Janne Ramstedt

SOVELLUSYMPÄRISTÖ JA KÄYTTÖLIITTYMÄKIRJASTO MONIKOSKETUSLAITTEISIIN

Tietotekniikan koulutusohjelma

2010

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SOVELLUSYMPÄRISTÖ JA KÄYTTÖLIITTYMÄKIRJASTO MONIKOSKETUSLAITTEISIIN

Ramstedt, Janne
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Joulukuu 2010
Ohjaaja: DI Niemi, Juha
Sivumäärä: 37
Liitteitä: 0

Asiasanat: järjestelmäarkkitehtuuri, kosketusnäyttö, käyttöliittymät, C#, Microsoft .NET

Isot monikosketuslaitteet ovat vielä melko harvinaisia ja ne ovat yleensä käytössä vain yhtä tiettyä tarkoitusta varten, kuten esimerkiksi messuilla interaktiivisina tuote-esitteinä. Laitteet ovat yleensä myös kalliita eivätkä sovellu kuluttajakäyttöön. Opinnäytetyön tarkoituksena oli kehittää uudenlainen monikäyttöinen järjestelmä monikosketuslaitteisiin, joka soveltuu usealle yhtäaikaiselle käyttäjälle. Järjestelmä on erityisesti suunniteltu laitteisiin joiden rakentaminen on suhteellisen halpaa ja helppoa.

Työn teko aloitettiin taustatutkimuksella. Taustatutkimuksessa selvitettiin eri monikosketustekniikoita ja tutustuttiin monikosketussovelluksissa käytettäviin ohjelmistokirjastoihin. Samalla valittiin myös työssä käytettävät työkalut.

Opinnäytetyössä luotiin isoihin monikosketuslaitteisiin tarkoitettu järjestelmä. Järjestelmä on suunniteltu usealle yhtäaikaiselle käyttäjälle niin että jokainen käyttäjä voi ajaa tarvitsemiaan ohjelmia tai jakaa ohjelmia toisten käyttäjien kanssa. Se koostuu sovellusympäristöstä ja käyttöliittymäkirjastosta, jonka avulla voidaan luoda uusia sovelluksia järjestelmään. Sovellusympäristöön luotiin myös muutamia sovelluksia, joita voidaan käyttää esimerkkinä ja pohjana uusien sovellusten kehittämiseen.

APPLICATION ENVIRONMENT AND USER INTERFACE LIBRARY FOR MULTI-TOUCH EQUIPMENT

Ramstedt, Janne
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Technology
December 2010
Supervisor: MSc Niemi, Juha
Number of pages: 37
Appendices: 0

Keywords: system architecture, touch screen, user interfaces, C#, Microsoft. NET

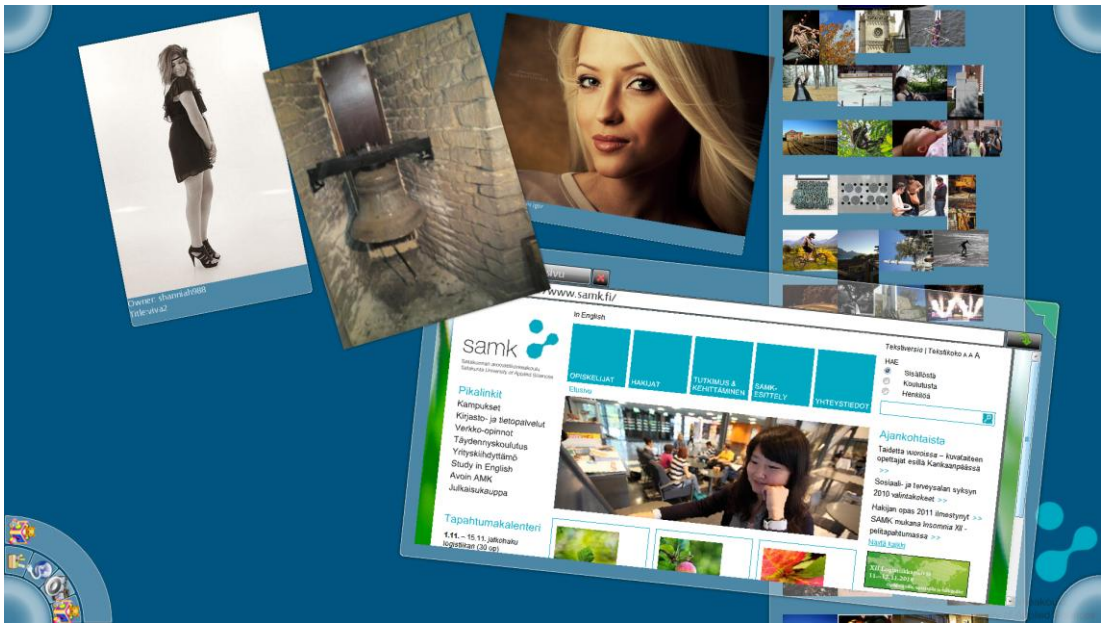
Large multi-touch devices are still relatively rare and are usually only available for a specific purpose, such as in fair as interactive product brochures. Devices are usually expensive and not suitable for consumer use. The aim was to develop a new multi-use system for multi-touch devices, suitable for multiple simultaneous users. The system is specially designed for equipment which construction is relatively cheap and easy.

Working was initiated with background research. In background research different multi-touch technologies were explored and different libraries, used in multi-touch software applications, were familiarized with. At the same time tools used in the work were chosen.

In this thesis a system intended for large multi-touch devices was created. The system is designed for multiple simultaneous users so that each user can run programs they need, or they can share them with other users. It consists of an application environment and user interface library, which can be used to create new applications to the system. Few applications were created to the application environment that can be used as an example and a basis for development of new applications.

SISÄLLYS

1 JOHDANTO



Kuva 1. Kuvakaappaus sovellusympäristöstä, jossa käynnissä kuva- ja internetselain.

Monikosketustekniikat ovat yleistyneet kuluttajatuotteissa huomattavasti viimeisen kolmen vuoden aikana. Suuremmat yrityskäyttöön tarkoitetut monikosketustekniikat ovat myös yleistyneet optisten tekniikoiden kehityttyä.

Työn tarkoitus oli luoda monelle yhtäaikaiselle käyttäjälle soveltuva sovellusympäristö ja käyttöliittymäkirjasto (Kuva 1) isoihin optisia monikosketustekniikoita käyttäviin laitteisiin. Opinnäytetyö tehtiin osana Satakunnan ammattikorkeakoulun tutkimus ja kehitystoimintaa Iiro Uusitalon rakentamaan optiseen monikosketuslaitteistoon.

Tämä opinnäytetyöraportti on pyritty kokoamaan niin, että raportti etenee matalan laitteistotason asioista ylöspäin päätyen itse demosovelluksiin. Näin lukija ymmärtää ylemmällä tasolla olevien käsitteiden toiminnan.

Seuraavassa luvussa käsitellään monikosketustekniikoita, joita voidaan käyttää monikosketuslaitteiston rakentamiseen. Kolmannessa luvussa käsitellään taustajärjestelmiä, jotka hoitavat kosketuksista saadun informaation käsittelyn ja välittämisen. Luvuissa neljä ja viisi kerrotaan ohjelmointityökaluista ja

ohjelmointiympäristöstä joilla opinnäytetyö luotiin. Luvussa kuusi käsitellään opinnäytetyössä luotua sovellusympäristöä monikosketussovellusten ajamiseen. Luku seitsemän käsittelee sovellusten kehittämistä varten luotua käyttöliittymäkirjastoa. Luvussa kahdeksan esitellään muutamia käyttöliittymäkirjastolla luotuja demosovelluksia.

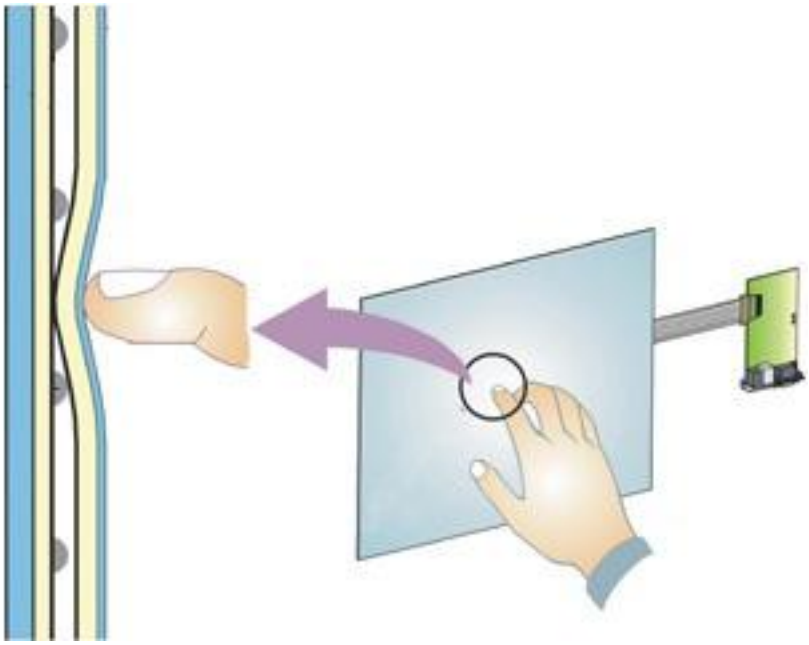
2 MONIKOSKETUSTEKNIIKAT

Monikoskestustekniikka ei ole uusi asia. Se on ollut olemassa eri muodoissa 1970-luvulta lähtien. Useat patentit (Pat. US 3,673,327; Pat. US 3,846,826; Pat. US 4,346,376; Pat. US 4,484,179) selittävät, miten kamera- tai sensoripohjaisen kosketuspinnan voi rakentaa. (Schöning, Brandl, Daiber, Echtler, Hilliges, Hook, Löchtefeld, Motamedi, Muller, Olivier, Roth & von Zadow 2008, 1)

Frustrated Total Internal Reflection (FTIR) -periaatteen uudelleen löytäminen vuonna 2005 Jeff Hanin toimesta vauhditti suuresti uusien monikosketussovellusten kehitystä. Hänen esittelemä kamerapohjainen järjestelmä oli sekä halpa, että helppo rakentaa ja sai suuren suosion harrastelijoiden keskuudessa. (Schöning ym. 2008, 1)

Applen vuonna 2007 julkaisema iPhone oli ensimmäinen monikosketustekniikkaa käyttävä kuluttajatuote ja sen saama julkisuus toi monikosketustekniikan kuluttajien tietoon. Vuotta myöhemmin Microsoft julkaisi oman monikosketustekniikkaa käyttävän Microsoft Surface -tuotteen, joka oli kohdistettu palvelualoilla oleviin yrityksiin, kuten ravintoloihin ja hotelleihin, sekä armeijan käyttöön. Vuonna 2009 Toshiba, Sony ja HP (List of Multi-Touch Computers and Monitors 2010) julkaisivat ensimmäiset monikosketustekniikkaa käyttävät kuluttajatietokoneet.

