen rakenteen ansiosta MultiTouchin analysointiohjelmaa voidaan myös käyttää kosketusdatan lähteenä, mutta yleensä se on kehittäjän kannalta hankalampi ja kömpelömpi ratkaisu kuin TUIO-protokollaa käyttävät ratkaisut.

```
//Luodaan TUIO-viestien vastaanottaja ja objektimanageri.
TuioClient *client = new TuioClient(3333);
MTObjectManager *manager = new MTObjectManager();

//Asetetaan ruudun koko, jotta pisteden koordinaatit saadaan laskettua oikein.
manager->setScreenSize(1920, 1080);

//Kytketään client ja manager toisiinsa Qt:n signaali-järjestelmällä.
connect(client, SIGNAL(addFinger(QFinger), manager, SLOT(addFinger(QFinger));
connect(client, SIGNAL(moveFinger(QFinger), manager, SLOT(moveFinger(QFinger));
connect(client, SIGNAL(removeFinger(QFinger), manager, SLOT(removeFinger(QFinger));

//Luodaan käyttöliittymäelementti
MTWidget *objekti = new MTWidget();
//Lisätään se manageriin
manager->addTouchable(objekti);
```

Kuvio 9. Kirjaston käyttö.

Opinnäytetyössä kehitetyn ohjelmointikirjaston käyttäminen on yksinkertaista, ja siihen tarvittava koodi lyhennettynä kuviossa 9. Monikosketussovellusta tehtäessä luodaan objektit luokista TuioClient ja MTObjectManager. MTObjectManagerille asetetaan näytön koko, jotta kosketuspisteiden koordinaatit saadaan laskettua oikein. Jotta TUIO-viestit tulisivat perille, yhdistetään TuioClient verkkoporttiin 3333 kutsumalla TuioClient-luokan funktiota connect. Käyttöliittymäelementit luodaan ohjelmakoodista. Jotta elementit voisivat vastaanottaa monikosketustapahtumia, täytyy niistä ilmoittaa MTObjectManagerille. Tämä tapahtuu kutsumalla MTObjectManagerin funktiota addTouchableWidget.

Erottamalla TUIO-viestien vastaanottaminen ja sovelluksen sisäinen monikosketuslogiikka toisistaan tuodaan kirjaston käyttöön joustavuutta. Qt:n signaalijärjestelmä mahdollistaa sen,

että samaan MTObjectManager-objektiin pystytään yhdistämään monta TUIO-viestejä vastaanottavaa komponenttia, joiden data tulee eri kosketusnäytöiltä.

Kirjasto sisältää muutaman luokan, joista voi lähteä kehittämään omia monikosketusobjekteja ja sovelluksia. Nämä luokat ovat nimeltään MTWidget, MTGraphicsItem ja MTGraphicsView. Seuraavissa luvuissa käsitellään kirjaston ja luokkien toimintaa sekä niiden ominaisuuksia tarkemmin.

6.2 Tangible User Interface Object -signaalien vastaanottaminen

TuioClient-luokka sisältää TUIO-viestien vastaanottamiseen ja kosketustietojen eteenpäin MTObjectManagerille lähettämiseen tarvittavat funktiot. Taustajärjestelmien lähettämät TUIO-viestit käyttävät viestimuotonaan Open Sound Control (OSC) -muotoa, joten TuioClient käyttää hyväkseen avoimen lähdekoodin OSC-kirjastoa. Tämä OSC-kirjasto hoitaa verkkoliikenteen ja osan viestien tulkinnasta. Koska OSC-kirjaston lisensointi on riittävän avoin ja salliva, ei ollut syytä tehdä omia luokkia verkkoliikenteen hoitamiseen.

TuioClient periytyy OSC-kirjaston luokasta PacketListener. TuioClient-luokassa on määritetty kolme funktiota tulevien pakettien prosessointiin: ProcessPacket, ProcessBundle ja ProcessMessage. Varsinainen viestien tulkinta tapahtuu ProcessMessage-funktiossa, jota kutsutaan jokaisen viestin kohdalla erikseen.

TuioClient-luokka pitää listaa kaikista näytöllä olevista kosketuspisteistä, ja tarkistaa sitä sen mukaan kuin uusia viestejä tulee. ProcessMessage-funktio tarkistaa onko viesti SET, ALIVE vai FSEQ tyyppiä, ja reagoi sen mukaan. SET-viestin tapauksessa funktio tarkistaa onko kysymyksessä täysin uusi kosketuspiste. Jos kosketuspiste on uusi, lisätään se muistiin uudella tunnuksella. Jos piste ei ole uusi, päivitetään sen tiedot. Kun TuioClient saa ALIVE-viestin, se poistaa muistista kaikki kosketuspisteet, joita ei ALIVE-viestissä mainita. Kosketuspisteiden tiedot lähetetään eteenpäin MTObjectManagerille, kun TuioClient saa FSEQ-viestin. FSEQ-viestin tullessa TuioClient tarkistaa, että FSEQ-viestin aikamerkintä ei ole liian vanha. Jos

aikamerkintä on tuore, lähettää TuioClient kaikki edellisen FSEQ-viestin jälkeen tapahtuneet muutokset Qt:n signaali-järjestelmällä MTObjectManager-objektille.

Lähettämällä tiedot kosketuspisteistä vain FSEQ-viestin tullessa vältetään ongelmat, joita UDP-verkkoliikenne aiheuttaa. Verkkoliikenteessä saattaa olla hidastuksia, paketteja saattaa hukkua matkalla ja niiden järjestys saattaa vaihtua. FSEQ-viestien aikamerkintöjen avulla varmistetaan että vanhaa kosketusdataa ei lähetetä edelleen MTObjectManagerille.

6.3 Objektien hallinta ja tapahtumien lähettäminen

MTObjectManager-luokka huolehtii tapahtumien lähettämisestä monikosketusinteraktiota sisältäville käyttöliittymäelementeille. Luokka pitää listaa monikosketusobjekteista ja niiden kanssa interaktiossa olevista monikosketuspisteistä.

