Jaakko Hautamäki

Augmented Reality -tuki paikannukseen perustuvaan palveluun

Opinnäytetyö

Kevät 2013

Tekniikan yksikkö

Tietotekniikan koulutusohjelma

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Koulutusohjelma: Tietotekniikan koulutusohjelma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Hautamäki Jaakko

Työn nimi: Augmented Reality -tuki paikannukseen perustuvaan palveluun

Ohjaaja: Mäkelä Petteri

Vuosi: 2013 Sivumäärä: 45 Liitteiden lukumäärä: 0

Työn aiheena oli kehittää paikannukseen perustuva Augmented Reality sovellus Android-käyttöjärjestelmälle. Käyttäjä opastetaan lisättyä todellisuutta apuna käyttäen perille valituille kohteille. Sovellus tulee toimimaan osana toimeksiantajan kehittämää mobiiliohjelmistoa.

Augmented Reality eli lisätty todellisuus yhdistää todellisen ja virtuaalisen ympäristön interaktiiviseksi kokemukseksi. Lisättyä todellisuutta tarkastellaan työssä lähinnä sovelluskehittäjän, mobiilisovellusten ja oman työn näkökulmasta.

Työssä tutustutaan myös käytettyyn Android-käyttöjärjestelmään ja Android-sovellusten rakenteeseen. Lisäksi käyttöjärjestelmän yhteydessä esitellään ARToolKit kirjastoa ja sen seuraajia. Lopuksi työssä syvennytään toteutetun sovelluksen rakenteeseen ja sen olennaisimpiin toimintoihin.

Avainsanat: Lisätty todellisuus, Android, ARToolKit

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Software Development

Author: Hautamäki Jaakko

Title of thesis: Augmented reality support for a location based service.

Supervisor: Mäkelä Petteri

Year: 2013 Number of pages: 45 Number of appendices: 0

The aim of this thesis was to create a location based augmented reality application for the Android operating system. The application uses an augmented reality based view to guide the user to the point of interest.

Augmented reality mixes real world with virtual elements to create an interactive experience for the user. In this work augmented reality is viewed mainly from the points of view of mobile applications, the developers and the Android operating system.

Some of the most important augmented reality libraries are also introduced and the structure of Android applications and operating system are viewed briefly. Finally, the application that was created for this thesis is also introduced using examples that demonstrate its structure.

Keywords: augmented reality, Android, ARToolKit

SISÄLTÖ

Kuvio- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

A-GPS	A-GPS eli avustettu GPS (Assisted GPS) ydistää GPS-järjestelmän matkapuhelinverkkoon ja käyttää palvelimia apuna paikannuksessa.
GPS	Global Positioning System (GPS) on Yhdysvaltain puolustusministeriön kehittämä maailmanlaajuinen navigointi- ja paikannusjärjestelmä.
HMD	Head Mounted Display (HMD) eli päähän asetettava näyttö.
HHD	Hand Held Display (HHD) eli käsin pideltävä näyttö.
SAR	Spatial Augmented Reality (SAR) -tekniikan avulla projektoidaan tietokoneella muodostettua ja ympäristön kanssa vuorovaikutuksessa olevaa kuvaa todellisessa ympäristössä sijaitseviin kappaleisiin.
TTFF	Time To First Fix (TTFF) on aika, joka kuluu ensimmäisen GPS-signaalin vastaanottamiseen.
CAD	Computer-aided Design (CAD) eli tietokoneavusteinen suunnittelu.
XML	Extensible Markup Language (XML) on merkintäkieli, jota käytetään rakenteellisen tiedon välttämiseen.
SDK	Sofware Development Kit (SDK) on sovelluskehityspakkaus, joka sisältää tarvittavat työkalut sovellusten kehittämiseen.
LIFO	LIFO on tietorakenne, jonka alkioita lisätään ja poistetaan pinon päältä last-in-first-out periaatteen mukaisesti.
OpenGL	OpenGL on grafiikkaohjelmoinnissa käytettävä laitteistoriippumaton rajapinta.

1 Johdanto

1.1 Työn tausta

Työn aiheena oli aloittaminen paikannukseen perustuvan Augmented Reality -sovelluksen kehitystyö. Työ tulee olemaan osa toimeksiantajan jo olemmassa olevaa mobiiliohjelmistoa. Ohjelmisto on monipuolinen opastussovellus, joka toimii joukkoliikenteen reittioppaana. Työ käsittelee Augmented Realityä lähinnä Android-käyttöjärjestelmän ja sovelluskehityksen näkökulmasta.

Augmented Reality eli lisätty todellisuus kasvattaa jatkuvasti suosiotaan niin pelimaailmassa kuin hyötykäytössäkin. Todellisen ja virtuaalisen maailman yhdistäminen tuo sovellusten kehittäjille valtavasti uusia mahdollisuuksia, joita ollaan vasta löytämässä tai oppimassa hyödyntämään. Kasvavat käyttömahdollisuudet ja potentiaali on nykyään huomioitu etenkin mobiilimarkkinoilla. Älypuhelimet ja tablet-tietokoneet kasvattavat jatkuvasti suosiotaan ja tarjoavat erinomaisen alustan Augmented Reality -sovelluksille. Käyttömahdollisuuksista oli kiinnostunut myös työn toimeksiantajana toiminut yritys, joka on erikoistunut etenkin mobiilisovellusten kehitystyöhön.

1.2 Työn tavoite

Työn ensisijaisena tavoitteena oli aloittaa ohjeistuksen mukaisen Augmented Reality -sovelluksen kehitystyö. Paikannukseen perustuvan sovelluksen on opastettava käyttäjä perille etsityille kohteille. Puhelimen tarjoama reaaliaikainen kamerakuva ja lisätty todellisuus tulevat muodostamaan interaktiivisen kokonaisuuden. Sovelluksen on automaattisesti etsittävä käyttäjän läheisyydessä sijaitsevat kohteet ja havainnollistaa niiden etäisyydet käyttäjälle. Sovellus kehitetään toimeksiantajan ohjelmiston tavoin Android-käyttöjärjestelmälle.

1.3 Työn sisältö

Työn toinen luku käsittelee yleisesti Augmented Realityä, sen historiaa ja käyttömahdollisuuksia. Luvussa käsitellään myös tarvittavaa laitteistoa ja tavanomaisimpia toimintaperiaatteita. Kolmannessa luvussa esitellään hieman Android-käyttöjärjestelmää ja käydään läpi Android-sovellusten rakennetta kehittäjän näkökulmasta. Kappaleen lopussa esitellään tärkeimpiä Augmented Reality -kirjastoja. Neljäs luku keskittyy toteutettuun sovellukseen ja sen toimintaan. Augmented Realityn kannalta olennaisimpia toimintoja tarkastellaan sovelluksessa käytettyjen esimerkkien avulla.

2 Augmented Reality

Augmented reality (lisätty todellisuus) tarkoittaa todellisen ja virtuaalisen ympäristön yhdistämistä interaktiiviseksi kokonaisuudeksi. Virtuaalisen ympäristön kanssa vuorovaikutuksessa olevan käyttäjän aistihavaintoja pyritään muokkaamaan tietokoneen avustuksella. Virtuaalisella ympäristöllä ei siis aina tarkoiteta pelkästään näettävissä olevaa osiota, vaan sillä voidaan viitata myös muilla aisteilla tehtäviin havaintoihin. Sovelluskehityksessä Augmented Reality tarkoittaa usein kamerakuvan ja tietokonegrafiikalla luotujen elementtien reaaliaikaista yhdistämistä. Kamerakuva toimii ikään kuin ikkunana todelliseen maailmaan, jonka kehyksiin virtuaalinen ympäristö luodaan. (Carmigniani & Furth 2011, 3.)

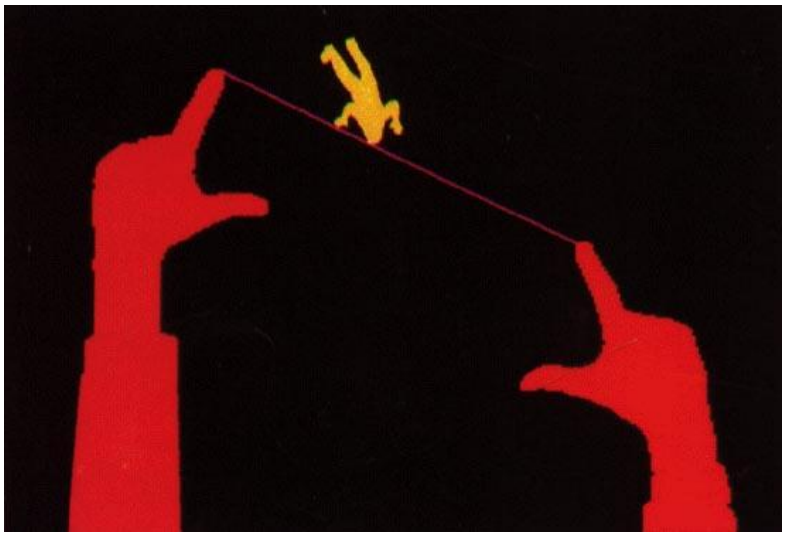
2.1 Historia

Ensimmäisen kerran Augmented Reality käsite esitettiin jo 1950-luvulla Morton Heiligin toimesta. Elokuva-alalla työskennellyt ja myöhemmin kirjailijana tunnettu Heilig keksi ajatuksen piirtää katsoja osaksi valkokankaan tapahtumia. Katsojan tulisi myös pystyä vaikuttamaan tapahtumiin kaikilla aisteillaan. Vuonna 1962 Morton patentoi Sensoramaksi nimeämänsä prototyypin ajatuksensa pohjalta. (Carmigniani & Furth 2011, 4.)

Kuvio 1. Sensorama Motion Picture Projector
(Sensorama Machine [Viitattu 12.5.2013].)