2.1 Resistiivinen

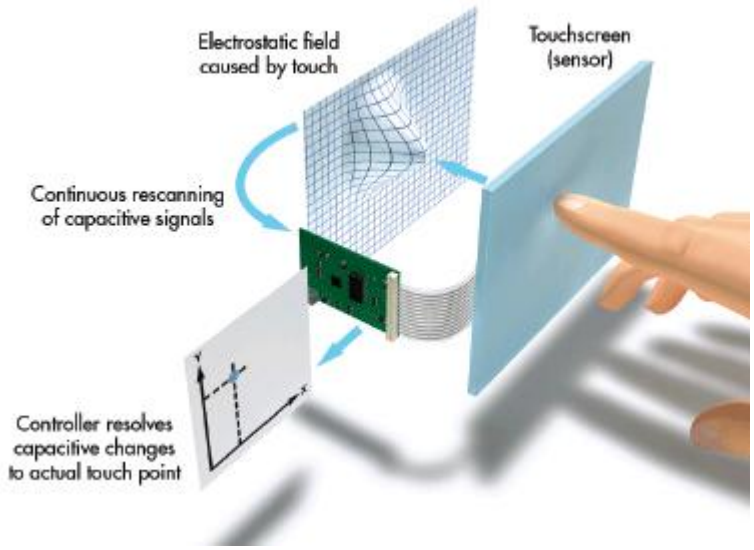


Kuva 2. Resistiivisen kosketuspinnan toimintaperiaate.

Resistiivinen kosketuspinta (Kuva 2) koostuu kahdesta joustavasta kalvosta, jotka ovat päällystetty virtaa vastustavalla materiaalilla. Kalvot on erotettu toisistaan käyttäen ilmaa tai mikropisteitä, jotka ovat jotain sähköä eristävää ainetta, kuten esimerkiksi piitä. Ohjain ohjaa vuorotellen kalvoja antaen niille tietyn virran ja tarkkaillen toisen kalvon virtaa. Kun kosketuspintaa painetaan, kalvot koskettavat toisiaan muodostaen virtapiirin, josta voidaan lukea vaaka- ja pystytason virtojen muutokset. (Schöning ym. 2008, 2)

Resistiivisen kosketuspinnan etuina on vähäinen energiankulutus, edullinen hinta ja toimivuus lähes kaikkien osoittimen kanssa. (Schöning ym. 2008, 2)

2.2 Kapasitiivinen



Kuva 3. Kapasitiivisen kosketuspinnan toimintaperiaate.

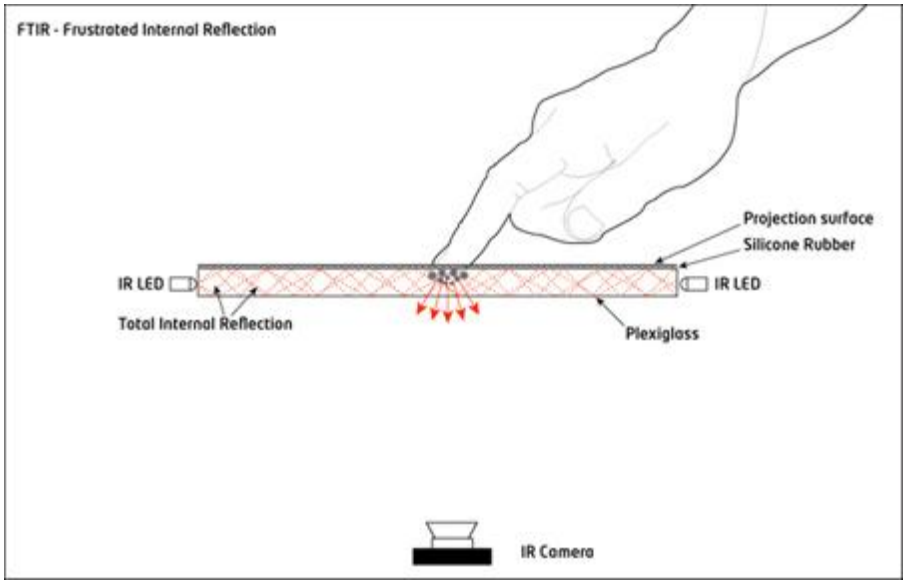
Kapasitiivisessa kosketuspinnassa (Kuva 3) on eristävä lasi, jonka toisen puolen pinta on päällystetty läpinäkyvällä johtavalla aineella tasaiseksi pinnaksi tai ruudukoksi. Kun lasin toiselle puolelle viedään jokin johdin, kuten ihmisen sormi, ne muodostavat kondensaattorin. Ohjain määrittelee sijainnin sensorien havaitsemien kapasitanssierojen avulla. (Schöning ym. 2008, 3)

Kapasitiiviset kosketuspinnat ovat resistiivisiä pintoja nopeatoimisempia ja rakenteeltaan kestävämpiä. Tekniikka häviää resitiivisielle tekniikalle valmistuskustannuksessa ja tarkkuudessa, sekä vaatii sähköä johtavan osoittimen, joten sitä ei pysty käyttämään esimerkiksi hanskojen kanssa. Valmistuskustannuksia nostaa myös vaatimus häiriöttömästä virtalähteestä tarkkuuden kasvattamiseksi. (Schöning ym. 2008, 3)

2.3 Optinen

Optiset järjestelmät ovat kaikki kamerapohjaisia ja toimivat samalla periaatteella, jossa kamerasta saatu kuva kosketuspinnasta suodatetaan ja analysoidaan.
Optisten järjestelmien peruskomponentteja ovat projektori tai LCD-paneeli, kamera eli sensori ja infrapunavalonlähde. Kuvattavan kosketuspinnan valaisutapoja on useita ja niistä yleisimpiä käsitellään tässä luvussa. Optisten järjestelmien ongelmana on suuri tilantarve optisen sensorin takia. Etuina ovat edullinen hinta-koko suhde ja mahdollisuus toteuttaa suuria yhtenäisiä kosketuspintoja taustaprojektion avulla.

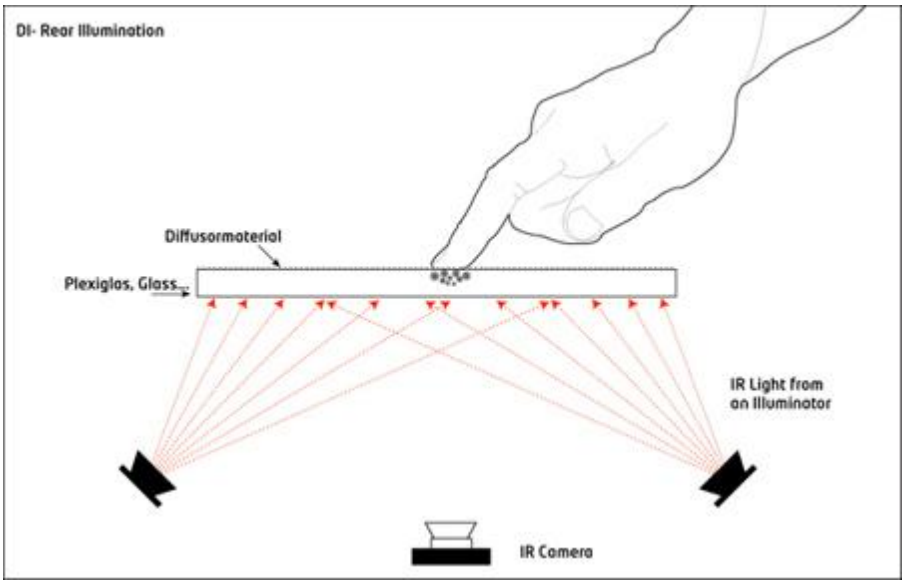
2.3.1 Frustrated Total Internal Reflection (FTIR)



Kuva 4. Optisen FTIR-tekniikan toimintaperiaate.