Jotta viestit menisivät oikeille käyttöliittymäelementeille, tarkistetaan jokaisen QTuioClientilta tulleen viestin sisältämän kosketuspisteen koordinaatit ja verrataan niitä muistissa olevien käyttöliittymäelementtien peittämään alueeseen näytöllä. MTObjectManager-luokka sisältää alkeellisen tasojärjestelmän, jolla pystytään varmistamaan, että päällimmäisenä oleva elementti on se jolle tapahtuma lähetetään. Tasojärjestelmä perustuu siihen missä järjestyksessä elementit on lisätty, ja siihen ovatko ne näkyvissä vai ei.

Testattavan alueen muoto on aina tavallisesti nelikulmio riippumatta, siitä minkä muotoinen varsinainen käyttöliittymäelementti on. GraphicsItem-luokasta periytyvien objektien kohdalla testattavan alueen voi kuitenkin myös määrätä itse, minkä muotoiseksi haluaa, mutta yleensä nelikulmio on riittävän tarkka muoto käyttöliittymäelementeille.

Kosketuspisteen ollessa jonkin käyttöliittymäelementin kohdalla lähetetään sille sopiva tapahtuma. Tapahtumia on kolmenlaisia: TouchBegin, TouchUpdate ja TouchEnd. TouchBegin lähetetään silloin, kun käyttäjä asettaa sormensa elementin päälle. TouchUpdate-funktiota kutsutaan, kun käyttäjä liikuttaa sormeaan näytöllä ja TouchEnd funktiota kutsutaan, kun käyttäjä poistaa sormensa näytöltä.

Tapahtumien erotteleminen kolmeen osaan helpottaa sovelluskehittäjän työtä ja selkeyttää ohjelmakoodia. Eri interaktion vaiheita käsittelevä ohjelmakoodi voidaan erotella loogisiin osiin, jolloin ohjelman rakennetta ja toimintaa on helpompi suunnitella ja valmista ohjelmakoodia helpompi lukea ja ymmärtää.

6.4 Yleisimpien eleiden toteutus

Kirjasto sisältää toteutukset muutamille yleisimmille manipulaatioeleille. Symbolisten eleiden toteutusta ei ole tehty, sillä niiden tunnistamiseen tarvittava järjestelmä, toteutettaessa järkevästi, muodostaisi liian suuren kokonaisuuden käsiteltäväksi tässä opinnäytetyössä. Symbolisia eleitä myös hyödynnetään käyttöliittymissä niin harvoin, että saavutettu hyöty olisi ollut varsin pieni suhteessa työmäärään. Kirjaston sisältämät eleet ovat nipistysele eli skaalaus kahden sormen avulla, pyöritys ja siirto kahden sormen avulla sekä yhden sormen napautukset.

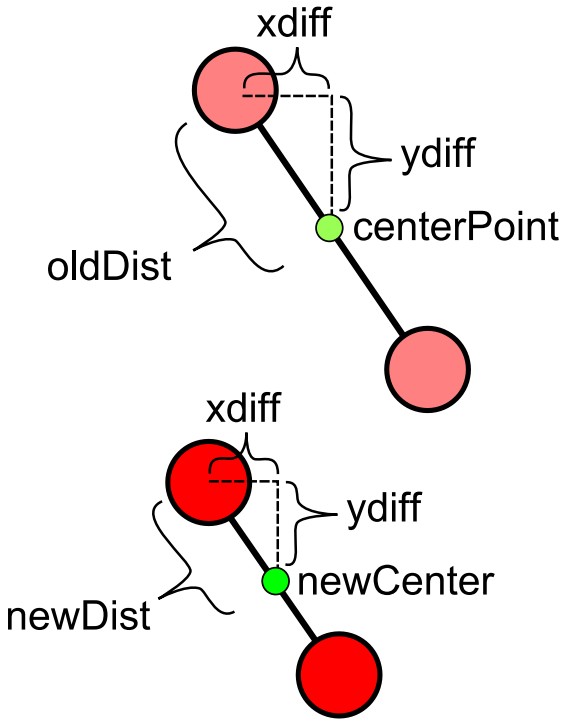
Nipistysele on yksi parhaita esimerkkejä manipulaatioeleistä. Elettä tehdessään käyttäjä asettaa kosketuspinnalle kaksi sormea samasta kädestä, yleensä etusormi ja peukalo, ja joko tuo niitä lähemmäs toisiaan tai loitontaa niitä toisistaan. Nipistyselettä käytetään yleensä skaalauksen hallitsemiseen sovelluksissa.

```
//Lasketaan sormien edellisten paikkojen ja edellisen keskipisteen
//välinen etäisyys. Tämä tehdään jokaiselle sormelle erikseen
double xdiff = finger.getLastPos().x() - centerPoint.x();
double ydiff = finger.getLastPos().y() - centerPoint.y();
oldDist += sqrt(pow(xdiff, 2) + pow(ydiff, 2));

//Lasketaan sormien nykyisten paikkojen ja uuden keskipisteen välinen
//etäisyys.
xdiff = finger.getPos().x() - newCenter.x();
ydiff = finger.getPos().y() - newCenter.y();
newDist += sqrt(pow(xdiff, 2) + pow(ydiff, 2));

//Jaetaan uusi etäisyys vanhalla, jotta saadaan niiden välinen suhde.
double scaleAmount = newDist / oldDist;
//Käytetään saatua suhdetta skaalaukseen.
scale(scaleAmount);
```

KUVIO 10. Nipistyseleen toteutus yksinkertaistettuna.

Nipistyseleen toteuttaminen työssä tehtiin hieman poiketen tavallisesta lähestymistavasta. Yleensä nipistysele toteutetaan siten, että skaalauksen määrä lasketaan vain kahdesta sormesta vaikka kosketuspinnalla olisi useampiakin sormia. Tai niin, että nipistyseleen voi tehdä vain jos kosketuspinnalla on ainoastaan kaksi sormea. Tässä tapauksessa toteutus ohjelmoitiin siten, että kaikki kosketuspinnalla olevat sormet otetaan huomioon.