Vuonna 1966 Ivan Sutherland kehitti ensimmäisen päähän asetettavan näytön. Valtava näyttö oli liian raskas ihmisen kannettavaksi ilman kiinnitystä kattorakenteisiin. Vuotta aiemmin ilmestyi Sutherlandin kirjoittama Ultimate Display, joka oli mukana luomassa teoreettista pohjaa virtuaalitodellisuudelle. (Sung 2011.)

1970-luvun puolessa välissä Myron Krueger kehitti huoneen, jossa käyttäjä pystyi ensimmäistä kertaa olemaan vuorovaikutuksessa virtuaalisten esineiden kanssa. Kruegeria seurasivat lukuisat Augmented Realityä edistävät keksinnöt muun muuassa ilmailualan toimesta. (Carmigniani & Furth.2011, 4.)



Kuvio 2. Videoplace.
(Young 2010.)

Seuraava merkittävä askel otettiin professori Tom Caudellin ja David Mizellin toimesta. Auttaessaan työntekijöitä lentokoneiden sähkötöissä ja kaapelinasennuksessa he ottivat käyttöön termin Augmented Reality. Samana vuonna L.B Rosenberg kehitti yhden ensimmäisistä Augmented Reality Järjestelmistä nimeltä Virtual Fixtures. (Carmigniani & Furth 2011, 4.)

Ensimmäinen ulkonakin käytettävä Bruce Thomasin kehittämä mobiili Augmented Reality -peli ARQuake ilmestyi vuonna 2000. Pian ARQuaken jälkeen kehitettiin kameratekniikoita, joiden avulla pystyttiin analysoimaan ympäristöä reaaliajassa ja suhteuttamaan etäisyyksiä esineiden ja ympäristön välillä. (Carmigniani & Furth 2011, 5.)

Kuvio 3. ARQuake pelikuva.
(ARQuake: Interactive Outdoor Augmented Reality Collaboration System [Viitattu 12.05.2013].)

Seuraavan vuosikymmenen aikana Augmented Reality sovelluksia alettiin kehittämään yhä kiihtyvässä tahdissa etenkin mobiililaitteille. Lisätty todellisuus huomioidaankin nykyään yhä useammin uusien mobiililaitteiden kehitystyössä. Vasta älypuhelimien yleistyminen ja kehittyminen toi Augmented Realityn kaikkien ulottuville ja mahdollisti sen läpimurron. (Carmigniani & Furth 2011, 5.)

2.2 Käyttömahdollisuudet ja tulevaisuuden näkymät

Tietokoneiden ja pelikonsolien lisäksi useimpien älypuhelimien suorituskyky on nykyään riittävä Augmented Reality -pohjaisten sovellusten pyörittämiseen. Älypuhelin onkin erinomainen alusta sovelluksille, sillä se sisältää jo valmiiksi tarvittavan laitteiston ja kulkee käyttäjän mukana. Yleinen harhaluulo onkin, että Augmented Reality kehitettiin vasta älypuhelimien ilmaantumisen jälkeen. Yhtenä lupaavimmista alustoista älypuhelimien lisäksi pidetään tablet-tietokoneita. Niistä löytyy älypuhelimien tapaan valmiina tarvittava laitteisto ja samat käyttöjärjestelmät. Yksi suurimmista eduista älypuhelimiin verrattuna on niiden isompi näyttö ja parempi suorituskyky. (Carmigniani & Furth 2011, 10-11.)

Lisätty todellisuus on erinomainen apu yrityksille tuotekehityksen yhteydessä. Uudesta tuotteesta on mahdollista luoda entistä kokonaisvaltaisempi kuva virtuaalisella prototyypillä. Tuotetta ei ainoastaan mallinneta kolmiulotteisesti, vaan sille voidaan lisätä myös toiminnallisuuutta. Lisättyä todellisuutta voidaan tulevaisuudessa käyttää apuna esimerkiksi kokonaisten tuotantolinjastojen ja toimistotilojen suunnittelussa. Toinen yrityksille potentiaalinen käyttötarkoitus on markkinointi. Asiakkaalle olisi mahdollista toimittaa interaktiivisia luonnoksia yrityksen tuotteista. Eräs jo toimivaksi todettu tapa on käyttäjille ladattavissa oleva mobiilisovellus, jonka avulla voidaan tarkastella virtuaalisia mallinnuksia eri tuotteista. Yksi tähän asti menestyksekkäimmistä sovelluksista onkin mentaion kehittämä digitaalinen laatikko LEGO©-rakennuspalikoille. Myyntipisteissä oleva tietokone tunnistaa pakkauslaatikon ja käyttää sitä alustana tuotteen kolmiulotteiselle Augmented Reality mallinukselle. Digitaalinen pakkauslaatikko on esiteltynä kuviossa 4. (Carmigniani & Furth 2011, 12-13.)

Kuvio 4. Digitaalinen legopaketti.
(Lego Digital Box Augmented Reality Kiosk [Viitattu 12.5.2013].)

Kulttuuri- ja viihdeteollisuus tarjoavat myös lupaavia käyttömahdollisuuksia lisätylle todellisuudelle. Nykyään on olemassa muutamia museoita, jotka käyttävät

Augmented Reality -pohjaisia mobiilisovelluksia apuna vierailijoiden opastamisessa ja multimediapohjaisissa esittelyissä. Muotimaailmassa on jo käytössä peilin lailla toimivia näyttöjä, jotka mahdollistavat vaatteiden sovittamisen ilman vaatteiden vaihtamista. Myös opetus- ja rakennusalalla käytetään jo olemassa olevia sovelluksia apuna esimerkiksi suunnittelutyössä ja interaktiivisissa harjoituksissa. Peliteollisuus tarjoaa myös valtavasti uusia kehitysmahdollisuuksia. Lisätyn todellisuuden avulla käyttäjä voidaan tuoda osaksi pelikokemusta. Käyttäjä ei ole enää sidottuna tiettyyn paikkaan vaan pystyy omilla eleillään vaikuttamaan pelin tapahtumiin. (Carmigniani & Furth 2011, 29-30.)

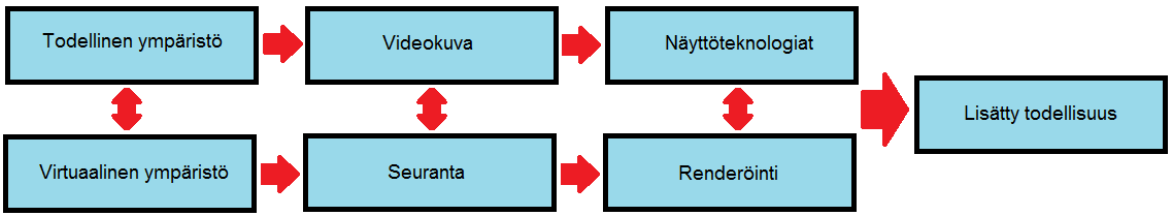
Myös lääketieteen alalta löytyy käytössä olevaa tietokoneohjattua huipputeknologiaa. Useissa tutkimuksissa yritetään selvittää voiko lisättyä todellisuutta käyttää apuna leikkausoperaatioissa. Potilaan tutkimusten yhteydessä suoritetuista kuvauksista voidaan muodostaa kolmiulotteinen mallinnus tutkitusta kohteesta. Mallinnusta käyttämällä leikkaussalissa olevalta monitorilta olisi mahdollista nähdä potilaan ihon alle. Mallinnusten avulla voitaisiin myös vähentää turhien leikkausten määrää. (Carmigniani & Furth 2011, 33-35.)

Augmented Realityn luomia mahdollisuuksia hyödynnetään myös sotilaskäytössä, joka on yksi alan tärkeimmistä edelläkävijöistä. Alalla tehdään jatkuvaa tutkimustyötä ja käytetään hyväksi jo olemassa olevia sovelluksia. Lisättyä todellisuutta käytetään apuna esimerkiksi sotilasoperaatioden suunnittelussa ja harjoittelussa. (Livingston ym. 2011, 671-672.)

Lukuisista muistakin käyttömahdollisuuksista huolimatta Augmented Reality on nykyisin arkikäytössä lähinnä pelien ja viihdesovellusten muodossa. Augmented Realityn tulevaisuus on paljolti riippuvainen sosiaalisesta hyväksynnästä ja käyttäjän yksityisyyden säilymisestä. Uusien laitteiden täytyy tarjota käyttäjälle uusi tekniikka muodinmukaisessa ja yleisesti hyväksyttävässä muodossa. (Carmigniani & Furth 2011, 38-40.)

2.3 Toimintaperiaate

Augmented Reality tavoittelee graafisten elementtien yhdistämistä todelliseen ympäristöön niin uskottavasti, että käyttäjä ei enää tunnista eroa todellisen ja virtuaalisen ympäristön välillä Sovellusten toiminta voidaan tarvittaessa jakaa kolmeen eri osa-alueeseen: käyttäjän seurantaan (eng. tracking), näyttötekniikoihin ja reaaliaikaiseen renderöintiin. (Bimber & Raskar 2005, 5-7.)



Kuvio 5. Tyypillinen Augmented Reality -sovelluksen rakenne

Useimpien sovellusten toiminta perustuu pitkälti laitteen asennon tunnistamiseen ja konenäköön. Moni sovellus vaatii lisäksi ajakohtaista tietoa käyttäjän sijainnista. Yhdistämällä tiedot laitteen sijainnista ja orientaatista saadaan jo melko tarkka käsitys käyttäjän sijoittumisesta todellisessa ympäristössä. Mitä tarkemmin laitteen asento ja sijainti pystytään määrittelemään, sitä uskottavammin lisätty todellisuus voidaan toteuttaa. Augmented Realityn yhteydessä laitteen asennon ja sijainnin tarkkailemisesta käytetään usein termiä seuranta. Seuranta kattaa myös konenäön ja muut sovellusten käyttämät syöttölaitteet. (Carmigniani & Furth 2011, 23-24.)