FTIR-tekniikka (Kuva 4) perustuu kosketuspinnan sisässä tapahtuvaan valon kokonaisheijastumiseen. Kosketuspinnan sisään lähetetty sähkömagneettinen säteily kokonaisheijastuu ulkopinnoilta, koska sisäisellä materiaalilla on suurempi taitekerroin kuin ulkopuolisella materiaalilla ja tulokulma on riittävän pieni. Tavallisissa FTIR-kosketuspinnoissa on läpinäkyvä akryylilevy, jonka kehyksen reunoilla on LED:ejä säteilemässä infrapunavaloa sen sisään. Kun käyttäjä koskettaa kosketuspintaa, valo pakenee ja heijastuu sormen pinnasta kosketuspintaa kuvaavaan kameraan. Koska akryylilevy on läpinäkyvä, voidaan sen taakse sijoittaa projektori ja muodostaa kuva taustaprojektiolla. (Schöning ym. 2008, 5)

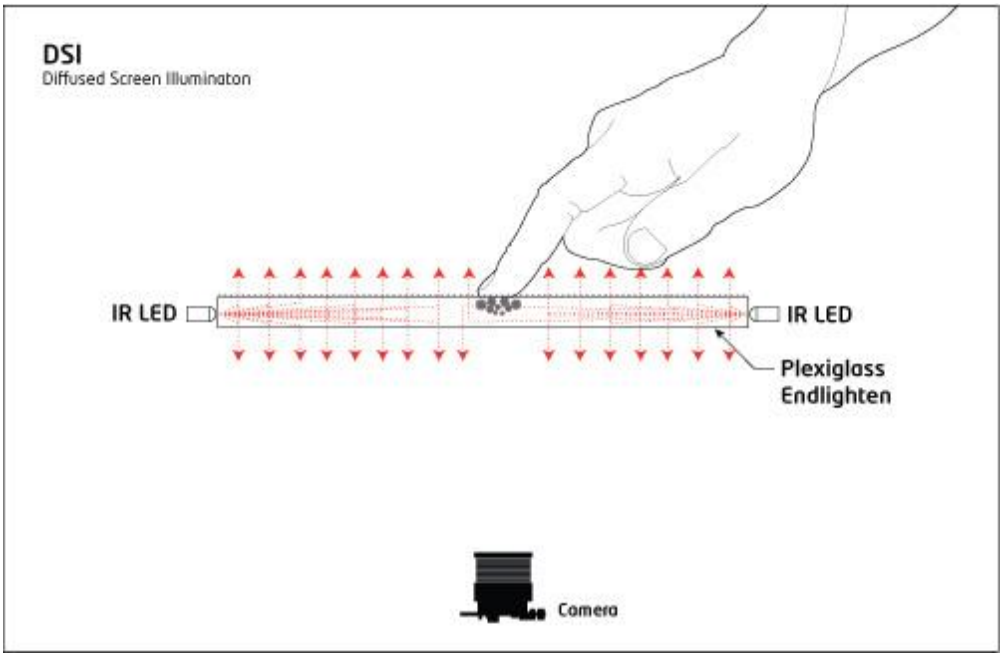
2.3.2 Diffuse Illumination (DI)



Kuva 5. Optisen DI-tekniikan toimintaperiaate.

DI-tekniikassa (Kuva 5) infrapunavalonlähde on sijoitettu kosketuspinnan taakse. Tämän johdosta kosketuspinnan edessä oleva tila on kirkkaasti valaistu, joten kamera havaitsee kaikki läheisyydessä olevat, että sitä koskettavat kohteet. Kosketuksen tunnistuksessa hyödynnetään kosketuspinnassa olevaa hajautinta, joka sumentaa kaukana olevat kohteet. Toisin kuin FTIR-tekniikka DI-tekniikka mahdollistaa esineiden seurannan sormien lisäksi. Tekniikka mahdollistaa turvalasin asentamisen kosketuspinnan ja projektiopinnan väliin, koska havainnointi ei ole riippuvainen kosketuksesta. (Schöning ym. 2008, 5)

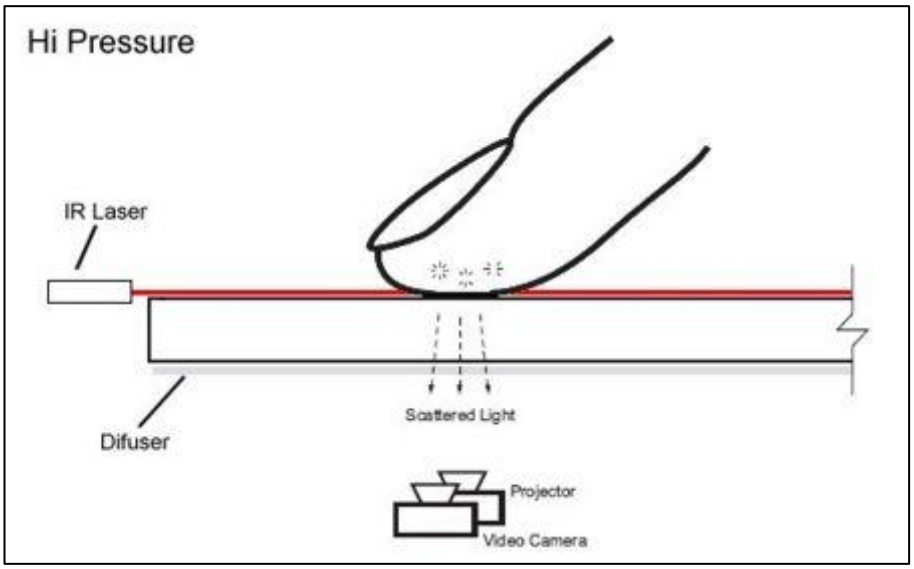
2.3.3 Diffused Surface Illumination (DSI)



Kuva 6. Optisen DSI-tekniikan toimintaperiaate.

DSI-tekniikka (Kuva 6) on FTIR-tekniikan ja DI-tekniikan sekoitus, jossa kirkas akryylilevy on vaihdetty erityiseen akryylilevyyn joka sisältää pieniä hiukkasia, jotka hajoittavat infrapunavalon. Infrapunavalonlähde on levyn reunoilla, kuten FTIR-tekniikassa. Tällä tekniikalla vältetään DI-tekniikassa oleva ongelma valon tasaiseksi saamisessa, mutta samalla menetetään osa kontrastista, joka saattaa aiheuttaa ongelmia taustalla olevan infrapunataustasäteilyn takia. (Schöning ym. 2008, 5)

2.3.4 Laser Light Plane Illumination (LLP)

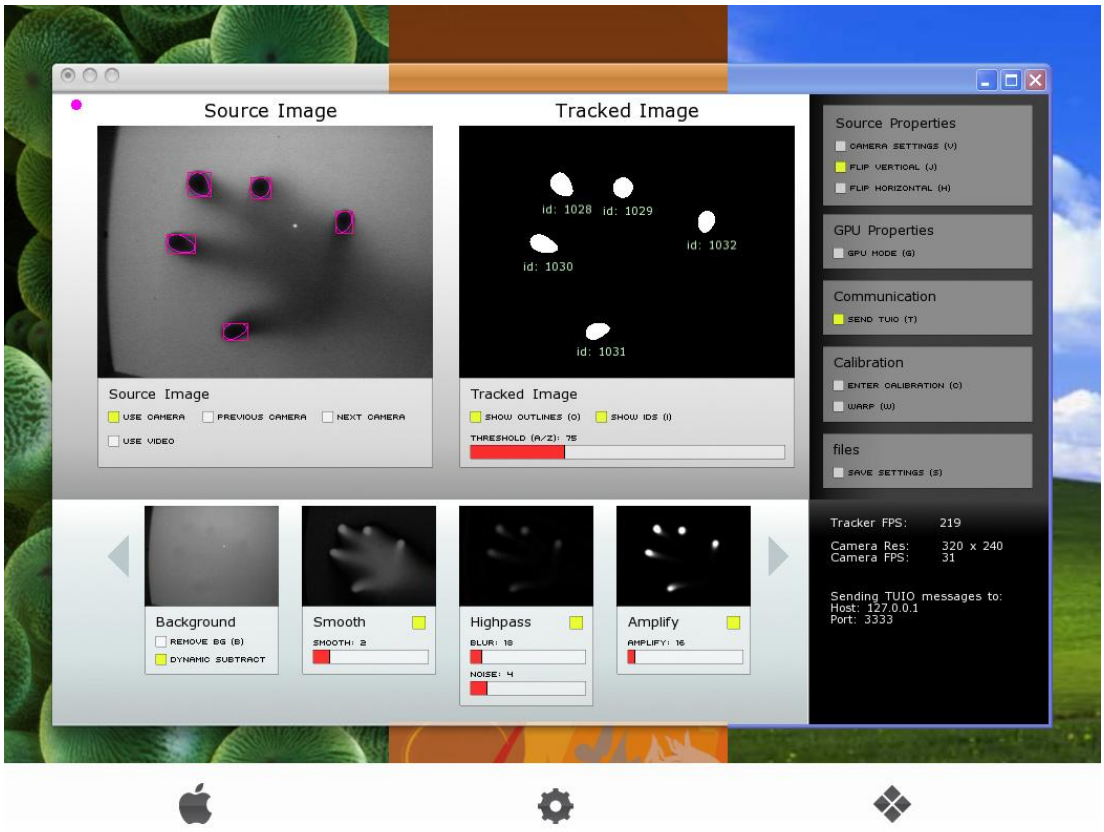


Kuva 7. Optisen LLP-tekniikan toimintaperiaate.

LLP-tekniikassa (Kuva 7) infrapunavalonlähde on toteutettu laserilla ja linssillä, joilla muodostetaan kosketuspinnan päälle ohut taso. Käyttäjän koskettaessa kosketuspintaa laserin valo heijastuu sitä kuvaavaan kameraan. (NUI Group Authors, 15)

3 TAUSTAJÄRJESTELMÄT

Optisissa monikosketustekniikoissa käytetään yleensä erillistä taustajärjestelmää, joka esikäsittelee kameralta saadun kuvan erilaisilla suodattimilla ja korjaimilla. Tämän jälkeen kuvasta analysoidaan seurattavat kohteet, kuten sormet tai esineet. Jälkikäsittelyssä kohteiden tietoja verrataan edellisiin tietoihin ja selvitetään mihin suuntaan kohteet ovat liikkuneet, onko uusia kohteita ja onko jokin kohde poistunut. (Schöning ym. 2008, 12)



Kuva 8. Kuvakaappauskooste Community Core Vision taustajärjestelmästä eri käyttöjärjestelmillä.

Kaupallisissa monikosketuslaitteissa, kuten Microsoft Surface:ssa ja suomalaisen MultiTouch:n laitteissa käytetään heidän omia suljettuja järjestelmiä. Tämän opinnäytetyön kehittämisessä käytettiin Community Core Vision (CCV) (Kuva 8) -taustajärjestelmää, jonka lähdekoodi on avoin. Muita avoimen lähdekoodin taustajärjestelmiä ovat muun muassa BBTouch, Bespoke Multi-Touch Framework, reacTIVision, Touché sekä Touchlib. Lähes kaikki taustajärjestelmät käyttävät tai

ainakin tukevat Tangible User Interface Object (TUIO) -protokollaa, jolla seurattavat kohteet ja niiden tiedot siirretään sovellukselle.

3.1 Tangible User Interface Object –protokolla

Tässä luvussa käsitellään TUIO 1.1 protokollaa. TUIO–protokolla on jatkuvasti kehityksessä ja siitä on lähitulevaisuudessa tulossa seuraavan sukupolven TUIO 2.0 –protokolla, joka korvaa ensimmäisen sukupolven versiot.