Käyttäjän sormien liikkuessa kosketuspinnalla algoritmi laskee kosketuspisteiden keskipisteen ja tallentaa sen muistiin. Tämän jälkeen algoritmi laskee sormien edellisten koordinaattien ja edellisen keskipisteen välisen yhteenlasketun etäisyyden ja sormien uusien koordinaattien ja uuden keskipisteen välisen yhteenlasketun etäisyyden. Viimeisenä lasketaan uuden ja vanhan etäisyyden suhde ja käytetään tätä arvoa skaalauksen tekemiseen. Sormien koordinaattien keskipistettä käytetään skaalauksen laskemisen lisäksi siirtojen laskemiseen. Esimerkiksi käyttöliittymänäkymää voidaan siirtää vanhan keskipisteen ja uuden keskipisteen erotuksen verran. Kuviossa 11 esitetään nipistyseleen toteutus graafisesti.



KUVIO 11. Nipistyseleen algoritmin graafinen esitys.

Toinen hyvä esimerkki manipulaatioeleistä on kahden sormen pyöritysele. Pyöritysele liittyy läheisesti nipistyseleeseen, ja yleensä ne ovatkin molemmat käytössä niin, että yhtä aikaa voi

sekä skaalata että pyörittää objekteja. Elettä tehdessään käyttäjä asettaa kaksi sormea kosketuspinnalle ja kääntää niitä pinnalla ottaen akseliksi joko toisen sormen tai sormien välisen keskipisteen. Elettä käytetään objektien tai sovellusnäkymän pyörittämiseen.

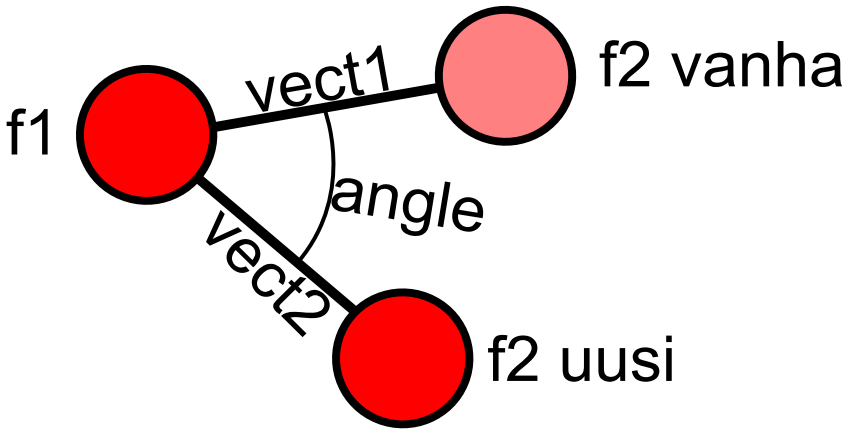
Pyöritysele toimii siten, että käyttäjän liikuttaessa sormiaan kosketuspinnalla algoritmi laskee liikkuvan sormen ja paikallaan pysyvän sormen välisen kulman, ja kääntää pyöritettävää objektia kulman verran. Algoritmi on kuitenkin toteutettu niin, että se, kumpi sormista on liikutettava ja kumpi pysyy paikallaan, ei ole ennalta määrättyä. Käyttäjä pystyy tekemään eleen liikuttaen kumpaa tahansa sormea, tai halutessaan molempia.

```
//Pyörityksen laskemiseen käytetään aina kahta ensimmäistä näytölle
//asetettua sormea
QFinger f1 = fingers[0];
QFinger f2 = fingers[1];

//Lasketaan sormien nykyisten paikkojen välinen vektori.
QpointF vect1 = QPointF(f2.getPos().x() - f1.getPos().x(), f2.getPos().y() -
f1.getPos().y());

//Lasketaan toisen sormen edellisen paikan ja toisen nykyisen paikan
//välinen vektori.
QPointF vect2 = QPointF(f2.getLastPos().x() - f1.getPos().x(),
f2.getLastPos().y() - f1.getPos().y());

//Lasketaan vektorien ja x-akselin väliset kulmat.
double angle1 = atan2(vect1.y(),vect1.x());
double angle2 = atan2(vect2.y(),vect2.x());

//Lopullinen kulma saadaan edellisten kulmien erotuksen negaatiosta.
//Muutetaan kulma asteiksi.
angle = -(angle2 - angle1);
angleDeg = (angle*180) / 3.14159265;
//Pyöritetään saadun kulman mukaan.
rotate(angleDeg);
```

KUVIO 12. Pyörityseleen toteutus

Pyörityseleen algoritmi näkyy kuviossa 12. Algoritmi tarkistaa aluksi, kumpi käyttäjän sormista liikkuu tällä hetkellä, ja siirtyy sen jälkeen laskemaan sormien välistä kulmaa.

Sormien välistä kulmaa laskettaessa lasketaan ensin vektorit liikkuvan pisteen nykyisten koordinaattien ja paikallaan pysyvän pisteen koordinaattien, sekä liikkuvan pisteen edellisten koordinaattien ja paikallaan pysyvän pisteen koordinaattien välillä. Tämän jälkeen lasketaan näiden vektorien ja x-akselin väliset kulmat. Näiden kulmien avulla lopulta lasketaan kulma jonka verran objektia tai sovelluksen näkymää pyöritetään. Kuviossa 13 on esitetty pyörityseleen algoritmi graafisesti.



KUVIO 13. Pyörityseleen algoritmin graafinen esitys.

6.5 Perusluokat monikosketuselementtien kehittämiseen

Opinnäytetyössä kehitetty ohjelmointikirjasto sisältää kolme luokkaa, joita voi käyttää pohjana omien monikosketuselementtien rakentamisessa. Nämä luokat ovat MTWidget, MTGraphicsItem ja MTGraphicsView. Sovelluskehittäjä voi käyttää näitä luokkia tekemällä oman luokan, joka periytyy näistä perusluokista. Luokat eivät sisällä mitään piirtofunktioita, vaan sovelluskehittäjän pitää tehdä omalle käyttöliittymäelementilleen oma piirtofunktio.