Oikean maailman ja virtuaalimaailman uskottava yhdistäminen vaatii tiedon käyttäjän katselukulmasta eli suunnasta mihin käyttäjä katsoo oikeassa maailmassa. Katselukulman avulla voidaan määritellä oikeassa maailmassa sijaitsevan pisteen sijainti laitteen näytöllä ja se toimii myös näkökulmana virtuaalisille objekteille. Useimmiten katselukulma vastaa laitteen sen hetkistä asentoa. Kolmiulotteisessa koordinaatistossa sijaitsevien pisteiden siirtäminen laitteen kaksiulotteiselle näytölle onkin yksi Augmented Reality -sovellusten keskeisimmistä toiminnoista. (Carmigniani & Furth 2011, 6.)

Kamerakuva vastaa sovellukselle käyttäjän näkemystä ympäröivästä maailmasta. Virtuaalinen maailma renderöidään reaaliaikaisesti seurannasta saatujen tietojen

pohjalta kamerakuvan päälle. Tavoitteena on aina mahdollisimman todenmukainen ja aidolta vaikuttava mallintaminen. (Bimber & Raskar 2005, 5.)

Konenäköä käyttävät sovellukset toimivat harvoin täysin itsenäisesti ilman seurantaan käytettävien laitteiden apua. Mikäli sovellukset eivät käytä konenäköä toimii kamerakuva lähinnä taustana virtuaalisille elementeille. (Carmigniani & Furth 2011, 8.)

2.4 Laitteisto ja seuranta

Kaikki Augmented Reality -sovellukset tarvitsevat toimiakseen ainakin jonkin keskusyksikön, näytön ja laitteen asentoa tarkkailevan anturin. Yleisimmät käyttäjän seurantaan käytetyt laitteet ovat kiihtyvyysanturi, kompassi ja gyroskooppi. Usein kompassilla tarkoitetaan nykyään magnetometriä. Sijainnin määrittämisessä käytetään apuna GPS ja A-GPS tekniikoita. (Carmigniani. & Furth 2011, 9-12.)

2.4.1 Näytöt

Augmented Reality näytöt voidaan karkeasti jakaa päässä pidettäviin näyttöihin (HMD Head Mounted Display), käsin pideltäviin näyttöihin (HHD Hand Held Display) ja spatiaalisiin näyttöihin. (Carmigniani & Furth 2011, 9.)

Päässä pidettävät näytöt toimivat peittämällä käyttäjän näkökentän. Näytöt voivat olla läpinäkyviä, läpinäkymättömiä tai pieniä projektoreja, jotka muodostavat kuvan käyttäjän verkkokalvoille (Retinal Display). Tyypillinen ratkaisu on käyttää laseja tai kypärää, joka kulkee käyttäjän mukana. Läpinäkyvä eli optinen näyttö mahdollistaa pelkästään lisätyn todellisuuden piirtämisen näytölle ja muun ympäristön jättämisen koskemattomaksi. Optisten näyttöjen käyttö on lisääntänyt huomattavasti viime vuosikymmenen aikana. Ne ovat lähestulkoon syrjäyttäneet tavanomaiset läpinäkymättömiä videonäyttöjä käyttävät ratkaisut. Optisten näyttöjen yleisimpänä ongelmana pidetään virtuaalisesti lisättyjen objektien huonoa sulautuvuutta muuhun ympäristöön. (Bimber & Raskar 2005, 72-76.)

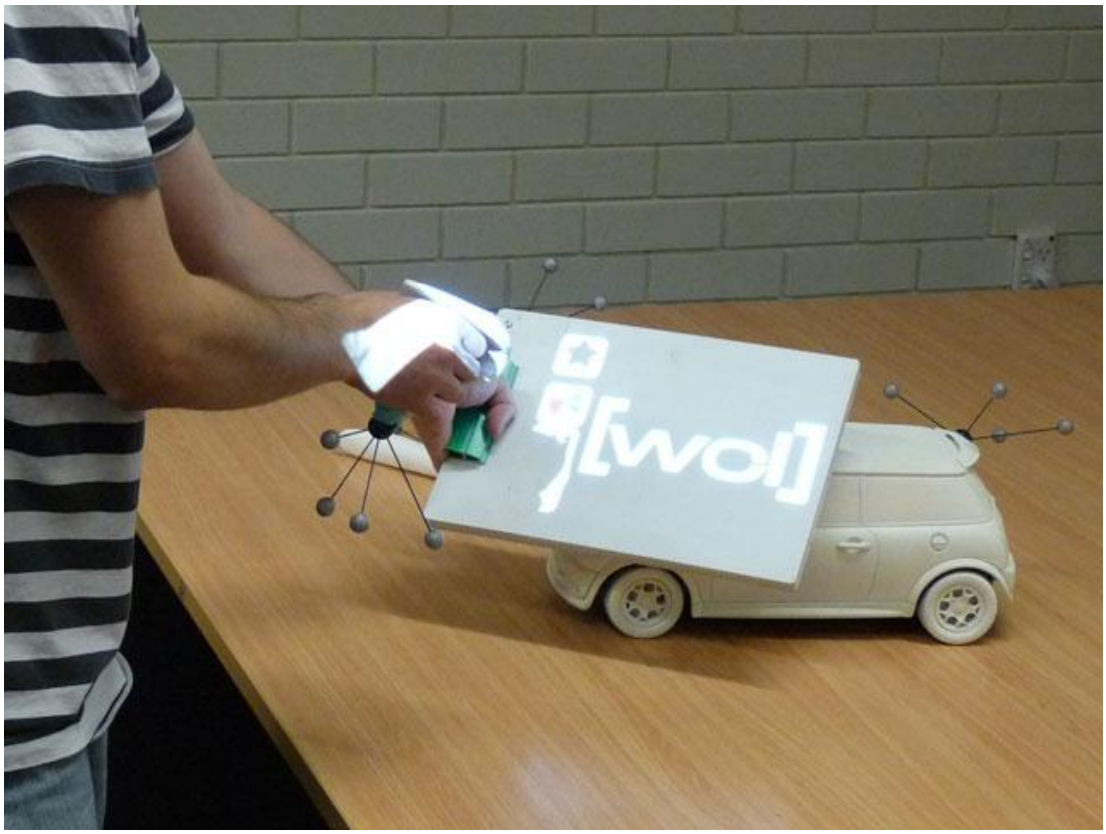
Kuvio 6. Google Glass Augmented Reality -älylasit.
(Rivington 2013.)

Etenkin kädessä pideltäville näytöille tyypillinen ratkaisu on käyttää kameraa läpinäkymättömän näytön ulkopuolella. Käyttäjä näkee ainoastaan kameran välittämän kuvan ympäristöstä, joka voi muodostua ongelmaksi etenkin päässä pideltävien näyttöjen kohdalla. Läpinäkymättömiä näyttöjen yleisimmät haasteet ovat kamerasta aiheutuvat viiveet ja syvyysnäön heikkeneminen. Hyvänä esimerkkinä mainittakoon matkapuhelimet ja tablet tietokoneet. (Bimber & Raskar 2005, 79-83.)

Kuvio 7. Paikannukseen perustuva Nokia City Lens -mobiilisovellus.
(Bonetti 2012.)

Spatial Augmented Reality (SAR) -tekniikan avulla projektoidaan tietokoneella muodostettua ja ympäristön kanssa vuorovaikutuksessa olevaa kuvaa todellisessa

ympäristössä sijaitseviin kappaleisiin. Spatiaalisilla näytöillä tarkoitetaan useimmiten kiinteästi paikallaan pysyviä näyttöjä tai videoprojektoreja. Projektorin tai näytön pysyvä sijainti mahdollistaa laitteiston sulauttamisen ympäristöön ja käyttäjän vapauttamisen seurantaan tarvittavasta laitteistosta. Se mahdollistaa myös tarkkuutta vaativien sovellusten toteuttamisen huomattavasti helpommin kuin käsin pideltävien tai päässä pideltävien näyttöjen tapauksessa. Tekniikka soveltuu käytettäväksi esimerkiksi videoprojektorin, verkkokameran tai minkä tahansa liikkumattoman näytön yhteydessä. Spatiaaliset näytöt kasvattavat jatkuvasti suosiotaan etenkin yritysmaailmassa ja lääketieteen parissa. (Carmigniani & Furth 2011, 11.)



Kuvio 8. Videoprojektoreja käyttävä SAR sovellus.
(Interactive SAR [Viitattu 12.05.2013].)

2.4.2 Asennon ja liikkeen tunnistaminen

Mobiilisovellukset käyttävät lähes aina kiihtyvyysanturia laitteen asennon tunnistamiseen. Kiihtyvyysanturin käyttö on lisääntynyt huomattavasti viimeisen vuosikymmenen aikana ja uusia käyttömahdollisuuksia löydetään jatkuvasti.

Kiihtyvyysanturi mittaa laitteen kiihtyvyyttä yhdellä tai useammalla akselilla. Nykyaikaisissa matkapuhelimissa käytetään useimmiten kiihtyvyysanturia, joka mittaa muutoksia kaikilla kolmella akselilla. (Techopedia explains Accelerometer [Viitattu: 21.05.2013].)

Laitteen asento kolmiulotteisessa tilassa määritellään usein käyttämällä termejä roll (kallistuminen), yaw (kääntyminen) ja pitch (nyökkääminen). Roll vastaa kääntymistä pituusakselin, yaw pystyakselin ja pitch poikkiakselin suuntaan. (Benson 2008.)