Protokollan tavoitteena on tarjota yleiskäyttöinen ja monipuolinen kommunikaatiorajapinta taustajärjestelmän ja sovelluksen välille. TUIO-protokollan joustava malli mahdollistaa lähetettävän tiedon sisällön määrittelemisen sovelluksen tarpeiden mukaiseksi. (Kaltenbrunner, Bovermann, Bencina & Costanza 2005, 1)

```
Viestit:
/tuio/[profiilinNimi] set tunniste [parameteriLista]
/tuio/[profiilinNimi] alive [lista tunnisteista]
/tuio/[profiilinNimi] fseq int32

2D Interactive Surface
      /tuio/2Dobj set s i x y a X Y A m r
      /tuio/2Dcur set s x y m r
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 m r
3D Interactive Surfaces
      /tuio/3Dobj set s i x y z a X Y Z A m r
      /tuio/3Dcur set s x y z m r
raw profile
      /tuio/raw_[profiilinNimi]
custom profile
      /tuio/_[parametrienMuoto]
      /tuio/_sixyP set s i x y 0.57       (esimerkki)

Parametrityypit:
s           tunniste, väliaikainen tunniste, int32
i           kohteen luokka, int32
x, y, z     normalisoitu sijainti, float32, välillä 0...1
a, b, c     orientaatio, float32, välillä 0..2PI
X, Y, Z     liikevektori (nopeus ja suunta), float32
A, B, C     kiertovektori (nopeus ja suunta), float32
m           liikkeen kiihtyvyys, float32
r           kierron kiihtyvyys, float32
P           vapaa muuttuja, tyyppi määritellään OSC otsikossa
```

Kuva 9. TUIO-protokollan syntaksimäärittely.

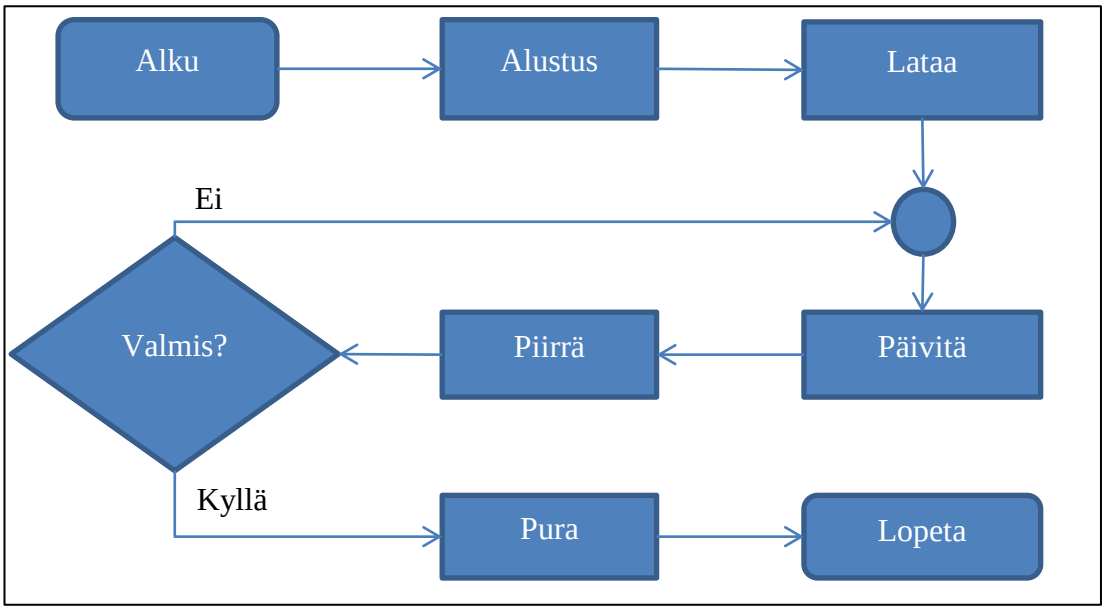
TUIO-protokolla määrittelee kolme pääviestiä: set, alive ja fseq viestit. Set viesti kertoo kohteen tietoja kuten tunnisteen, sijainnin, orientaation ja tilan. Alive viesti kertoo tällä hetkellä kosketuspinnalla olevien kohteiden yksilölliset tunnisteet. Fseq viesti sisältää järjestysnumeron, jonka mukaan set ja alive viestit järjestetään oikeaan järjestykseen. (Kaltenbrunner ym. 2005, 2)

Koska TUIO-protokolla käyttää alustana Open Sound Control (OSC) -protokollaa, se seuraa ennaltamäärättyä syntaksia (Kuva 9).

Eri profiilit määrittelevät mitä parametreja set-viesti sisältää. Esimääriteltyjä profiileja ovat 2D Interactive Surface (2Dobj), 2.5D Interactive Surface (25Dobj) ja 3D Interactive Surfaces (3Dobj). Esimääriteltyjen profiilien lisäksi voi lähettää itsemääriteltyjä viestejä käyttäen profiilinimen alussa raw_ tai _-etuliitettä. (Kaltenbrunner ym. 2005, 4)

4 MICROSOFT XNA

Microsoft XNA on kokoelma työkaluja, jotka helpottavat tietokonepelien kehittämistä ja hallintaa. Ohjelmointikirjasto sisältää laajan joukon luokkia, jotka ovat tarkoitettu Xbox 360 ja Windows peliohjelmointiin. Sitä voidaan käyttää kaikilla .NET yhteensopivilla ohjelmointikielillä, mutta vain C# on virallisesti tuettu. Ohjelmointikirjasto koteloi matalan tason tekniset yksityiskohdat ylemmän tason luokkiin ja pitää huolen eri alustojen yhteensopivuudesta. Tämä auttaa pelintekijöitä keskittymään sisältöön ja pelikokemukseen. Ajoympäristö on saatavilla Windows XP, Windows Vista, Windows 7 ja Xbox 360 järjestelmille. (Microsoft XNA 2010)

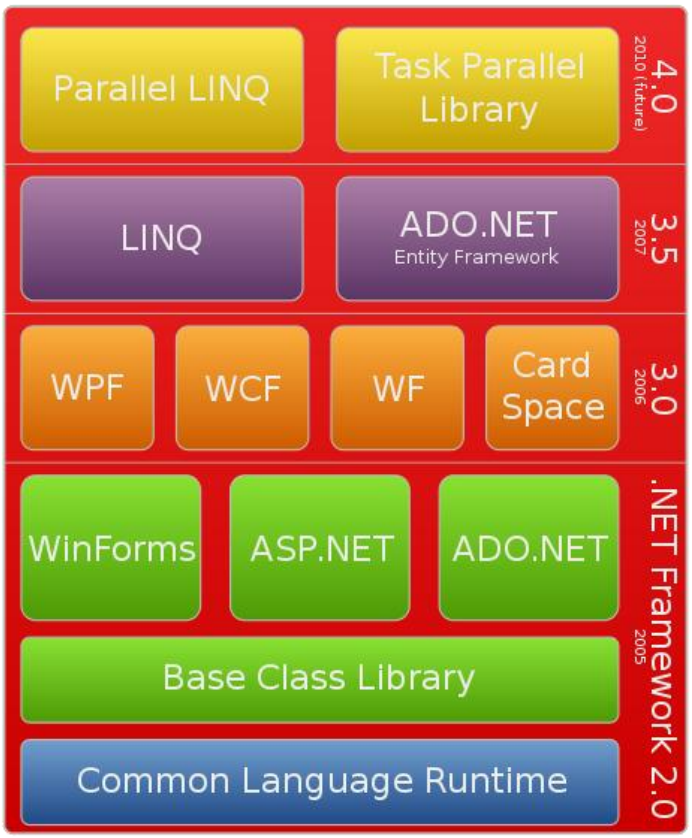


Kuva 10. XNA-ohjelmistokirjaston pääluuppi.

Kuva 10:ssä näkyy XNA-ohjelmointikirjaston pääluuppi, jonka päälle ohjelmisto rakennetaan.

5 MICROSOSOFT .NET-OHJELMISTOYMPÄRISTÖ JA C#-OHJELMOINTIKIELI

Microsoft .NET (Kuva 11) on sekä kehitysalusta Windows ohjelmien kehittämiseen että ympäristö niiden ajamiseen. Se sisältää ison ohjelmointikirjaston nimeltään Base Class Library (BCL), josta löytyy valmiita ratkaisuja yleisimpiin ohjelmoinnissa ilmeneviin tarpeisiin. Näitä ovat muun muassa käyttöliittymät, tiedon luku ja kirjoitus, tietokantaliittymä, salaus, numeeriset algoritmit ja tietoverkkokommunikaatio. Microsoft .NET:llä tehdyt ohjelmat ajetaan Common Language Runtime (CLR) -ympäristössä, joka on käytännössä virtuaalikone. Virtuaalikonetta käytettäessä ohjelmoijan ei tarvitse miettiä eri laitteiden tai suorittimien välisiä eroja vaan kaikki ohjelmat toimivat samalla tavalla jokaisella alustalla. Microsoft .NET tukee useita ohjelmointikieliä ja mahdollistaa niiden yhteistoiminnan, jossa eri kielet voivat suoraan toimia keskenään. (.NET Framework 2010)



Kuva 11. Microsoft .NET kehitysalusta.

C# on Microsoftin .NET -ympäristöä varten kehittämä ohjelmointikieli. Se on suunniteltu yksinkertaiseksi, moderniksi ja yleiskäyttöiseksi olio-ohjelmointikieleksi. Ensimmäinen versio ohjelmointikielestä julkaistiin .NET:in yhteydessa tammikuussa 2002. Kielen tavoitteena on yhdistää C++:n tehokkuus ja Java:n helppokäyttöisyys ja se mukailee vahvasti .NET:in Common Language Infrastructure (CLI) määritystä. (C Sharp 2010)

6 SOVELLUSYMPÄRISTÖ

Tähän työhön kehitetyssä monikosketussovellusympäristössä voidaan ajaa vain sille suunniteltuja ja tehtyjä sovelluksia. Ympäristö sisältää toimintoja ja palveluita joita siinä ajettavat ohjelmat tarvitsevat.