Luokat sisältävät yleisimpien eleiden toteutukset, ja näin ollen skaalaus,- pyöritys- ja siirtoeleitä voi käyttää näistä luokista tehtyjen käyttöliittymäelementtien kanssa. Luokat sisältävät myös signaalin `tapped`, joka lähetetään, kun käyttäjä painaa elementtiä yhdellä

sormella. Jos elementeistä haluaa tehdä staattisia voi eleiden käyttömahdollisuuden kytkeä pois päältä funktioilla enableRotate, enableScale ja enableMove. Funktioille annetaan parametrina boolean arvo `true` tai `false`. Tällöin elementti kuitenki vielä lähettää `tapped`-signaalin sitä painettaessa.

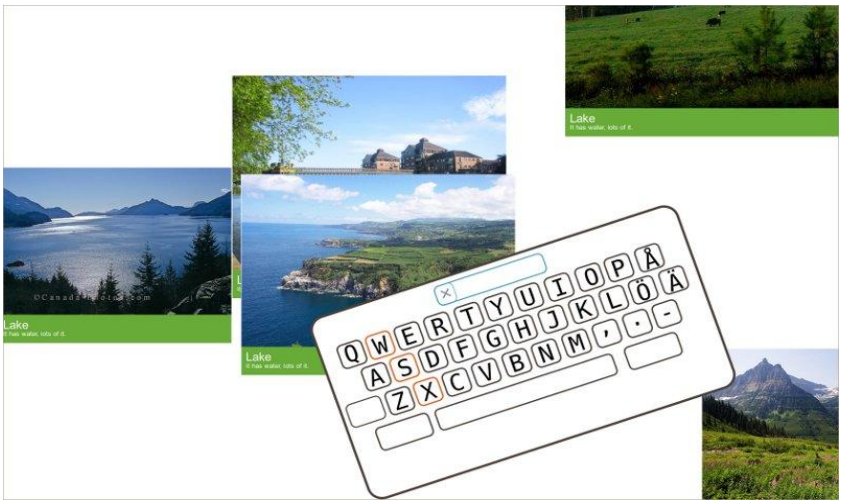
Omien elementtien kehittäminen luokkien avulla on helppoa, sillä suurinta työtä, eli koodia tapahtumien käsittelyyn, ei tarvitse tehdä itse. Omien eleiden lisääminen elementteihin onnistuu funktioiden touchDown, touchUp ja touchMove avulla. Nämä ovat virtuaalisia funktioita, joita kutsutaan kun käyttäjä koskettaa käyttöliittymäelementtiä. Virtuaalisilla funktioilla ei ole oletustoteutusta, vaan sovelluskehittäjä sijoittaa näihin funktioihin kosketustapahtumiin liittyvän ohjelmakoodinsa. Luokat myös sisältävät fingers-listan, joka sisältää kaikki kyseiseen elementtiin liittyvät kosketuspisteet. Tämä auttaa monimutkaisempien eleiden toteutuksessa.

7 ESIMERKKIOHJELMAT

Tässä luvussa käsitellään ohjelmistoja, jotka on kehitetty edellisessä luvussa esiteltyä kirjastoa käyttäen. Ohjelmat ovat hyvin yksinkertaisia demoja joiden tarkoitus on esitellä kirjaston ominaisuuksia, eikä niiden ole tarkoitus olla valmiita sovelluksia. Lukuun ottamatta valokuvademoa kaikki demot on tehty CENTRIA tutkimuksen & kehityksen tarpeisiin.

7.1 Valokuvademo

Valokuvien manipulointi monikosketuseleiden avulla on ollut suosittu tapa esitellä monikosketusnäyttöjä ja -käyttöliittymiä. Tavallisesti valokuvademo sisältää joukon valokuvia joita käyttäjä voi siirtää, pyörittää ja skaalata eleiden avulla.



KUVIO 14. Valokuvademon käyttöliittymä.

Opinnäytetyössä tehty demo sisältää valokuvien manipuloinnin lisäksi virtuaalisen näppäimistön, jonka avulla valokuviin voi liittää tageja. Tässä tapauksessa tagilla tarkoitetaan sanoja tai nimiä, jotka kuvailevat valokuvan sisältöä.

Kun käyttäjä lisää valokuvaan tageja, tulevat ne näkyviin ruudun oikeaan reunaan listana. Listassa olevia tageja painamalla siihen liittyvät kuvat voi piilottaa ruudulta. Monikosketuksen avulla joitakin toimintoja, kuten asioiden listasta valitsemista voidaan nopeuttaa. Valokuvademossa listasta voi valita usean elementin kerrallaan kahdella eri tavalla. Käyttäjä voi painaa tageja yhtä aikaa, tai painaa kahdella sormella yhtä tagia, ja yhdellä sormella toista. Tällöin kaikki näiden välissä olevat elementit valitaan.

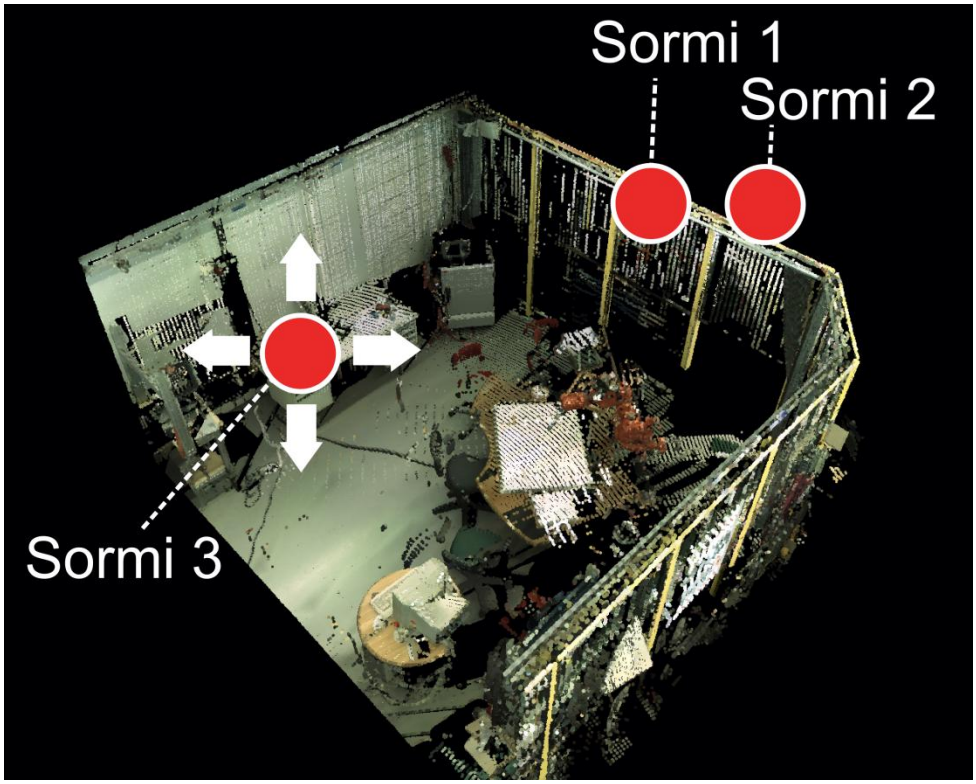
Demo käyttää hyväkseen kirjaston MTGraphicsItem ja MTGraphicsView luokkia. MTGraphicsView-luokkaa käytetään suoraan sellaisenaan, ja käyttöliittymäelementit käyttävät luokkia jotka periytyvät MTGraphicsItem-luokasta.