Toinen yleisesti Augmented Reality sovelluksissa käytetty laite on magnetometri. Magnetometriä käytetään mittaamaan muutoksia magneettikentän voimakkuudessa ja suunnassa. Magnetometri toimii kompassin tavoin ja sen avulla voidaan selvittää magneettinen pohjoissuunta. Usein Augmented Reality -sovellusten kohdalla pelkkä kiihtyvyysanturi ei pysty yksinään antamaan tarpeeksi tarkkaa tietoa laitteen orientaatiosta. Käyttämällä yhteistyössä magnetometriä ja kiihtyvyysanturia voidaan selvittää puhelimen tarkka orientaatio myös magneettiseen pohjoissuuntaan nähden. Useimmista älypuhelimista ja tableteista löytyy nykyään kolmiakselinen magnetometri. (Understanding Smart Phone Sensor Performance: Magnetometer 2011.)

Kiihtyvyysanturin ohella useissa mobliilisovelluksissa käytetään myös gyroskooppia. Gyroskoopin avulla tunnistetaan kiertoliikkeitä ja pystytään kiihtyvyysanturin tavoin selvittämään laitteen orientaatio. Gyroskoopin toiminta perustuu pyörimisliikkeen säilymiseen eli pyörivän kappaleen liiketilan muutoksia vastustavaan voimaan. (Hyrrävoimat ja gyroskooppi [Viitattu: 21.05.2013].)

Useimmissa Augmented Reality -sovelluksissa gyroskooppia käytetään lähinnä yhteistyössä kiihtyvyysanturin kanssa parantamassa liikkeentunnistuksen tarkkuutta. Gyroskooppi on yksi modernien älypuhelimien vakiovarusteista.

2.4.3 Käyttäjän paikantaminen

Lähes kaikki Augmented Reality -sovellukset vaativat tiedon käyttäjän sijainnista toimiakseen. Kiihtyvyysanturi, gyroskooppi tai magnetometri eivät yksinään kerro

mitään käyttäjän sijainnista. Paikannukseen käytetään yleisesti GPS (Global Positioning System) -satelliittipaikannusjärjestelmää. GPS-signaali voi kertoa käyttäjän sijainnin parhaimmillaan noin 5 - 10 metrin tarkkuudella. (Miettinen 2006, 10.)

GPS-järjestelmässä on kuitenkin omat heikkoutensa. Monet laitteet esimerkiksi matkapuhelin ei käytännössä voi jatkuvasti pitää GPS-paikannusta päällä virrankulutuksellisista syistä. Paikannuksen käynnistämisestä tarkan GPS-signaalin vastaanottamiseen saattaa kulua aikaa jopa useita minuutteja. GPS-signaali kulkee huonosti kiinteiden rakenteiden lävitse mistä johtuu sen huono toimivuus sisätiloissa. Myös säällä, vastaanottimen sijainnilla ja satelliittien kuormituksella on vaikutuksensa GPS-laitteiden toimivuuteen. (Miettinen 2006, 37.)

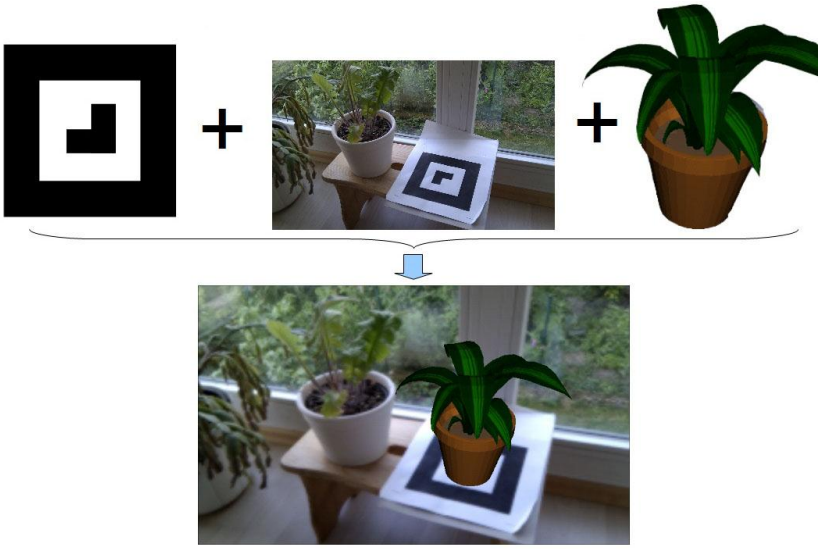
Matkapuhemissa kuuluvuuteen liittyvät ongelmat on ratkaistu jo yleisesti käytössä olevalla A-GPS (Avustettu GPS) -paikannustekniikalla. Se käyttää tavallisen GPS-järjestelmän lisäksi matkapuhelinverkkoja ja palvelimia (Location Server) apuna käyttäjän paikantamisessa. Palvelimien avulla yhdistetään GSM-matkapuhelinverkko ja GPS-järjestelmä toisiinsa. Palvelimet tarjoavat tavanomaisia GPS-laitteita tehokkaamman vastaanottimen ja nopeuttavat huomattavasti ensimmäisen signaalin vastaanottamiseen kuluvaa aikaa (TTFF Time to first fix). Laitteen sijainti on näin mahdollista selvittää sisätiloissakin, missä tavallinen GPS-vastaanotin ei enää toimisi. (GPS vs. aGPS: A Quick Tutorial 2009.)

2.4.4 Konenäkö ja markkerit

Reaaliaikaisesta kamerakuvasta on mahdollista tunnistaa muotoja ja hahmoja kuvantunnistamiseen käytettävillä algoritmeilla. Useimpien Augmented Reality sovellusten käyttämä konenäkö voidaan jakaa piirre- ja mallikohtaiseen hahmojen tunnistamiseen. Piirteitä tunnistavat algoritmit etsivät yhteläisyyksiä 2D kuvasta tunnistettavien piirteiden ja kamerakuvan väliltä. Mallikohtaisessa tunnistuksessa etsitään kamerakuvasta esimerkiksi kolmiulotteisen tai kaksiulotteisen CAD mallin geometriaa vastaavaa muotoja. (Carmigniani & Furth 2011, 6.)

Viime vuosien kehityksestä huolimatta sovellusten käyttämä konenäkö on vielä toiminnaltaan melko rajoittunutta. Augmented Reality sovellusten käyttämä konenäkö on varsin usein riippuvainen seurantaan käytettävästä laitteistosta. Monimutkaisten ja epäsäännöllisten muotojen tunnistaminen on vielä lähes mahdotonta, ja esimerkiksi valaistuksella ja kamerakulmalla on merkittävä vaikutus konenäön toimintaan. Uusin lähestymistapa ongelmiin onkin pyrkimys mallintaa ihmisen näköaistin toimintaa tietokoneelle, jonka yleisesti uskotaan olevan ratkaisu tämän hetken suurimpiin haasteisiin. (Carmigniani & Furth 2011, 7-9.)

Monen nykyaikaisen konenäköä käyttävän Augmented Reality -sovelluksen toiminta perustuu niin sanottujen markkereiden käyttöön. Konenäköä käyttävät sovellukset voidaan vielä jakaa kahteen eri ryhmään: markkereita käyttäviin ja markkereita käyttämättömiin sovelluksiin. Ulkoiset markkerit ovat yksinkertaisia kuviota, joiden geometria on valmiiksi ladattuna ohjelman muistiin. Tavanomaisesti markkerit ovat esimerkiksi paperille tulostettuja geometrisesti yksinkertaisia muotoja. Ohjelman on tarkoitus etsiä markkerit konenäön avulla kamerakuvasta ja käyttää niitä rakennusalustana tietokoneella lisätyille objekteille. Markerin tunnistamalla voidaan myös päätellä käyttäjän katselukulma. Toinen vaihtoehto on etsiä tunnistettavia luonnonmukaisia muotoja kamerakuvasta ja käyttää niitä alustana virtuaalisille objekteille. (Kan, Teng & Chen 2011, 340.)



Kuvio 9. Esimerkki mallinnuksesta käyttämällä ulkoista markkeria.
(AndAR - Android Augmented Reality [Viitattu 12.05.2013].)

3 Android ohjelmointialustana

3.1 Android-käyttöjärjestelmä

Tällä hetkellä maailman eniten käytetty mobiilikäyttöjärjestelmä Googlen Android julkaistiin virallisesti vuonna 2007. Muutama vuosi aiemmin Google osti yhdysvaltalaisten Andy Rubinin, Rich Minerin, Nick Searsin ja Chris Whiten perustaman Android Inc. -yhtymän tarkoituksenaan ainakin huhujen mukaan levittäytyä myös mobiilimarkkinoille. Androidin kehitys- ja ylläpitotyöstä vastaa Googlen johtama Open Handset Alliance, joka on usean mobiilialan yritysten yhteisprojekti. Open Handset Alliancen tärkeimpiä tavoitteita on kehittää avoimia standardeja mobiililaitteille. (Yadav 2011.)

Android-käyttöjärjestelmä on jo käytössä matkapuhelimien lisäksi esimerkiksi tableteissa ja digitaalisissa kameroissa. Lisäksi Androidia kehitetään epävirallisesti avoimen lähdekoodin projektina (Android-x86) etenkin kannettavien tietokoneiden ja minikannettavien käyttöjärjestelmäksi. (Android [Viitattu 21.05.2013].)

3.2 Android-sovellusten kehittäminen

Android on Linux-pohjainen käyttöjärjestelmä, jonka sovellukset pyörivät Dalvik-virtuaalikoneen päällä. Android ei kuitenkaan tue kaikkia Linux-käyttöjärjestelmän standardeja, mikä vaikeuttaa Linux-pohjaisten ohjelmien ja kirjastojen siirtämistä Androidille. Avoimeen lähdekoodiin perustuvalle käyttöjärjestelmälle kehittäminen on täysin ilmaista ja tarvittavat työkalut ovat kaikkien saatavilla. Dokumentointi ja materiaali on kattavaa sekä sovellusten julkaiseminen Google Play -palveluiden kautta on ilmaista. (Mikä on Android [Viitattu: 21.05.2013].)