6.1 Hiekkalaatikko

Hiekkalaatikointi tarkoittaa ohjelmakoodin ajamista rajatussa ympäristössä. Microsoft .NET -ohjelmistoympäristö tarjoaa ohjelmoijalle sovellusalueen, jossa voidaan ajaa aliohjelmia halutuilla oikeuksilla. Käytännössä tämä tarkoittaa sitä, että aliohjelmalta estetään suora pääsy pääohjelmaan, sekä rajoitetaan ohjelman oikeuksia käyttää eri toimintoja. Sovellusalue myös eristää pääohjelman aliohjelmissa tapahtuvilta virheiltä ja poikkeustapahtumilta, jolloin pääohjelman toiminta ei vaaraannu vaikka aliohjelma olisi viallinen.

6.2 Monikosketussovellusmoduuli

Jokaisen sovellusympäristössä ajettavan sovelluksen tarvii olla käännetty dynaamisesti linkitettävään kirjastoon (Dynamic-link library tai DLL) ja sisältää pääluokassaan MultitouchProgram -määrittely (Kuva 12). Määrittely sisältää ohjelman yksilöllisen tunnisteen, nimen, lyhyen kuvauksen ja luokan johon se kuuluu.

```
namespace PhotoViewer {
[MultitouchProgram(
    "9E950D3B-0098-49A0-BF5E-A659CB8C21B5",
    Name = "Photo Viewer",
    Description = "View your photos",
    Category = MultitouchProgramAttribute.CATEGORY_MULTIMEDIA)]
    public class Main {
        private...
```

Kuva 12. Esimerkki MultitouchProgram-määrittelystä

Sovelluksen tulee kuulua yhteen järjestelmään määritetystä neljästä luokasta: pelit, multimedia, internet ja muut. Yksi moduuli voi sisältää useita sovelluksia.

6.3 Päävalikot

Kuva 13. Päävalikkopainike normaalissa ja aktivoidussa tilassa.

Kuvassa 13 vasemmalla on sovellusympäristön yksi neljästä painonapista, jota painamalla aukeaa oikealla näkyvä päävalikko. Päävalikot sijaitsevat sovellusalueen jokaisessa nurkassa ja ne ovat käännetty niin, että niiden käyttäminen onnistuu oikealla kädellä. Päävalikko koostuu neljästä järjestelmään määritetystä sovellusluokasta ja niihin liitetyistä sovelluksista.

6.4 Kehystenhallinta ja kehykset

Kehystenhallinta pitää kirjaa ja hallitsee järjestelmässä olevia kehyksiä (Frame). Kehystenhallinan tehtävä on myös välittää kosketustapahtumat oikeille kehyksille sekä käskyttää niiden päivityksen ja piirtämisen. Ohjelmoija voi vapaasti luoda uusia ja poistaa vanhoja kehyksiä. Kuva 14 on esimerkki uuden kehyksen luomisesta ja näyttämisestä.

```
this.window = new Frame(this.container, 300, 300, 1, 1,
                        Frame.DecorationType.Normal);

this.window.title = "Photo Viewer";
this.window.scaleMode = Frame.ScaleMode.Resize;
this.window.show();
```

Kuva 14. Ohjelmakoodi kehyksen luomiseen ja näyttämiseen.

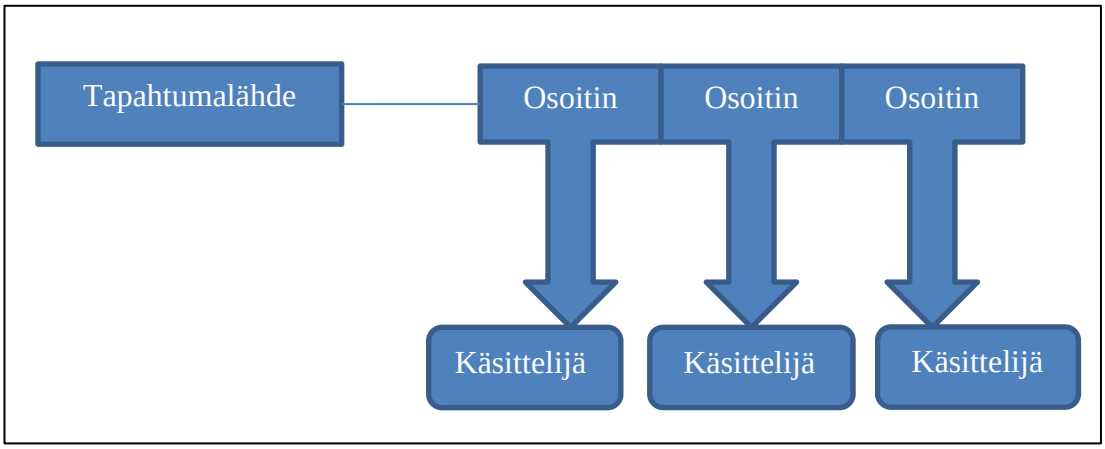
Kehyksissä on kolme eri reunakokoa joista ohjelmoija voi valita sopivan. Kehyksissä on viisi eri valikkoa, joista neljä sijaitsee kulmissa ja viides on päällysvalikko. Kuvassa 15 on esimerkki kulmassa olevasta valikosta. Päällysvalikko piirtyy kehyksen päälle jättäen alleen ikkunan sisällön. Valikoihin on asetettu oletuksena lopetus- ja virtuaalinäppäimistö-painikkeet. Painikkeiden sijainti on vapaasti valittavissa edellämainituista valikoista. Valikoihin voi vapaasti lisätä ja poistaa omia painikkeita. Kehykset periytyvät UIObject-luokasta, josta löytyy lisää tietoa osiosta 7.3 Komponentit.

Kuva 15. Kulmavalikko, jossa virtuaalinäppäimistön aktivointipainike.

6.5 Tapahtumajärjestelmä

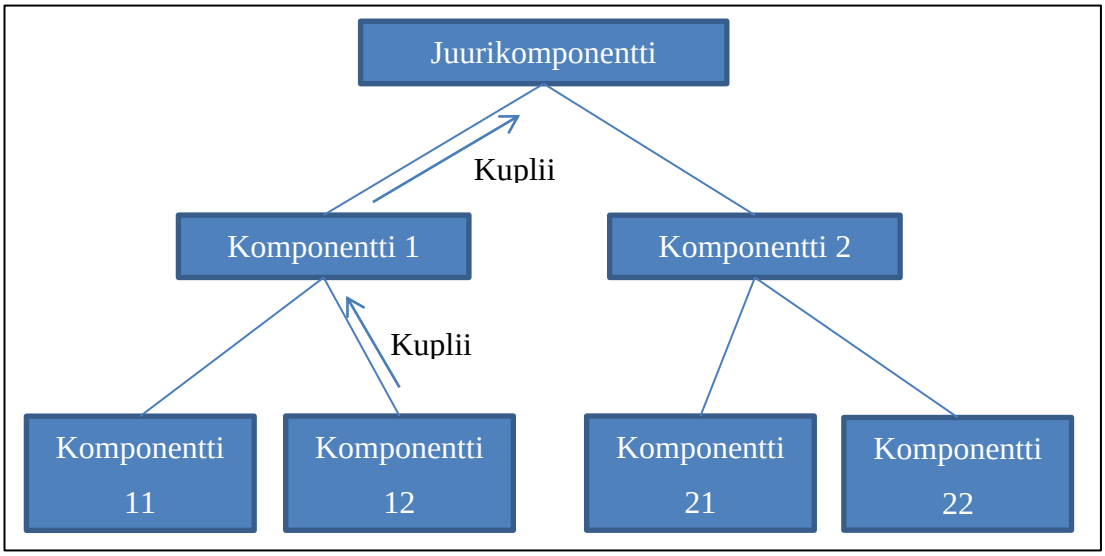
Sovellusympäristöön luotiin oma tapahtumajärjestelmä, joka on laajennus C#:n tapahtumajärjestelmään. C#:n tapahtumajärjestelmä perustuu delegaatteihin (Kuva 16), jotka ovat tyypitettyjä funktio-osoittimia. Tällaistä ratkaisua kutsutaan yleensä myös nimellä delegaatti-suunnittelumalli. Funktio-osoittimet eivät sovellu hyvin

käyttöliittymiin, koska komponenttihierarkiassa ylempänä olevilla komponenteilla ei ole suoraa pääsyä alempana oleviin.



Kuva 16. Delegaatin toimintaperiaate.

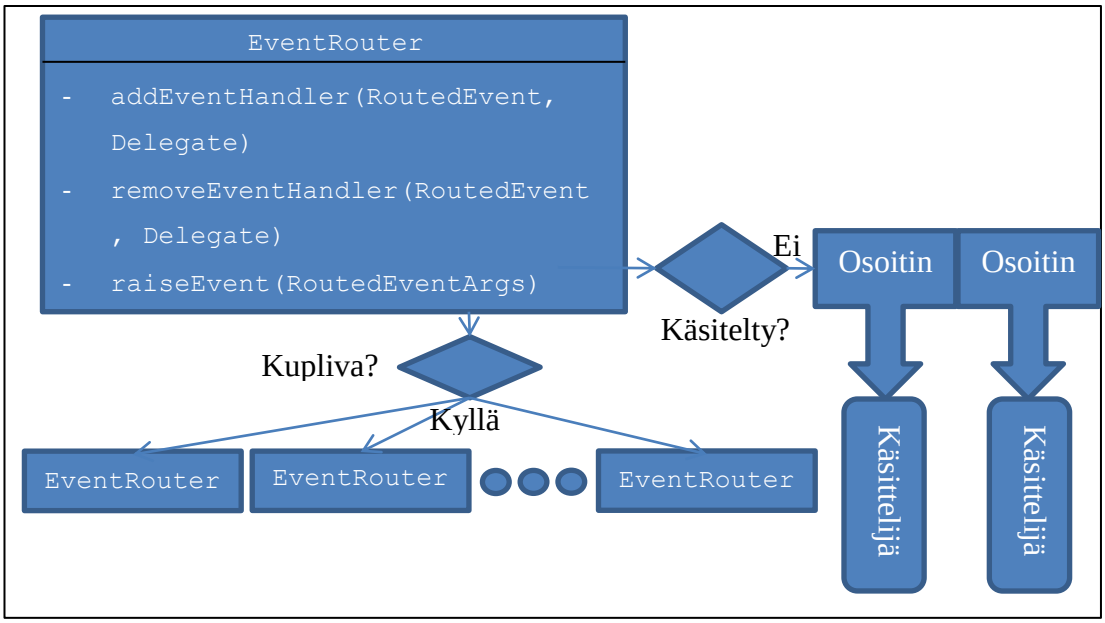
Sovellusympäristöön luotu tapahtumajärjestelmä on sekoitus delegaatti, tarkkailija ja viestin reititin-suunnittelumalleja. Näistä muodostuu kupliva tapahtuma-suunnittelumalli. Tällaistä suunnittelumallia käytetään useissa käyttöliittymäkirjastoissa, joista hyvinä esimerkkeinä voidaan mainita Windows Presentation Foundation (WPF) ja Adobe Flex.