Virtuaalisen näppäimistön nappeja tehdessä täytyi ottaa huomioon virheelliset kosketustapahtumat. Varsinkin MultiTouchin näytöllä testattaessa huomasin, että hyvin usein jotain näppäimistön nappia painettaessa näyttö tunnisti useampia painalluksia samaan kohtaan. Tämä oli kuitenkin helppo havaita ja estää koodissa asettamalla napin hyväksymille kosketustapahtumille aikaraja. Jotta nappi hyväksyisi uuden painalluksen, on edellisestä täytynyt kulua 0.4 sekuntia.

7.2 Pistepilven selaaminen

CENTRIA tutkimus ja kehityksen Leica-laserskannerilla voidaan kuvata tiloja ja esineitä. Tuloksena kuvauksista saadaan kolmiulotteista pistedataa. Pistepilvi tarkoittaa skannerista saatua dataa, joka koostuu pisteistä, joilla on kolmiulotteiset koordinaatit. Pistepilven selausdemossa voidaan Leican tuottamaa pistedataa selata ja tarkastella. Taustana demossa on CENTRIAn tutkijoiden kehittämä sovellus ja sovelluskomponentit, joihin opinnäytetyössä on tehty monikosketusominaisuuksia.

Sovelluksen käynnistyessä ladataan ennalta määritetty pistepilvi sovellukseen. Pistepilvi piirretään mustalle taustalle. Pistepilvi sisältää myös väritietoja, jotka on poimittu skannerin ottamista valokuvista.



KUVIO 15. Pistepilveä voi pyörittää siihen tarkoitetun eleen avulla.

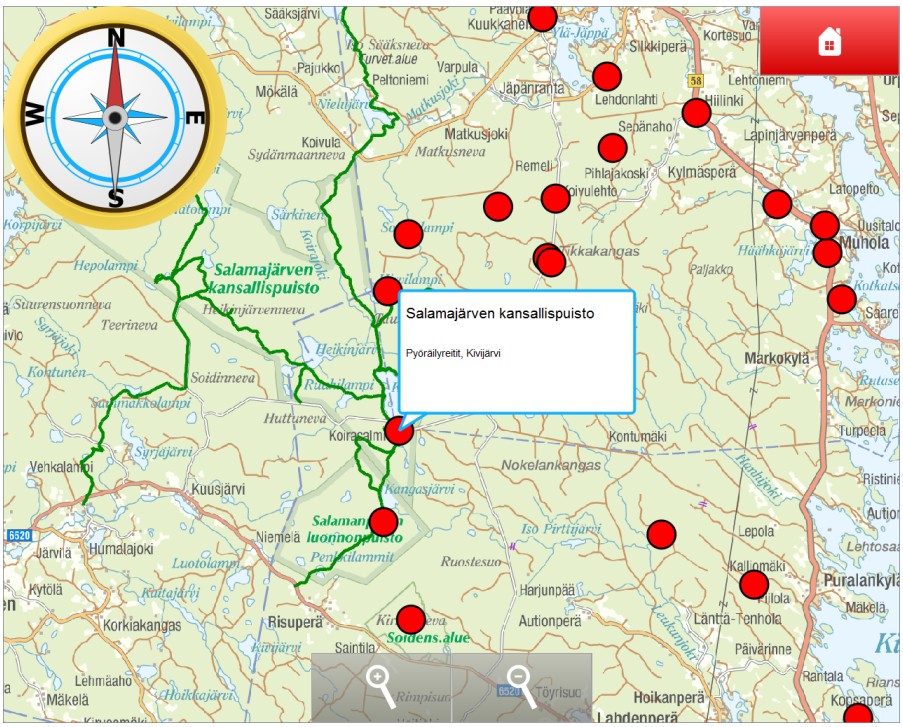
Pistepilven selaaminen tapahtuu kolmen eleen avulla. Yhdellä sormella kameraa liikutetaan ylös, alas, oikealle ja vasemmalle. Kahdella sormella kameraa voidaan liikuttaa lähemmäs tai kauemmas pisteestä johon kamera on sillä hetkellä kohdistettu. Käyttäjän laittaessa kolmannen sormen kosketuspinnalle voi hän tätä sormea käyttäen pyörittää kameraa sen pisteen ympäri, johon kamera on suunnattu. Näkymän pyörittämiseen käytettävä ele on esitetty kuviossa 15.

Pistepilven selausdemo käyttää ainoastaan kirjaston MTWidget-luokkaa käyttöliittymässään. Pistepilven piirtämisestä huolehtiva luokka vastaanottaa myös kosketustapahtumia, ja kaikki eleet ja niihin liitetyt toiminnot hoidetaan samassa luokassa.