Sovellukset kehitetään useimmiten Java-kielellä, vaikka itse käyttöjärjestelmä virtuaalikoneineen pohjautuukin C–kielelle. Kyseessä ei kuitenkaan ole perinteinen Java kehitysympäristö vaan Androidille kehitetty versio, joka sisältää omaa toiminnallisuuttaan ja Googlen kehittämiä kirjastoja. Käyttöliittymän kehitys tapahtuu XML–kielellä erillään ohjelman muusta toiminnallisuudesta. Lisäksi

sovellusten rakenne ja käyttöluvat on määriteltävä erillisessä Android manifest -tiedostossa. Valmis ohjelmakoodi ja tarvittavat lähdetiedostot käännetään lopulta kehitysympäristön avulla käyttöjärjestelmän ymmärtämään muotoon. Myös C- ja C++-kielet ovat tuettuna NDK:n (Native Development Kit) tarjoamien rajapintojen kautta. (Mikä on Android [Viitattu: 21.05.2013].)

3.2.1 Kehitystyökalut

Virallinen Googlen suosittelema ohjelmisto kehittäjille on ilmainen Eclipse-kehitysympäristö ja siihen kuuluva Android Developer Tools (ADT) -lisäosa. Lisäksi on hankittava vielä Android-sovelluskehitystyökalupaketti (SDK), joka on saatavilla useimmille käyttöjärjestelmille. SDK sisältää tarvittavien kirjastojen lisäksi emulaattorin, jonka avulla sovellusten testaus onnistuu ilman Android-laitetta. SDK:n päivitykset ja edeltävät versiot ovat vapaasti ladattavissa kehittäjien sivustolta tai Eclipse-ohjelmiston kautta. Vaihtoehtoisesti Eclipse ohjelmisto tukee myös ohjelmien kehittämistä natiivina. Tämä onnistuu käyttämällä NDK kehityspakettia, joka on saatavilla mm. Android-kehittäjien sivustolta. Google tarjoaa lisäksi kehittäjille kattavan materiaalin ohjeistuksineen sekä useita ilmaisia lisäpalveluja. Myös NetBeans-ohjelmistolle on saatavilla muutamia kehitystyökaluja, mutta ne eivät ole virallisesti Googlen tukemia. (Mikä on Android [Viitattu: 21.05.2013].)

3.3 Sovellusten rakenne ja komponentit

Androidissa käytettävä Linux-käyttöjärjestelmä käsittelee jokaisen sovelluksen omana käyttäjänään. Käyttöjärjestelmä tunnistaa ohjelmat toisistaan sen jakamien käyttäjätunnusten avulla. Jokaisen sovelluksen ohjelmakoodi pyörii eristyksessä muista ohjelmista omassa virtuaalikoneessaan. Vaihtoehtoisesti sovellukset voivat myös jakaa saman Linux-käyttäjätunnuksen, mikä mahdollistaa tiedostojen jakamisen sovellusten kesken. Myös virtuaalikoneen jakaminen on näin mahdollista, mikä säästää huomattavasti käyttöjärjestelmän resursseja. Sovellukset voivat myös pyytää asennusvaiheessa käyttäjältä käyttölupaa laitteen

omia tiedostoja varten esim. yhteystiedot, kamera ja muistikortin sisältö. (Application Fundamentals [Viitattu 12.05.2013].)

Android-sovellukset koostuvat yhdestä tai useammasta komponentista. Jokaisella neljällä komponentilla, aktiviteeteilla, palveluilla, sisällön tuottajilla ja yleisvastaanottimilla on oma roolinsa ohjelman suorituksen aikana. Kaikilla komponenteilla on myös oma elinkaarensa, joka määrittää tarvittavat toiminnot komponentin luomishetkestä tuhoutumiseen. Monista muista käyttöjärjestelmistä poiketen Android komponentteille on määritelty yhden main()-funktion sijasta useita entry pointteja. Tämä mahdollistaa toisille sovelluksille kuuluvien komponenttien käyttämisen osana omaa sovellusta. (Application Fundamentals [Viitattu 12.05.2013].)

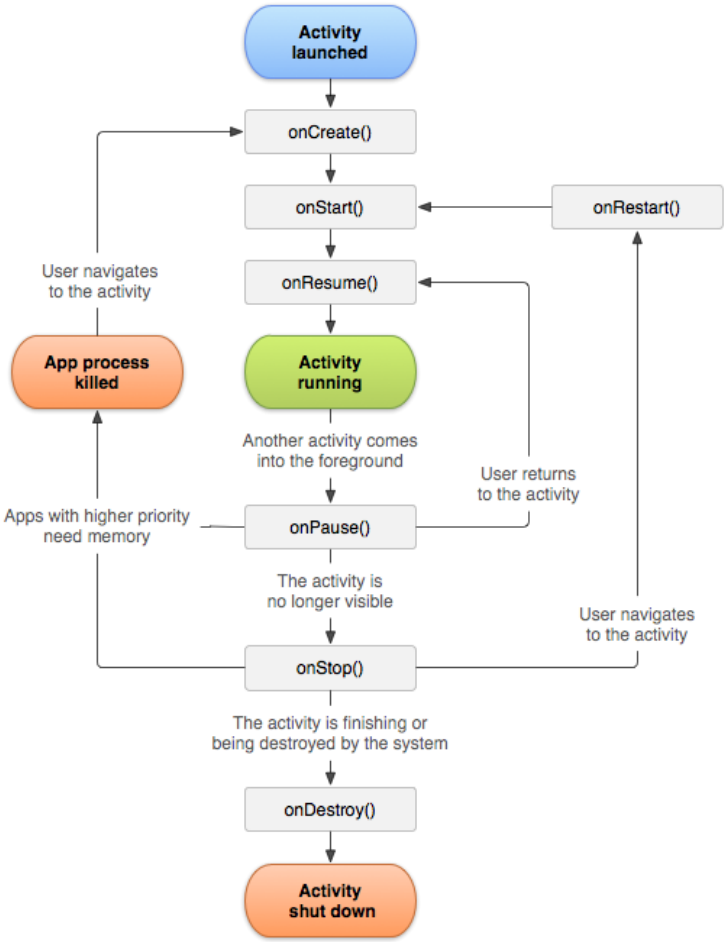
Komponenttien käynnistäminen ei kuitenkaan onnistu ilman käyttöjärjestelmän suostumusta. Ainoastaan käyttöjärjestelmällä on oikeudet käynnistää komponentit ja se vaatiikin sovellukselta tiedon (intent), miksi kyseinen komponentti pitäisi käynnistää. Intent on yksinkertainen asynkroninen viesti, jonka avulla voidaan välittää tietoa järjestelmän ja komponenttien välillä. (Application Fundamentals [Viitattu 12.05.2013].)

3.3.1 Aktiviteetit

Aktiviteetti (activity) on yksi tärkeimmistä käsitteistä Android-ohjelmoinnissa. Sillä tarkoitetaan sovelluksen yksittäistä sivua tai näkymää esim. karttanäkymä on aktiviteettinsa ja kohteiden etsintä tapahtuu omassa aktiviteetissaan. Aktiviteetit toimivat yhdessä muodastaen sovelluksen käyttöliittymän, mutta niiden toiminnallisuus on eritelty toisistaan. Tyypillisesti yksi aktiviteetti määritellään main activityksi, joka avataan aina sovelluksen käynnistyessä. Uuden aktiviteetin käynnistäminen pysäyttää aina edellisen, koska vain yksi aktiviteetti voi kerrallaan olla aktiivinen. Sovellus voi käynnistää myös toiselle sovellukselle kuuluvia aktiviteetteja, mikäli toinen ohjelma antaa siihen tarvittavan luvan. Tämä on yksi Android-järjestelmän tärkeimmistä ominaisuuksista, sillä se antaa sovelluksen kehittäjälle mahdollisuuden käyttää valmiita sovelluksia esim. kameraa osana omaa ohjelmistoaan. Käyttöjärjestelmä säilyttää pysäytettyjä aktiviteetteja

muistissa LIFO(last-in-first-out) -rakenteisessa pinossa. Laitteen paluu-nappi palauttaa aina pinosta uusimman activityn ja tuhoaa edellisen. (Activities [Viitattu 12.05.2013].)

Activity-luokka sisältää sisältää useita eri kutsumetodeja, joita järjestelmä kutsuu automaattisesti sovelluksen elinkaaren aikana. Jokainen luokan tila esim. käynnistyminen, pysäyttäminen ja tuhoaminen, on määritelty omaksi kutsukseen. Tämä mahdollistaa kullekkin tilalle ominaisten toimintojen toteutuuttamisen omissa lohkoissaan. (Activities [Viitattu 12.05.2013].)



Kuvio 10. Aktiviteettien elinkaari.
(Activities: Implementing the lifecycle callbacks [Viitattu 12.05.2013].)

3.3.2 Palvelut

Palveluja (service) käytetään suorittamaan pitkäkestoisia operaatioita käyttöliittymän taustalla omassa säikeessään. Muut ohjelmakomponentit voivat

toimia yhteistyössä palvelujen kanssa mahdollistaen tiedonsiirron komponenttien välillä (Interprocess communication). Palvelun käynnistäminen onnistuu myös toisen ohjelmakomponentin toimesta ja se voi jäädä suoritustilaan sovelluksen sammuttamisenkin jälkeen. Palvelu voi olla myös sidottuna toiseen komponenttiin, joka mahdollistaa esim. palvelun sammuttamisen komponentin tuhouduttua. Tyypillisesti palveluita käytetään esim. hoitamaan internet-liikennettä, soittamaan musiikkia ja tiedostojen lukemiseen. (Application Fundamentals [Viitattu 12.05.2013].)