Kuva 17. Esimerkki kuplivasta tapahtumasta.

Kuvassa 17 näkyy esimerkkitapahtuma, jossa komponentti 12 aktivoi kuplivan tapahtuman. Komponentti 12:een liitetyt tapahtumakäsittelijät aktivoidaan, jonka jälkeen tapahtuma kuplii komponentti 1:lle, jossa myös aktivoidaan kaikki

tapahtumaan liitetyt käsittelijät. Tapahtuma kuplii ylöspäin kunnes se on merkattu käsitellyksi tai se on saapunut juurikomponentille. Jos tapahtuma ei ole kupliva vain tapahtuman aktivoivaan komponenttiin liitetyt tapahtumakäsittelijät aktivoidaan.



Kuva 18. EventRouterin toimintaperiaate.

Tapahtumajärjestelmän toiminta perustuu pääosin EventRouter luokkaan. EventRouter (Kuva 18) sisältää assosiatiivisen taulukon tapahtumakäsittelijöistä, jossa avaimena on tapahtuman tyyppi ja arvona lista käsittelijöistä (delegaatti). EventRouterilla on myös RouteEvent-tapahtuma, johon komponenttihierarkiassa ylempänä olevat EventRouterit liittävät tapahtumakäsittelijänsä, joissa hoidetaan ylöspäin kuplinta. Tämä toiminta on tapahtumajärjestelmää ja käyttöliittymäkirjastoa käyttävälle ohjelmoijalle piilossa. Ohjelmoijalle näkyvillä ovat addEventHandler-, removeEventHandler- ja raiseEvent-metodit.

```
public delegate void TabEvent(BrowserTab source, TabEventArgs
args);

public static RoutedEvent titleChangeEvent = new
RoutedEvent("titleChange", typeof(TabEvent));

public event TabEvent TitleChanged {
      add { this.addEventHandler(titleChangeEvent, value); }
      remove { this.removeEventHandler(titleChangeEvent, value); }
}
```

Kuva 19. Esimerkki uuden tapahtuman luonnista tapahtumajärjestelmään.

Kuvassa 19 on esimerkki uuden tapahtuman luonnista tapahtumajärjestelmään, sekä sen paketoiminen C#:n delegaatti-suunnittelumallin näköiseksi. Paketointi ei ole toiminnan kannalta pakollinen, mutta tekee käsittelijöiden lisäämisestä mukavampaa. Aluksi määritellään TabEvent delegaatin tyyppi, esimerkissä sillä on kaksi parametria: lähde ja tapahtumaan liittyvä muuttujakokoelma. Tämän jälkeen määritellään uusi reititettävä tapahtuma, jonka nimi on titleChange ja delegaatin tyyppinä TabEvent. Lopussa kuuntelijan lisäys ja poisto paketoidaan C#:n delegaatti-suunnittelumallin näköiseksi.

```
this.raiseEvent(new TabEventArgs(titleChangeEvent));

this.raiseEvent(new UITouchEventArgs(pointDownEvent,
                projectedTouchPoint));
```

Kuva 20. Esimerkki titleChangeEvent- ja pointDownEvent-tapahtumien aktivoinnista.

Kuvassa 20 aktivoidaan kaksi tapahtumaa. Ensimmäinen on titleChangeEvent tyyppiä. Jälkimmäisenä aktivoidaan pointDownEvent-tapahtuma, jolla on parametrina projectedTouchPoint-olio. Tapahtuman aktivointi tapahtuu raiseEvent-metodilla. Se ottaa parametriksi tapahtuman muuttujakokoelman, joka sisältää tapahtuman tyypin ja siihen kuuluvat parametrit. Aktivoitaessa tapahtuma muuttujakokoelman originalSource-muuttujaan asetetaan alkuperäinen lähdekomponentti, jos tapahtuma kuplii ylöspäin kuplinnan jokaisessa vaiheessa source-muuttujaan asetetaan komponentti josta tapahtuma on saapunut.

Muuttujakokoelma välitetään jokaiselle tapahtumaa tarkkailevalle tapahtumakäsittelijälle.

```
this.htmlView.TitleChanged += new
HtmlView.HtmlViewEvent(htmlView_TitleChanged);

void htmlView_TitleChanged(HtmlView source,
                           HtmlViewEventArgs args) {
      . . .
}
```

Kuva 21. Tapahtumakäsittelijän asettaminen.

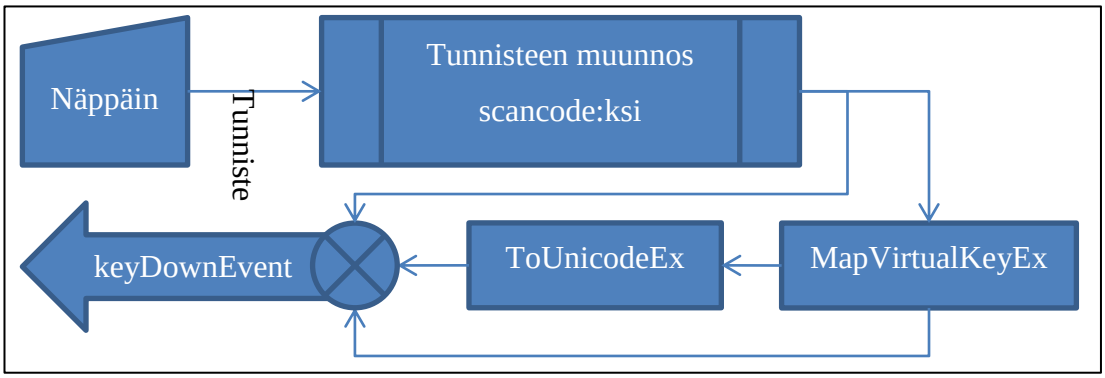
Kuvassa 21 asetetaan käsittelijä TitleChanged-delegaattiin, joka on oikeasti C#:n delegaatti-suunnittelumallin näköiseksi paketoitu liitäntä tapahtumajärjestelmään. Käsittelijä saa parametreikseen lähteen ja muuttujakokoelman.

6.6 Virtuaalinäppäimistö

Kuva 22. Virtuaalinäppäimistö.

Virtuaalinäppäimistö (Kuva 22) on käyttäjän kannalta yksi tärkeimmistä sovellusympäristön toiminnoista. Virtuaalinäppäimistö tukee amerikkalaista, kansainvälistä ja aasialaista näppäinasettelua. Näiden lisäksi käyttäjällä on käytössään kaikki Microsoft Windows:n tarjoamat näppäimistön kielet. Koska Microsoft .NET -kirjasto ei sisällä tarvittavia toimintoja monikielisyys on toteutettu käyttäen Windows USER komponenttia, joka löytyy user32.dll-kirjastosta. Komponentttia käytettiin C#:n Platform Invocation Services (PInvoke) toiminnolla,

jonka avulla voidaan kutsua hallitsemattomia funktiota dynaamisesti linkitettävästä kirjastoista. Tässä tapauksessa Windows USER -komponentista käytettiin MapVirtualKeyEx, LoadKeyboardLayout, UnloadKeyboardLayout, ToUnicodeEx, GetKeyNameText ja ActivateKeyboardLayout toimintoja. Käyttäjä voi vapaasti manipuloida virtuaalinäppäimistön kokoa, sijaintia ja kiertoa mieleisekseen.



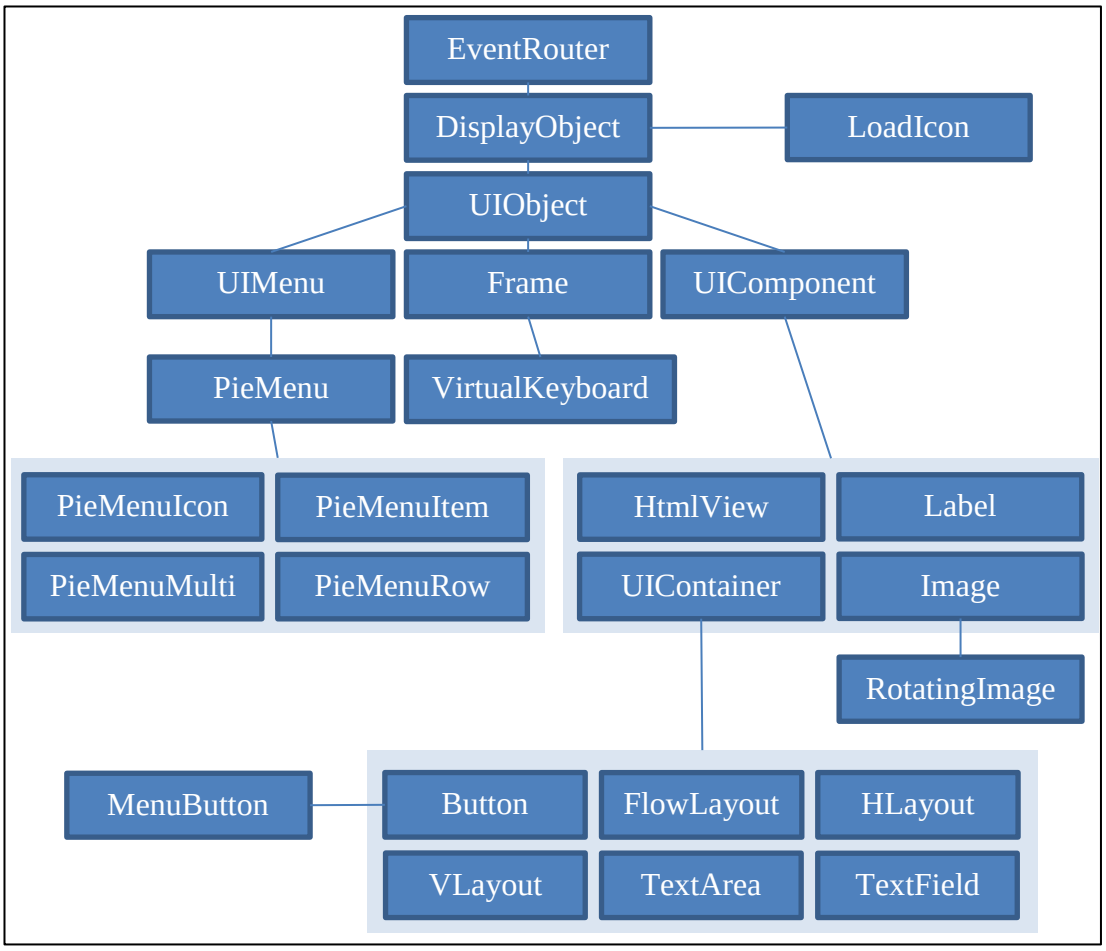
Kuva 23. Näppäimen tunnisteen käsittely.