7.3 Karttasovellukset

Kirjaston kehitystyön aikana aloitettiin karttasovelluksen kehitys, joka on tällä hetkellä loppusuoralla. Alunperin karttasovellus oli tarkoitus rakentaa CENTRIA tutkimus ja

kehityksen oman karttamoottorin pohjalle, mutta projektin loppuvaiheessa siirryttiin käyttämään rasterikarttoja. Rasterikartat ovat kuvista tehtyjä karttoja, joita ei tarvitse piirtää ohjelman ajon aikana. Tästä seuraa kuitenkin se, että ne eivät ole yhtä tarkkoja kuin vektorikartat. Vaikka sovellus onkin toteutettu rasterikartoilla, tehtiin myös Centrian karttamoottorista versio, jota voi käyttää monikosketussovelluksissa.


N
W
E
S
Salamajärven kansallispuisto
Pyöräilyreitit, Kivijärvi
Salamajärven kansallispuisto
Salamajärven luonnonpuisto
Soidens.alue
Muhola
Markkylä
Puralankylä
Risuperä
Korkiakangas
Kuusjärvi
Humalajoki
Koivula
Matkusjoki
Kylmäsperä
Lapinjärvenperä
Kontumäki
Nokelankangas
Hoikanperä
Autionperä
Harjunpää
Lepola
Kallomäki
Päivärinne
Rantala
Lehtoniemi
Lehdonlahti
Sepänaho
Pihlajakoski
Jäpänranta
Varpula
Pajukko
Mökälä
Peltoniemi
Kuukkaheni
Siikkiperä
Kortesuo
Hiilinki
Lantta-Tenhola
Saintila
Töyrisuo
Rimpisuo
Lehmäaho
Mäkelä
Järvilä
Kytölä
Vehkalahti
Niemelä
Huttuneva
Soidinneva
Teerineva
Hepolampi
Sääksjärvi
Sääksneva
Turvetalue
Sydänmaanneva
Matkusneva
Remeli
Paljakko
Nokelankangas
Iso Pirttijärvi
Ruostesuo
Kangasjärvi
Kaitajärvi
Kivijärvi
Hoikkajärvi
Heikinjärvi
Heikinjärvenneva
Kontunen
Suurensuonneva
Korpijärvi
Sarkinen
Kivijärvi
Ruokolampi
Kuusilampi
Pentilammit
Luotolampi
Syrjäjärvi
Kuusjärvi
Turpeela
Ristinie
Autio
Lehtosa
Mäkelä
Rantala
Koosaperä
Lahdenperä
Häähkajärvi
Kolkats
Säärel
Lappelto
Uusitalo
Rutuse
Markonie
Hoikanjoki
Kiviojoki
Ruokolampi
Kuusilampi
Pentilammit
Luotolampi
Kaitajärvi
Kivijärvi
Hoikkajärvi
Kangasjärvi
Kuusjärvi
Syrjäjärvi
Vehkalahti
Korpijärvi
Hepolampi
Suurensuonneva
Teerineva
Kontunen
Soidinneva
Huttuneva
Koirasalmi
Nielujärvi
Matkusjoki
Matkusneva
Sydänmaanneva
Koivula
Mökälä
Pajukko
Peltoniemi
Varpula
Jäpänranta
Lehdonlahti
Sepänaho
Hiilinki
Lapinjärvenperä
Kylmäsperä
Pihlajakoski
Remeli
Kuukkaheni
Siikkiperä
Kortesuo
Lehtoniemi
Muhola
Markkylä
Puralankylä
Lepola
Kallomäki
Päivärinne
Lantta-Tenhola
Autionperä
Harjunpää
Risuperä
Korkiakangas
Humalajoki
Kytölä
Järvilä
Niemelä
Lehmäaho
Mäkelä
Saintila
Töyrisuo
Rimpisuo
Hoikanperä
Lahdenperä
Koosaperä
Rantala
Säärel
Kolkats
Lappelto
Uusitalo
Häähkajärvi
Paljakko
Kontumäki
Nokelankangas
Iso Pirttijärvi
Ruostesuo
Kangasjärvi
Kaitajärvi
Kivijärvi
Hoikkajärvi
Heikinjärvi
Heikinjärvenneva
Kontunen
Suurensuonneva
Korpijärvi
Hepolampi
Sääksjärvi
Sääksneva
Turvetalue
Sydänmaanneva
Matkusneva
Matkusjoki
Nielujärvi

tiedot ovat paikannimi, retkeilyreitistö, pitempi esittelyteksti ja yksi tai useampia valokuvia. Pistettä painamalla aukeaa infokupla, josta voi myös aukaista, joko tekstin selaamiseen tarkoitetun näkymän, tai kuvien selaamiseen tarkoitetun näkymän. Tämä karttasovellus on laajin kirjaston avulla tehty sovellus, ja kirjaston kehitystyö tapahtuikin pitkälti tämän sovelluksen vaatimien toimintojen ehdoilla.

8 POHDINTA JA TULOKSET

Monikosketustekniikan ja -käyttöliittymien tulevaisuutta on vaikea ennustaa. Kuluttajamarkkinoilla ei ole vielä sellaisia laitteita, jotka vaikuttaisivat tottumuksiimme käyttää tietokoneita. Monikosketustekniikoilla on edessä vielä pitkä tie, ennen kuin niitä voidaan edes ajatella hiiren ja näppäimistön korvaajina. Monikosketustekniikka on kuitenkin löytänyt paikkansa etenkin mobiililaitteista, kuten älypuhelimet ja tablet-tietokoneet.

Monikosketustekniikkaa hyödyntäviä mobiililaitteita tulee markkinoille koko ajan enemmän ja enemmän, ja ainakin monikosketuksen lähitulevaisuus riippuu paljon näiden laitteiden menestyksestä. Näitä laitteita ovat esimerkiksi Applen Ipad, Hewlett-Packardin Slate ja muut tablet-tietokoneet. Tablet-tietokoneet ovat kannettavaa tietokonetta pienempiä laitteita, joissa ei ole fyysistä näppäimistöä, vaan tietokonetta hallitaan kosketusnäytön avulla. Tällä hetkellä tablet-tietokoneet eivät näytä sopivan kannettavien tietokoneiden korvaajiksi, vaan ne ovat ennemmin uusi kannettavien multimedia- ja Internet-laitteiden kategoria. Joka tapauksessa tablet-tietokoneet ovat ensimmäisiä laitteita, jotka tuovat monikosketuskäyttöliittymät kuluttajamarkkinoille niin isoilla näytöillä, että monikosketusta voidaan hyödyntää muuhunkin kuin nipistyseleellä skaalamiseen.