3.3.3 Sisällön tuottajat

Android-järjestelmä käyttää sisällön tuottajien (Content provider) rajapintoja hallinnoimaan tiedonvälitystä prosessien välillä. Jaetut tiedostot voivat olla tallennettuna esim. laitteen omaan tiedostojärjestelmään tai yhteiseen tietokantaan. Rajapinnan salliessa sovellukset pääsevät käsiksi ja jopa muokkaamaan yhteiseksi määriteltyjä tiedostoja. Hyvä esimerkki sisällön tuottajasta on käyttäjän yhteistiedot. Yhteystietoihin päästään käsiksi useasta eri sovelluksesta ja niitä voidaan tarpeen tullen myös muokata. (Application Fundamentals [Viitattu 12.05.2013].)

3.3.4 Yleisvastaanottimet

Yleisvastaanotin (Broadcast receiver) on komponentti, jonka avulla hoidetaan sovellusten ja käyttöjärjestelmän tai pelkästään sovellusten välistä viestintää. Viestintä on tyypillisesti käyttöjärjestelmän ilmoituksia laitteiston tilasta tai sovellusten välistä kommunikointia tiedostojen saatavuudesta. Broadcast receiver on lähinnä tarkoitettu helpoksi tavaksi välittää minimaalista liikennettä komponenttien välillä. Yleisvastaanotin komponentilla ei palvelujen tapaan ole käyttöliittymää, mutta sen avulla voidaan informoida käyttäjää Status bar -valikossa. (Application Fundamentals [Viitattu 12.05.2013].)

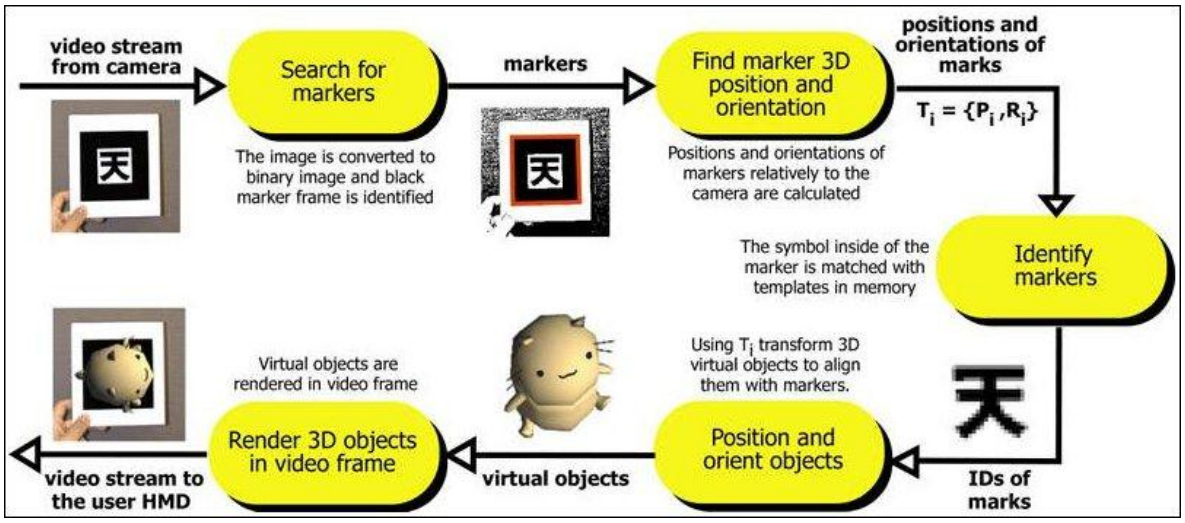
3.4 ARToolKit ja ARToolKitPlus

Augmented reality -sovellusten kehittämistä varten on kehitetty useita ulkoisia kirjastoja. Ensimmäinen avoimen lähdekoodin Augmented Reality -kirjasto on tohtori Hirokazun kehittämä ARToolKit julkaistiin jo vuonna 1999. ARToolKitin kehitystyö jatkuu yhä Human Interface Technology Laboratoryn (HIT Lab) tukemana. ARToolKit on yhä laajasti käytetty ja toimii pohjana muille kirjastoille. ARToolKit ja monet sen johdannaisista ohjelmoidaan C- tai C++-kiellillä. Tästä syystä ne soveltuvat huonosti lähinnä Java-kielellä ohjelmoitavaksi tarkoitetulle Android-ohjelmointialustalle. (Introduction to ARToolKit [Viitattu 12.05.2013].)

ARToolkit tarjoaa nopean ja helposti lähestyttävän rajapinnan omien Augmented Reality -sovellusten kehittämisen. Kirjaston avulla konenäkö ja muut vaikeasti toteutettavat toiminnot ovat helposti kehittäjän ulottuvissa. Kirjasto tukee videonäyttöjen lisäksi optisia-näyttöjä ja toimii lähes kaikissa käyttöjärjestelmissä ongelmitta. ARToolKit-pohjaisten sovellusten toiminta perustuu ulkoisiin markkereihin, joita käytetään rakennusalustoina virtuaalisille objekteille. Markkerit voivat olla joko itse tehtyjä tai kirjaston mukana tulleita yksinkertaisia paperille tulostettavia muotoja. Niiden avulla selvitetään myös käyttäjän katselukulma ja kameran etäisyys markkeriin nähden. (Introduction to ARToolKit [Viitattu 12.05.2013].)

Konenäköön perustavalle Augmented Reality –järjestelmälle tyypillisesti ARToolKit sisältää toimintarajoituksia. Markkereiden koolla, valaistuksella ja kamerakulmalla on vaikutuksensa ohjelman toimintaan. Vähän yksityiskohtia sisältävät ja asymmetritset muodot toimivat usein parhaiten. (How does ARToolKit work? [Viitattu 12.05.2013].)

Sovellukset toimivat etsimällä kamerakuvasta neliömäisiä muotoja. Ohjelman tunnistaessa muodon lasketaan kameran etäisyys löydettyyn neliöön nähden. Kameran sijainnin selviinnyttyä piirretään tietokonegrafiikat suhteessa käyttäjän sijaintiin. Tietokonegrafiikat piirretään reaaliaikaisesti markkerin osoittamaan paikkaan kamerakuvan päälle. Laitteen asennon muuttuessa laskenta suoritetaan uudelleen luoden illuusion lisätystä todellisuudesta. (How does ARToolKit work? [Viitattu 12.05.2013].)



Kuvio 11. ARToolKit sovellusten toimintaperiaate.
(How does ARToolKit work? [Viitattu 12.05.2013].)

Sovellukset ovat vielä toiminnaltaan melko rajoitettuja. Konenäön toiminta on täysin riippuvainen ympäristöön lisätyistä markkereista, jotka rajoittavat virtuaalisten objektien kokoa ja liikkuvuutta. Markkerit mahdollistavat myös kameran etäisyyden selvittämisen ja niiden näkyvyys on avainasemassa ohjelman toimivuuden kannalta. Heijastuksilla, valaistuksella, markkerien koolla ja laitteen orientaatiolla on myös vaikutuksensa konenäön toimintaan. (How does ARToolKit work? [Viitattu 12.05.2013].)

Pian ARToolKitin jälkeen kehitetty ARToolKitPlus lisäsi edeltäjäänsä uusia ominaisuuksia ja luokkapohjaisia rajapintoja. Suorituskykyä ja konenäköä on myös parannettu edeltäjäänsä nähden. ARToolKitPlus-kirjaston kehitystyön loputtua vuonna 2006 seurasi sen jalanjäljissä Studierstube Tracker. Yhtäläisyyksistään huolimatta Studierstube Tracker on kehitetty alusta pitäen omana projektinaan, eikä se ole edeltäjiensä tavoin avoimen lähdekoodin projekti. (Studierstube Tracker 2011.)

3.4.1 Java-kieltä tukevat kirjastot

Toistaiseksi Java-kieltä tukevat vain harvat ilmaiset kirjastot, mikä on valitettavaa Android-kehittäjien kannalta. ARToolKit on saanut useita menestyksekkäitä seuraajia, jotka tarjoavat parannuksia edeltäjänsä toimintoihin. Kirjastot ovat edeltäjänsä tapaan avoimen lähdekoodin projekteja, mutta eivät ole yhtä tarkkaan

dokumentoituja. Yksi tunnetuimmista myös Java-kielellä ohjelmoitavissa olevista kirjastoista on NyARToolkit. NyARToolkit on japanilaisen Nyatla-nimisen miehen kehittämä ARToolKit-kirjastoon perustuva projekti, joka julkaistiin vuonna 2008. Natiivisti Java-kielelle kehitetty kirjasto on saatavilla myös muille ohjelmointikielille esim. C# ja C++. Etuna edeltäjäänsä NyARToolkit tarjoaa monipuolisemman markkereiden tunnistamisjärjestelmän ja paremman suorituskyvyn sovelluksille. Grafiikoiden piirtämiseen käytetään edeltäjänsä tapaan OpenGL-rajapintaa. (NyARToolkit project [Viitattu 12.05.2013].)

Toinen ilmainen erityisesti Androidille suunnattu kirjasto on AndAr (Android Augmented Reality). AndAr on myös ARToolKit-kirjastoon pohjautuva avoimen lähdekoodin projekti, joka sisältää suunnilleen samat toiminnot kuin edeltäjänsäkin. AndAr tarjoaa Android kehittäjille helposti lähestyttävän alustan, joka on kehitettävissä Java-kielellä. (HowToBuildApplicationsBasedOnAndAR 2010.)

4 Toteutettu sovellus

Projekti toteutettiin 2012 kesän aikana. Sovellus on suunniteltu Android-käyttöjärjestelmälle ja tulee olemaan osa toimeksiantajan ohjelmistoa. Ohjelma opastaa käyttäjän perille etsityille kohteille näytölle piirtyvien ikonien ja tutkanäkymän avulla.