Kuvassa 23 olevassa kaaviossa näkyy, kuinka näppäimen tunniste käsitellään ennen keyDownEvent-tapahtuman lähettämistä. KeyDownEvent sisältää scancoden, virtuaalisen näppäinkoodin ja syötettävän merkin tekstinä.

7 KÄYTTÖLIITTYMÄKIRJASTO

Käyttöliittymäkirjasto koostuu peruskomponenteista, joita ohjelmoija voi käyttää sovellusten käyttöliittymän luomiseen. Niitä voidaan käyttää sellaisenaan tai tarpeen mukaan laajentaa periyttämällä. Peruskomponentit toteuttavat käyttöliittymän luonnissa tarvittavia asioita kuten valikot, kuvat, tekstikentät, painonapit ja sijoitteluun liittyviä toimintoja.

7.1 Komponenttihierarkia



Kuva 24. Komponenttihierarkia.

Kuvassa 24 näkyy käyttöliittymäkirjaston komponenttihierarkia. Hierarkiassa ylimpänä on EventRouter, josta löytyy tarkemmin tietoa kohdasta 6.3 tapahtumajärjestelmä. EventRouterista seuraavana on DisplayObject, joka sisältää komponentin piirtämiseen tarvittavia tietoja. Ohjelmoijan kannalta kiinnostavin

komponentti on UIObject, joka käytännössä sisältää kaiken tarvittavan uusien käyttöliittymäkomponenttien luontiin. UIMenu sisältää tarvittavat toiminnot valikkohierarkioiden tekoon. Tähän käyttöliittymäkirjastoon luotiin pelkästään piirakkavalikko, josta kerrotaan tarkemmin kohdassa 7.3 komponentit. UIComponent sisältää komponenttien asettelua auttavia parametreja, joita muun muassa UIContainer ja sen alaluokat hyödyntävät. UIComponent sisältää myös virtuaalinäppäimistön syötteisiin ja komponentin piirtoon liittyviä toimintoja.

7.2 Kosketustapahtumat ja komponenttien manipulaatio

Taustajärjestelmältä tulleet kosketustapahtumat muutetaan XNA:n pääluupin päivitysmetodissa sovellusympäristössä käytettävään TouchPoint muotoon. Kosketustapahtumat annetaan kehystenhallinnalle (Kohta 6.3), joka välittää ne eteenpäin. Jokainen komponentti käsittelee sille tulevat kosketuspisteet ja välittää ne eteenpäin, mikäli sillä on lapsikomponentteja. Ennenkuin kosketuspisteet välitetään eteenpäin, ne muunnetaan välittävän komponentin omaan koordinaatistoon. Kosketuspisteellä on neljä eri tilaa: liikutettu, painettu, irroitettu. Sen tilan ja sijainnin muutoksen mukaan määräytyy minkä tapahtuman komponentti, jossa kosketuspiste sijaitsee, aktivoi. Jokainen komponentti toteuttaa hitTest-metodin, jolla määritetään missä komponentissa kosketuspiste sijaitsee.

```
// Lasketaan kahden ensimmäisen kosketuspisteen sijaintien välinen
// vektori.
Vector2 vec = handledTouchPoint[0].position -
              handledTouchPoint[1].position;

// Lasketaan edellisten sijaintien välinen vektori.
Vector2 lastVec = handledTouchPoint[0].lastPosition –
                  handledTouchPoint[1].lastPosition;

// Lasketaan komponentin kierto radiaaneissa.
double rotation = Math.Atan2(vec.Y, vec.X) –
                  Math.Atan2(lastVec.Y, lastVec.X);

// Kiertokulman asteluvusta muodostetaan quaternioni.
Quaternion quatRot = Quaternion.CreateFromAxisAngle(Vector3.UnitZ,
                                                    (float)rotation);

// Asetetaan origo ja lasketaan komponentin siirtymävektori.
Vector2 origin = handledTouchPoint[0].lastPosition;
Vector2 deltaOrigin = handledTouchPoint[0].position - origin;

// Kutsutaan muunnosmetodia.
this.transform(origin, vec, lastVec, quatRot, deltaOrigin);
```

Kuva 25. Komponentin manipulaatioparametrien laskenta-algoritmi.

Komponenttien manipulaatiolla tarkoitetaan niiden sijainnin, kulman ja koon muokkaamista. Kuvassa 25 on listattu ohjelmakoodi miten kahden kosketuspisteen avulla lasketaan kuinka komponenttia tulee manipuloida. Käyttöliittymäkomponenttien kokoa voidaan muokata venyttämällä komponenttia tai suurentamalla sitä. Kuvassa 26 on listattu ohjelmakoodi venytystapauksessa.

```
// Lasketaan nykyisten ja edellisten kosketuspisteiden
// etäisyyksien suhde.
float scale = vector.Length() / lastVector.Length();

// Haetaan world-matriisin determinantti.
float det = this._world.Determinant();

// Tarkastetaan että etäisyyksien suhde on järkevä.
if (float.IsInfinity(scale) || float.IsNaN(scale)
    || scale < 0.01 || (scale < 1f && det < 0.001))
      scale = 1f;

// Lasketaan uusi world-matriisi.
this._world *= Matrix.CreateTranslation(new Vector3(-origin, 0f))*
               Matrix.CreateScale(scale) *
               Matrix.CreateFromQuaternion(rotation) *
               Matrix.CreateTranslation(new Vector3(origin +
                                        originDelta, 0f));

// Lasketaan komponentin sisäisille muuttujille uudet arvot.
this._scale *= scale;
this._rotation += rotation;
this._translation = this.world.Translation;
```

Kuva 26. Komponentin sijainnin, venytyksen ja kierron laskenta.

Suurennettaessa koodi on lähes sama, mutta venytyksen sijasta muutetaan komponentin kokoa.

7.3 Komponentit

Komponenttien hierarkia löytyy Kuvasta 24. Tässä osiossa käsitellään komponenttihierarkiassa UIComponentin alla olevia komponentteja.

HtmlView-komponentti on HyperText Markup Languagen (HTML) ja muiden world wide web standardien näyttämistä varten. HtmlViewssä voidaan myös käyttää Flash- ja Java-pohjaisia sovelluksia. HtmlView perustuu Berkelium Sharp -kirjastoon, joka perustuu Googlen Chromium-selaimeen.

Label-komponentti on yksi- tai monirivisen tekstin näyttämiseen.

TextField- ja TextArea-komponentit ovat tekstin syöttöä ja muokkaamista varten. TextField-komponenttia käytetään silloin kun yksi tekstirivi riittää. TextArea-komponenttia voidaan käyttää monirivisessä tekstissä.

Image-komponenttia voidaan käyttää kuvien näyttämiseen. Komponentti osaa ladata kuvan sille annetusta Internet-osoitteesta tai sille voidaan antaa kuva bittikarttana.

Button-komponentti on tavallinen painonappi. Painonapille voidaan asettaa kuvake sekä teksti, jonka sijainti on valittavissa.

UIContainer on yläluokka kaikille komponenteille, joiden halutaan sisältävät lapsikomponentteja. Sitä voidaan myös käyttää sellaisenaan käyttöliittymien luomiseen. UIContainer ei ota kantaa lapsikomponenttien sijoitteluun, joten ne voidaan sijoitella vapaasti mihin kohtaan tahansa.

FlowLayout-komponenttia käytetään lapsikomponenttien automaattiseen sijoitteluun. Se pyrkii rivittämään sille annetut lapsikomponentit.

Hlayout- ja Vlayout-komponentteja käytetään listojen luomiseen. Ne sijoittelevat lapsikomponentit automaattisesti vaaka- tai pystyriviin. Komponentin nimen etukirjain merkitsee rivin suuntaa. H-kirjain nimen alussa tarjoittaa vaakariviä ja V-kirjain pystyriviä.