On varsin todennäköistä, että myös optisista monikosketusnäytöistä saadaan tehtyä versioita, joissa on pienempi kotelointi, ja jotka eivät näin ollen vaadi niin paljon tilaa. Tämä on mahdollista erilaisten valosensoreiden kehityksen myötä. Tällöin saadaan tehtyä isoja optisia näyttöjä, jotka eivät kuitenkaan vaadi kohtuuttomasti tilaa. Sovelluskehitys isoilla näytöillä tulee varmasti vielä pitkään keskittymään yleisiin tiloihin sijoitettuihin infopöytä-tyylisiin sovelluksiin.

Monikosketuksen mahdollisuudet eivät hahmotu täysin, jos sitä ajatellaan vain tietokoneen käyttönä usealla sormella. Hiirelle suunnitellut työpöytäsovellukset ja niihin liittyvien käyttöliittymäajattelumallien suora siirtäminen monikosketussovelluksiin ei anna monikosketuksen mahdollisuuksista parasta kuvaa. Käyttöliittymät tulisi suunnitella

uudestaan, jotta monikosketussovelluksen käyttämisestä tulee helppoa ja tehokasta. Tulevaisuudessa monikosketuksen ja muiden NUI-tekniikoiden paras anti ei ole itse tekniikka tai sen mahdollistamat käyttöliittymät. Luonnolliset käyttöliittymät muuttavat digitaalisen informaation kanssa vuorovaikutuksessa olemisen vastaamaan enemmän fyysisten esineiden kanssa vuorovaikutuksessa olemista.

Opinnäytetyössä kehitetty kirjaston kehitystyö onnistui hyvin, sillä se täytti sovelluskehityksen tarpeet. Koska kirjasto on alun perin suunniteltu melko yksinkertaisten karttasovellusten kehittämistä varten, ei se välttämättä sisällä muunlaisten sovellusten kehittämiseen tarvittavia ominaisuuksia.

Pahimpina kirjaston puutteina voidaan pitää monikosketusobjektien tasojen puuttumista ja sitä, että objektien kosketusalue on aina nelikulmion muotoinen. Tasot ovat tekniikka, jolla hallitaan sitä, missä järjestyksessä monikosketuselementit saavat monikosketustapahtumia. Opinnäytetyössä tehdyssä kirjastossa elementit saavat monikosketustapahtumat aina siinä järjestyksessä, jossa ne on lisätty ruudulle. Jos ohjelmassa on päällekkäisiä käyttöliittymäelementtejä, niitä ei voi siirtää alemmalle tai ylemmälle tasolle. Kumpikaan puute ei vaikuttanut esimerkkiohjelmien tai karttasovellusten kehittämiseen. Siitä huolimatta nämä ominaisuudet olisivat jatkokehityksen kannalta tärkeitä. Qt:n Graphics View –rajapintaa käyttäviä ohjelmia varten nämä muutokset eivät olisi kovin suuria.

Monikosketustekniikat ja ohjelmistorajapinnat kehittyvät hyvin nopeasti. Tämän opinnäytetyön tekemisen aikana on Qt-kirjastoihin tullut oma tuki monikosketukselle ja eleille. Qt:n monikosketusominaisuuksia voi käyttää esimerkiksi Windows 7–käyttöjärjestelmän kanssa, ja tulevaisuudessa Nokian kosketusnäytöllisten puhelinten kanssa. Samaan aikaan MultiTouch on kehittänyt ajureita Windows 7–käyttöjärjestelmälle.

Nämä kehityssuunnat ovat päällekkäisiä tässä opinnäytetyössä esitetyn monikosketuskirjaston kanssa ja onkin mahdollista, että tulevaisuudessa Centrian kehittäessä sovelluksia MultiTouchin monikosketusnäytölle, voidaan harkita Qt:n ja MultiTouchin omien järjestelmien käyttämistä.

Tässä työssä tehty monikosketuskirjasto on kuitenkin MultiTouchin järjestelmää joustavampi, sillä se ei ole sidottu mihinkään käyttöjärjestelmään tai laitealustaan. Qt:n omien monikosketusominaisuuksien käyttö optisten monikosketuspöytien kanssa saattaa olla myös ongelmallista. Tällöin kosketusdataan ei päästä yhtä hyvin käsiksi, vaan väliin tulee monia ohjelmistokerroksia. Tämä saattaa vaikeuttaa esimerkiksi virhetunnistusten pois suodattamista. MultiTouchin näyttöjä on mahdollista liittää toisiinsa, ja muodostaa näin yksi suurempi näyttö. Yhdistetyn näytön käyttö kuitenkin vaatii Linux-pohjaisen käyttöjärjestelmän, ja silloin vaihtoehdoiksi ohjelmointirajapintojen suhteen jäävät MultiTouchin omat rajapinnat, sekä tässä työssä tehty kirjasto.

Tässä työssä kehitetyn kirjaston etuja suhteessa MultiTouchin sovelluskehityskirjastoon ovat sen pieni koko, yksinkertaisuus ja se, että oma kirjastoni käyttää käyttöliittymien tekemiseen Qt-kirjastoja. Qt-kirjastot ovat hyvin yleisessä käytössä, ja niitä käytetään myös työpöytä- ja mobiilisovellusten kehittämiseen. MultiTouchin kirjastot sen sijaan käyttävät omia käyttöliittymäkirjastojaan, joita ei voi käyttää muuhun sovelluskehitykseen.

LÄHTEET

Albinsson, P-A & Zhai, S. 2003. High Precision Touch Screen Interaction. Yhdysvallat. ACM Press.

Blake, J. 2009. WIMP is to GUI as OCGM (Occam) is to NUI. Www-dokumentti. Saatavissa: http://nui.joshland.org/2009/12/wimp-is-to-gui-as-ocgm-occam-is-to-nui.html. Luettu 5.2.2010.