4.1 Käytetyt kehitystyökalut ja testilaitteisto

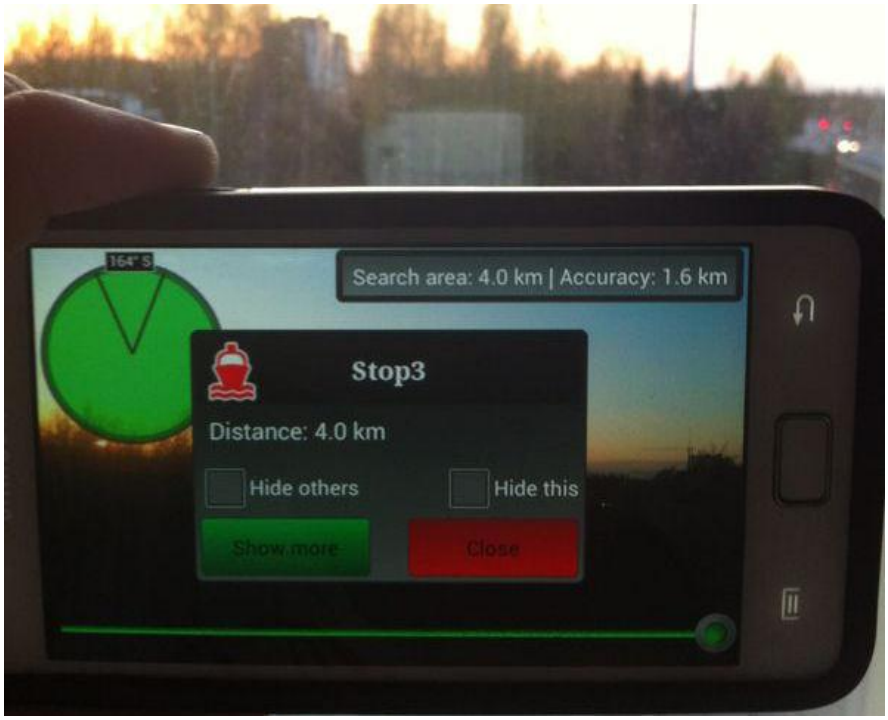
Sovelluksen ohjelmointialustaksi valittiin toimeksiantajan ohjelmiston tavoin Android käyttöjärjestelmä. Valintaan vaikutti myös Androidin laaja dokumentointi ja saatavilla olevan materiaalin määrä. Ohjelmointikielenä toimi Androidin virallinen kehityskieli eli Java. Ohjelmointi suoritettiin käyttämällä ilmaista Eclipse-ohjelmistoa ja siihen ladattavissa olevaa Android-kehitystyökalupakettia.

Augmented Reality -sovellukset ovat usein raskaita ja vaativat paljon älypuhelimen suorittimelta. Sama koskee myös toteutettua sovellusta. Tästä johtuen testi- ja kehityskäytössä toimi Samsung Galaxy SII -älypuhelin, joka oli aikansa tehokkain malli. Ensimmäiset testaukset suoritettiin kehitystyökalupaketin mukana tulevalla Android-emulaattorilla.

4.2 Käyttöliittymä

Yksinkertainen käyttöliittymä sisältää reaaliaikaisen kamerakuvan, kompassinäkymän ja etsintäalueen säätimen. Kompassinäkymässä näkyy etsintäalueella sijaitsevat pysäkit, näköalue ja kompassilukema. Puhelimen alareunaan on sijoitettu etsintäalueen säädin, jonka säde on rajoitettu kahdeksaan kilometriin. Oikeassa yläreunassa näytetään GPS-signaalin tarkkuus ja valittu etsintäalueen säde. Jokainen näköalueella oleva kohde esitetään etäisyyksineen omana ikoninaan reaaliaikaisessa kamerakuvassa. Ikonia koskettamalla saa auki valikon, joka sisältää lisätietoa kohteesta.

Kuvio 12. Sovelluksen käyttöliittymä



Kuvio 13. Näkymä valitusta kohteesta

4.3 Android Augmented Reality Framework

Sovellus käyttää Justin Wetherellin kehittämää runkokoodia, joka on suunniteltu pohjaksi Augmented Reality -sovelluksille. Projekti on julkaistu Apache License 2.0

lisenssin alaisena ja löytyy dokumentteineen http://code.google.com/p/android-augment-reality-framework/ sivustolta. Runkokoodi sisältää myös alunperin Mixarelle kehitettyjä luokkia. Mixare on avoimen lähdekoodin Augmented Reality projekti, joka käyttää General Public License (GPLv3) lisenssiä (Mixare – Open Source Augmented Reality Engine [Viitattu 21.05.2013]).

4.4 Sovelluksen toimintaperiaate

Ohjelman toiminta perustuu käyttäjän paikantamiseen ja puhelimen asennon tunnistamiseen. Etsittyjen kohteiden koordinatit tullaan hakemaan tietokannasta ja niiden etäisyydet lasketaan aina käyttäjän sijainnin muuttuessa. Sijaintitietojen, puhelimen orientaation ja magneettikentän avulla päätellään kohteen sijainti puhelimennäytöllä. Useimmat tietokannat eivät sisällä tietoa kohteiden korkeudesta, mistä johtuen kohteiden korkeus asetetaan aina samaksi kuin laitteen korkeus. Ohjelma käyttää A-GPS-tekniikkaa käyttäjän paikantamiseen ja laskee sen avulla käyttäjän etäisyyden valittuihin kohteisiin. GPS-signaalin tarkkuudella on merkittävä vaikutus sovelluksen toimintaan, mistä johtuen sovellusta ei suositella käytettäväksi sisätiloissa. Erona moneen Augmented Reality –sovellukseen on, että työssä ei tarvitse käyttää konenäköä.

Puhelimen asennon tunnistamiseen käytetään puhelimen omaa kiihtyvyysanturia ja magnetometriä. Molemmat sensorit ovat melko herkkiä häiriöille, mistä johtuen mittauslukemat kulkevat ohjelmoidun alipäästösuodattimen kautta. Pelkkä kiihtyvyysanturi ei tässäkään tapauksessa yksinään riitä kertomaan puhelimen tarkkaa orientaatiota, vaan se tarvitsee jonkin vertailukohteen. Sovelluksessa vertailukohteen tarjoaa kompassin lailla toimiva magnetometri. Magnetometrin tietojen lisäksi on selvitettävä eranto (deklinaatio), eli kuinka monta astetta kompassin osoittama magneettinen pohjoissuunta eroaa maantieteellisestä pohjoissuunnasta (Miettinen 2006, 100). Käyttäjän sijainnin päivittyessä eranto selvitetään Android-ohjelmointialustan tarjoaman GeomagneticField-luokan avulla. Sensoritietojen ja erannon avulla lasketaan kääntömatriisi, joka toimii ohjelman toiminnan perustana. Kääntömatriisista saadaan vektorilaskennan avulla selvitettyä todellisessa maailmassa sijaitsevan pisteen sijainti laitteen näytöllä.

Käyttäjän sijainnin perusteella näytetään hakualueen säteellä sijaitsevat kohteet käyttöliittymän kompassinäkymässä. Kameranäkymässä näkyy ainoastaan kompassinäkymässä määritetyn näköalueen sisällä olevat kohteet. Lisäksi etäisyyksiä kohteisiin havainnollistetaan käyttäjälle ikonien muuttuvalla koolla.

4.5 Sensoreiden käyttöönotto

Yksi sovelluksen tärkeimmistä toiminnoista on käyttäjän seurantaan tarvittavien sensorien käyttöönotto. Android ohjelmointialusta sisältää sensoreiden käyttöä varten tarvittavat rajapinnat ja niiden käyttäminen on tehty melko yksinkertaiseksi. Yleinen ja tässäkin sovelluksessa käytetty tapa on luoda uusi luokka, joka perii Activity-yliluokan ja käyttää ohjelmointialustan mukana tulevaa SensorEventListener-rajapintaa.

Ohjelman aktivoituessa Android käyttöjärjestelmä kutsuu Activity-pääluokan onStart()-metodia, jossa otetaan käyttöön eli rekisteröidään sovelluksen käyttämät sensorit. Lista saatavilla olevista tai halutuista sensoreista on saatavilla SensorManager-olion kautta. Mikäli etsitty sensori löytyy haetusta listasta, tallennetaan se Sensor-tyyppiseen luokkamuuttujaan. Lopulta löydetyt sensorit rekisteröidään SensorManager-olion registerListener()-metodia kutsumalla.

```
public void registerSensors()
{
     try {

         if (sensorMgr == null)
         sensorMgr = (SensorManager) getSystemService(SENSOR_SERVICE);

         sensors = sensorMgr.getSensorList(Sensor.TYPE_ACCELEROMETER);
         if (sensors.size() > 0) sensorGrav = sensors.get(0);

         sensors = sensorMgr.getSensorList(Sensor.TYPE_MAGNETIC_FIELD);
         if (sensors.size() > 0) sensorMag = sensors.get(0);

         sensorMgr.registerListener(this, sensorGrav, SensorManager.SENSOR_DELAY_GAME);
         sensorMgr.registerListener(this, sensorMag, SensorManager.SENSOR_DELAY_NORMAL);

    } catch (Exception e) {
        unregisterSensors();
    }
```

Kuvio 14. Sensoreiden rekisteröinti

Metodi tarvitsee parametreikseen SensorEventListener- ja Sensor-luokkien ilmentymät sekä Int-tyyppisen muuttujan määrittämään sensorissa käytettävän viiveen. Toteutetun sovelluksen tapauksessa SensorEventListener-rajapinta on jo

luokan käytössä, joten parametriksi voidaan laittaa pelkästään this. Sensoreiden viiveet on määritelty SensorManager-luokan vakioiksi.