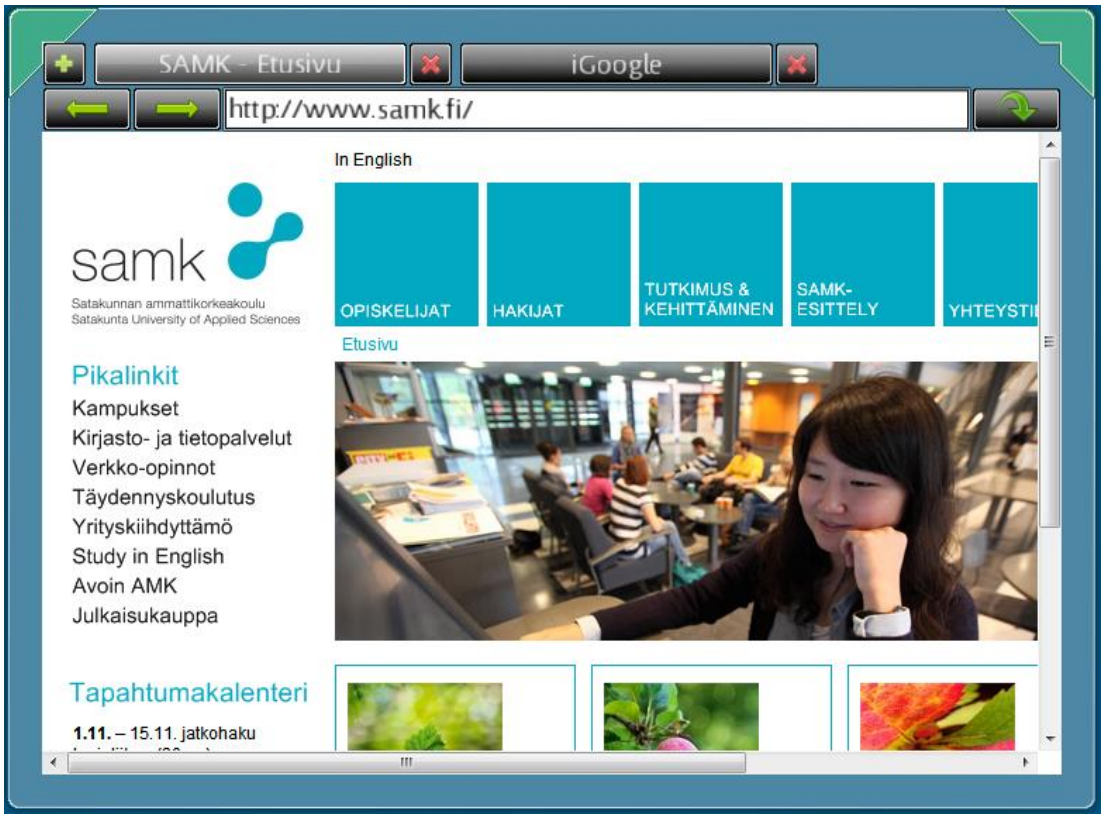
8 DEMOSOVELLUKSET

8.1 Kuvaselain

Kuva 27. Kuvakaappaus kuvaselaimesta.

Kuvaselain (Kuva 27) on yksinkertainen kuven katseluun tarkoitettu sovellus. Se osaa hakea kuvat tietokoneen kiintolevyltä sekä internetistä Flickr- ja Irc-galleria-sivustojen uusimmat kuvat. Esikatselukuvaa koskettamalla kuva aukeaa suurempana omaan kehykseen. Kuvan sijaintia, kokoa ja kiertoa voidaan vapaasti manipuloida. Kuvan päällä paikallaa sormea pidettäessä avautuu päällysvalikko, josta voidaan vaihtaa kuvan tietojen näkyvyyttä ja sulkea kuva.

8.2 Internetselain



Kuva 28. Kuvakaappaus internetselaimesta.

Internetselain (Kuva 28) on tehty mukailemaan perinteistä internetselaimen ulkonäköä. Yläosasta löytyy lista välilehdistä, joita voidaan lisätä painamalla vasemmassa reunassa olevasta plus-painikkeesta. Jokainen välilehti sisältää osoitekentän, päivityspainikkeen sekä seuraava- ja edellinen-nuolet, joilla voidaan liikkua sivuhistoriassa. Selaimen kokoa voidaan vapaasti suurentaa.

9 POHDINTA, TULOKSET JA JATKOKEHITYS

Markkinoille tulee jatkuvasti uusia monikosketustekniikoita ja –käyttöliittymiä hyödyntäviä mobiililaitteita. Älypuhelimien vauhdittamana taulutietokoneet ovat saapuneet kuluttajamarkkinoille. Näistä voidaan erityisesti mainita Applen Ipad sekä Samsungin Galaxy Tab, jotka molemmat perustuvat käyttöjärjestelmiltään ja -liittymiltään älypuhelimiin. Taulutietokoneita on ollut markkinoilla pitkään, mutta niiden huono kysyntä kuluttajamarkkinoilla on ainakin osaksi johtunut käyttöliittymän sopimattomuudesta kosketukseen perustuvaan käyttöön. Toisin kuin perinteiset hiirellä ja näppäimistöllä käytettävät käyttöjärjestelmät Applen iOS ja Googlen Android -käyttöjärjestelmät ovat alusta asti suunnteltu kosketusnäytöllisiin laitteisiin. Nämä järjestelmät eivät kuitenkaan sovi isoihin monikosketuslaitteisiin, joilla on mahdollisesti useita samanaikaisia käyttäjiä.

Tämän työn tarkoitus oli puuttua juuri muiden järjestelmien puutteeseen useista yhtäaikaisista käyttäjistä, jotka haluavat käyttää eri sovelluksia. Microsoftin Surface ja Suomalaisen MultiTouch Ltd:n järjestelmissä voidaan ajaa kerrallaan vain yhtä sovellusta. Työn kehitys onnistui hyvin ja siihen varatussa ajassa saatiin toteutettua tarvittavat perustoiminnot ja suppea käyttöliittymäkirjasto. Kaikkia haluttuja demosovelluksia ei kuitenkaan ehditty toteuttaa.

Sovellusympäristön ja taustajärjestelmän välissä käytettävä TUIO-protokolla tukee hallintaesineitä. Sovellusympäristö ei kuitenkaan osaa näitä erotella muista kosketuspisteistä ja tulkitsee ne normaaleiksi kosketuspisteiksi. Hallintaesineiden käyttöönotto mahdollistaisi monenlaisia toimintoja.

Suunnitelmissa oli myös luoda tuki Bluetooth-laitteille ja USB-muisteille, joiden avulla käyttäjät voisivat jakaa tiedostoja sekä ajaa omia sovelluksiaan järjestelmässä.

Ennen kehityksen alkua taustatutkimusta tehdessä grafiikkakirjastoksi valittiin Microsoft XNA. Kehitystyön edetessä huomattiin, että se ei sovellu kovin hyvin tällaisen järjestelmän tekoon ja siitä saadut edut eivät vastanneet sen aiheuttamia rajoituksia. Jatkokehityksen kannalta olisikin hyvä vaihtaa Microsoft XNA

perinteiseen Windows-sovellukseen ja käyttää grafiikkakirjastona suoraan Microsoft DirectX:ää. Kehityksessä käytettiin Microsoft XNA:n 3.1 versiota, joka ei mahdollistanut Windows 7:n monikosketus-rajapinnan käyttöä. Kyseisellä rajapinnalla järjestelmään pystyttäisiin tuomaan helposti kaikki Windows 7:n tukemat monikosketuslaitteet. Microsoft XNA:n uusin 4.0 versio tukee monikosketuspisteitä.

Järjestelmään oli aluksi tarkoitus luoda myös laajennus Qt-ohjelmistokirjastoon, jonka avulla Qt:llä luotuja ohjelmia voitaisiin helposti siirtää järjestelmään. Taustatutkimuksen aikana sen toteutus todettiin mahdolliseksi ja suhteellisen helpoksi. Kehitykseen varattu aika ei kuitenkaan riittänyt ominaisuuden toteuttamiseen.

Microsoft julkaisi marraskuussa 2010 Kinect -tuotteen Xbox 360 pelikonsolille. Kinect muodostaa kuvatusta alueesta joukon kolmiulotteisia pisteitä, joista voidaan muodostaa reaaliaikainen kolmiulotteinen malli. Kinectiä voidaan käyttää muun muassa esineiden-, kasvojen-, eleiden- ja puheentunnistukseen. Tämä avaa täysin uusia mahdollisuuksia tulevaisuuden käyttöliittymiin. Saattaa olla, että monikosketustekniikat jäävät vain mobiililaitteisiin ja suuremmat laitteet tulevat käyttämään tämänkaltaista järjestelmää osana käyttöliittymää.

LÄHTEET

Pat. US 3,673,327. 1972. Touch actuable data input panel assembly. Johnson Ralph G., Fryberger David, P. Appl. 05/086,011, November 2, 1970. Publ. June 27, 1972.

Pat. US 3,846,826. 1974. Direct television drawing and image manipulating system. Mueller Robert E., P. Appl. 05/323,874, January 15, 1973. Publ. November 5, 1974.

Pat. US 4,346,376. 1982. Touch position sensitive surface. Bell Telephone Laboratories, Incorporated. Mallos James B., P. Appl. 06/140,716, April 16, 1980. Publ. August 24, 1982.

Pat. US 4,484,179. 1984. Touch position sensitive surface AT&T Bell Laboratories. Kasday Leonard R., P. Appl. 06/333,744, December 23, 1981. Publ. November 20, 1984.

Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O., Hook, J., Löchtefeld, M., Motamedi, N., Muller, L., Olivier, P., Roth, T. & von Zadow, U. 2008. Multi-Touch Surfaces: A Technical Guide. Munich:in tekninen yliopisto. Tekninen raportti TUM-I0833. Viitattu 31.10.2010.
http://www.dfki.de/~jschoen/website/Publications_files/TUM-I0833.pdf

List of Multi-Touch Computers and Monitors. Wikipedia. Viitattu 31.10.2010.
http://en.wikipedia.org/wiki/List_of_Multi-Touch_Computers_and_Monitors

NUI Group Authors. 2009. Multi-Touch Technologies. Viitattu 3.1.2010.
http://nuicode.com/attachments/download/115/Multi-Touch_Technologies_v1.01.pdf

Kaltenbrunner, M., Bovermann, T., Bencina, R. & Costanza, E. TUIO - A Protocol for Table-Top Tangible User Interfaces. Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005). Vannes (Ranska). Viitattu 3.1.2010.
http://opensoundcontrol.org/files/tuio_gw2005.pdf

Microsoft XNA. Wikipedia. Viitattu 4.1.2010.
http://en.wikipedia.org/wiki/Microsoft_XNA

C Sharp. Wikipedia. Viitattu 4.1.2010.
http://en.wikipedia.org/wiki/C_Sharp_(programming_language)

.NET Framework. Wikipedia. Viitattu 4.1.2010.
http://en.wikipedia.org/wiki/.NET_Framework