Blanchette, J. 2008. The Little Manual of API Design. Trolltech.

Bloch, J. 2006. How to Design a Good API and Why it Matters. Www-dokumentti. Saatavissa: http://lcsd05.cs.tamu.edu/slides/keynote.pdf. Luettu 28.2.2010.

Buxton, B. 2009a. Multi-Touch Systems that I Have Known and Loved. Www-dokumentti. Saatavissa: http://www.billbuxton.com/multitouchOverview.html. Luettu 22.12.2009.

Buxton, B. 2009b. Gesture based interaction. Yhdysvallat.

Cooper, A & Reimann, R. 2003. About Face 2.0: The Essentials of Interaction Design. Yhdysvallat. Wiley Publishing.

Dietz, P & Leigh, D. 2001. Diamondtouch: a multi-user touch technology. New York. ACM Press.

Ettrich, M. 2005. Designing Qt-Style C++ APIs. Www-dokumentti. Saatavissa: http://doc.trolltech.com/qq/qq13-apis.html. Luettu 24.2.2010.

Helander, M., Landauer & T. Prabhu, P. 1997. Handbook of human-computer interaction. North Holland.

Hunt, A. & Thomas, D. 2009. The Pragmatic Programmer. Stoughton: Courier Stoughton.

JazzMutant. 2009. Www-dokumentti. Saatavissa: http://www.jazzmutant.com/. Luettu 5.1.2010.

Kaltenbrunner, M. 2009. Www-dokumentti. Saatavissa: http://tuio.org/. Luettu 6.5.2010.

Kaltenbrunner, M., Bovermann, T., Bencina, R. & Costanza, E. 2005. Barcelona. Pompeu Fabran Yliopisto.

Kim, J., Park, J., Kim, H & Lee, C. 2007. Human Computer Interaction Using Multi-touch Tabletop Display. Yhdysvallat. IEEE Computer Society.

Koskimies, K. & Mikkonen, T. 2005. Ohjelmistoarkkitehtuurit. Jyväskylä: Gummerus Kirjapaino Oy.

Kourakis, S. 2004. On gestural interaction and modular gestures. Espanja. Pompeu Fabran yliopisto.

L.Y.L. Muller 2008. Multi-touch displays: design, applications and performance evaluation. Amsterdamin Yliopisto.

Microsoft. 2010. Microsoft Surface Architecture. Www-dokumentti. Saatavissa: http://msdn.microsoft.com/en-us/library/ee804755.aspx. Luettu 6.5.2010.

Miller, C. Video monikosketuskäyttöliittymistä. 2009. Saatavissa: http://10gui.com. Katsottu. 10.1.2010.

Miller, P. 2009. Stantum's mind-blowing multitouch interface on video! Www-dokumentti. Saatavissa: http://www.engadget.com/2009/02/19/stantums-mind-blowing-multitouch-interface-on-video/. Luettu 13.1.2010.

Nardi, A. Grafiti. 2008. Italia. Pisan Yliopisto.

Nespolous, J., Perron, P & Lecours, R. 1986. The Biological Foundations of Gestures: Motor and Semiotic Aspects. Hillsdale. Lawrence Erlbaum Associates.

Nielsen, J. 1993. Usability Engineering. Yhdysvallat. Academic Press.

Norman, D. 1990. The Design of Everyday Things. Yhdysvallat. Basic Books.

Norman, D. 2010. Natural User Interfaces Are Not Natural. Www-dokumentti. Saatavissa: http://jnd.org/dn.mss/natural_user_interfaces_are_not_natural.html. Luettu 11.4.2010.

NUIGroup 2010. Www-dokumentti. Saatavissa: http://wiki.nuigroup.com/Natural_User_Interface. Luettu 5.2.2010.

NUIGroup Authors. 2009. Multi-Touch Technologies. Www-dokumentti. Saatavissa: http://nuicode.com/attachments/download/112/First_Edition_Community_Release.pdf. Luettu 18.12.2009.

Raisamo, R. 1999. Multimodal Human-Computer Interaction: a constructive and empirical study. Tampere. Tampereen Yliopisto.

Reactable Systems. 2010. Www-dokumentti. Saatavissa: http://www.reactable.com

Saffer, D. 2009. Designing Gestural Interfaces. Kanada. O'Reilly Media.

Seow, S., Wixon, D., MacKenzie, S., Jacucci, G., Morrison, A. & Wilson, A. 2009. Multitouch and Surface Computing. Boston. ACM Press.

Stantum. 2010. Www-dokumentti. Saatavissa: www.stantum.com. Luettu 15.1.2010.

Thörnlund, M. 2007. Gesture Analyzing for Multi-Touch Screen Interfaces. Ruotsi. Lulean yliopisto.

UMCPortal. 2009. Multi-touch comes to resistive screens. Www-dokumentti. Saatavissa: http://www.umpcportal.com/2009/02/multi-touch-comes-to-resistive-touchscreens. Luettu

20.2.2010.

Westerman, W., Elias, J. & Hedge, A. 2001. Multi-Touch: New Tactile 2-D Gesture Interaction for Human-Computer Interaction. Yhdysvallat.

Widgor, D., Fletcher, J. & Morrison, G. 2009. Designing User Interfaces for Multi-Touch and Gesture Devices. Yhdysvallat. ACM Press.

Wikipedia 2010a. Www-dokumentti. Saatavissa: http://en.wikipedia.org/wiki/Touchscreen. Luettu 15.1.2010.

Wikipedia 2010b. Www-dokumentti. Saatavissa: http://en.wikipedia.org/wiki/Resistive_touchscreen. Luettu 17.1.2010.

Wilson, T. 2007. Www-dokumentti. Saatavissa: http://electronics.howstuffworks.com/iphone2.htm. Luettu 20.1.2010.

Wright, M & Freed, A. 1997. Open Sound Control: A New Protocol for Communicating with Sound Synthesizers. Thessaloniki. International Computer Music Association.

Wu, M., Shen, K., Ryall, K., Forlines, C., & Balakrishhnan, R. 2006. Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces. Yhdysvallat. IEEE Computer Society.