Muutokset sensorien tilassa kutsuvat automaattisesti SensorEventListener-rajapinnan onSensorChanged()-metodia. Metodi sisältää parametrina SensorEvent-luokan ilmentymän. Olio sisältää tarvittavat tiedot kutsun aiheuttaneeen sensorin tilasta. Tarvittavat mittauslukemat on kerätty float-tyyppiseksi taulukoksi values-nimiseen muuttujaan. Kiihtyvyysanturin ja magnetometrin tapauksessa lukemat ovat kolmialkioisessa taulukossa, joka vastaa mittausarvoja kolmiulotteisessa koordinaatistossa. Lopuksi lukemat vielä suodatetaan LowPassFilter-luokan avulla mahdollisten häiriöiden poistamiseksi ja tallennetaan ohjelman muistiin.

```
@Override
public void onSensorChanged(SensorEvent evt)
{

            if (evt.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {

                if(evt.values == null) return;

                smooth = LowPassFilter.filter(GRAV_FILTER, evt.values, grav);
                grav[0] = smooth[0];
                grav[1] = smooth[1];
                grav[2] = smooth[2];

            } else if (evt.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {

                if(evt.values == null) return;

                smooth = LowPassFilter.filter(MAG_FILTER, evt.values, mag);
                mag[0] = smooth[0];
                mag[1] = smooth[1];
                mag[2] = smooth[2];
            }
}
```

Kuvio 15. Sensoritietojen lukeminen

4.6 GPS–järjestelmän käyttö

Android-ohjelmointialustassa on yleisesti käytössä Location-luokka, jota käytetään sijaintitietojen ja niihin liittyvien toimintojen yhteydessä. Android alusta tarjoaa myös valmiin LocationListener-rajapinnan sekä LocationManager-luokan GPS-järjestelmän käyttöä varten.

Sovelluksessa LocationListener- ja SensorEventListener-rajapinnat ovat käytössä yhteisessä luokassa, jonka avulla tarkkaillaan puhelimen sensoreita ja vastaanotetaan sijaintipäivitykset. Uusia päivityksiä verrataan aina viimeisimpään hyväksyttyyn sijaintiin hyläten vanhat ja epätarkat päivitykset. Ainoastaan selvästi tarkemmat tai uudemmat päivitykset hyväksytään.

Locationmanager-luokasta on luotu uusi ilmentymä, joka alustetaan sensoreiden tapaan kutsumalla Activity-pääluokan getSystemService()-metodia. Oikea palvelu haetaan jälleen Context-luokan vakiomuuttujista. LocationManager-olion kautta kutsutaan edelleen requestLocationUpdates()-metodia, jonka avulla GPS- ja A-GPS-paikannusjärjestelmät saadaan sovelluksen käyttöön. Metodiin määritetään parametreiksi: palvelun tarjoaja, pienin aikaväli päivitysten välillä (millisekunteina), pienin etäisyys sijaintipäivitysten välillä (metreinä) sekä LocationListener-rajapintaa käyttävä luokka. Sijaintipalvelujen tarjoajat löytyvät LocationManager-luokan vakioista.

```
locationMgr = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
locationMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER, MIN_TIME,
                                   MIN_DISTANCE, SensorsActivity.this);
locationMgr.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, MIN_TIME,
                                                    50, SensorsActivity.this);
```

Kuvio 16. Paikannuspalvelun käyttöönotto

Ensimmäisten päivitysten ilmaantumiseen saattaa kulua jokin aikaa, mistä johtuen ohjelma yrittää aluksi hakea käyttäjän sijaintia puhelimen muistista. Tämä onnistuu kutsumalla LocationManager olion getLastKnownLocation() metodia.

```
Location gps=locationMgr.getLastKnownLocation(LocationManager.GPS_PROVIDER);
Location network=locationMgr.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
```

Kuvio 17. Sijaintitietojen hakeminen puhelimen muistista

LocationListener rajapinnan keskeisessä asemassa on onLocationChanged()-metodi, jota kutsutaan aina signaalin vastaanoton yhteydessä. Metodi sisältää Location-luokan ilmentymän parametrina. Olio voi koordinaattien lisäksi sisältää tietoa esimerkiksi GPS–signaalin tarkkuudesta ja palvelun tarjoajasta. Metodin sisällä arvioidaan yksinkertaisen apufunktion avulla onko uusi päivitys tarkempi kuin nykyinen sijainti.

```
@Override
public void onLocationChanged(Location location) {

    if(ARData.getCurrentLocation() != null)
    {
        if(isBetterLocation(location, ARData.getCurrentLocation()))
            {
                ARData.setCurrentLocation(location);
                calculateGMF(); //Eranto
            }
    }

}
```

Kuvio 18. Sijaintitietojen vastaanotto

4.7 Kääntömatriisin muodostaminen

Sovelluksen toiminnan perustana on kääntömatriisi, joka muodostetaan kiihtyvyysanturin ja magnetometrin mittauslukemien pohjalta. Sen avulla pystytään siirtämään leveys ja pituusasteina esitettyjen paikkojen sijainnit kolmiulotteiseen koordinaatistoon ja lopulta laitteen näytölle. Kääntömatriisi mahdollistaa myös laitteen orientaation selvittämisen. Matriisi- ja vektorilaskennoissa käytetään apuna alunperin Mixarelle kehitettyjä luokkia. Luokat sisältävät kaikki yleisimmät matriisi- ja vektorilaskennassa käytetyt laskutoimitukset.

Koordinaatistona toimiva kääntömatriisi muodostetaan kutsumalla SensorManager-luokan staattista getRotationMatrix() metodia. (Android-Augment-Reality-Framework: How it works 2012.)

```
SensorManager.getRotationMatrix(temp, null, grav, mag);
```

Kuvio 19. Kääntömatriisin muodostaminen kiihtyvyysanturin ja magnetometrin tietojen pohjalta

Sovellusta käyttäessä puhelin on aina vaakatasossa, mistä johtuen koordinaatisto joudutaan vielä kääntämään. Kääntäminen tapahtuu kutsumalla SensorManager-luokan staattista remapCoordinateSystem() metodia. Metodi vaatii ensimmäiseksi parametrikseen alkuperäisen kääntömatriisin arvoja vastaavan taulukon. Kaksi seuraava parametria määrittävät koordinaatiston uudet X- ja Y-akselit. Koordinaatisto saadaan käännettyä (90 astetta vasemmalle) määrittelemällä akseleita vastaavien parametrien arvoiksi SensorManager-luokan AXIS_Y- ja

AXIS_MINUS_X-vakiomuuttujat. Uudelleen lasketut arvot sijoitetaan viimeisenä parametrina olevaan muuttujaan. Lopuksi uudelleen lasketut arvot tallennetaan uuteen matriisiin (Kuvio 20, muuttuja worldCoord), jota tarvitaan myöhemmin apuna lopullisen kääntömatriisin muodostamisessa. (Android-Augment-Reality-Framework: How it works 2012.)

```
SensorManager.remapCoordinateSystem(temp, SensorManager.AXIS_Y,
                                    SensorManager.AXIS_MINUS_X, rotation);

worldCoord.set(rotation[0], rotation[1], rotation[2], rotation[3], rotation[4],
                            rotation[5], rotation[6], rotation[7], rotation[8]);
```

Kuvio 20. Koordinaatiston kääntäminen ja matriisin luominen

Pelkästään magnetometrin ja kiihtyvyysanturin arvoja käyttävässä kääntömatriisissa ei vielä ole otettu huomioon erantoa. Ohjelma selvittää erannon aina sijaintipäivitysten yhteydessä. Tämä onnistuu luomalla olio GeomagneticField-luokasta, joka alustetaan viimeisimmän hyväksytyn sijainnin koordinaateilla. Magneettiset navat ja kentät muuttuvat hitaasti, mistä johtuen luokan konstruktori vaatii tiedon myös ajanhetkestä. Järjestelmän kellonaika saadaan haettua kutsumalla System-luokan currentTimeMillis()-metodia. Olion luomisen jälkeen eranto selvitetään kutsumalla getDeclination()-metodia. Vastaus täytyy vielä kääntää asteista radiaaneiksi. (GeomagneticField [Viitattu 12.05.2013].)

```
Location location = ARData.getCurrentLocation();
        gmf = new GeomagneticField((float) location.getLatitude(),
        (float) location.getLongitude(),
        (float) location.getAltitude(),
        System.currentTimeMillis());

angleY = Math.toRadians(gmf.getDeclination());
```

Kuvio 21. Erannon selvittäminen käyttämällä GeomagneticField luokkaa

Vastauksen pohjalta muodostetaan seuraavaksi erantoa vastaava matriisi (kuvio 22, muuttuja mageticNorthCompensation). Ennen operaatiota matriisi on kuitenkin muutettava yksikkömatriisiksi kutsumalla toIdentity()-metodia. Matriisi joudutaan vielä lopuksi kääntämään vastaamaan puhelimen asentoa. (Android-Augment-Reality-Framework: How it works 2012.)

```
synchronized (mageticNorthCompensation) {
//Identity matrix
// [ 1, 0, 0 ]
// [ 0, 1, 0 ]
// [ 0, 0, 1 ]
    mageticNorthCompensation.toIdentity();


// [ cos,  0, sin ]
// [ 0,    1, 0   ]
// [ -sin, 0, cos ]
    mageticNorthCompensation.set( (float) Math.cos(angleY),
                        0f,
                        (float) Math.sin(angleY),
                        0f,
                        1f,
                        0f,
                        (float) -Math.sin(angleY),
                        0f,
                        (float) Math.cos(angleY));

//Rotate the matrix to match the orientation
    mageticNorthCompensation.prod(xAxisRotation);
```

Kuvio 22. Erantomatriisin kääntäminen vastaamaan puhelimen asentoa

Lopullinen kääntömatriisi saadaan muodostamalla tulomatriisi aiemmin lasketun kääntömatriisiin ja erantoa vastaavan matriisin pohjalta. Operaatio vaatii vielä kolmannenkin apumatriisin, joka jälleen alustetaan ennen operaatiota yksikkömatriisiksi. Seuraavaksi muodostetaan tulomatriisi apumatriisin (kuvio 23, magneticCompensatedCoord-muuttuja) ja erantomatriisin pohjalta kutsumalla apumatriisin prod()-metodia. Apumatriisille lasketaan vielä toinen tulomatriisi alkuperäisen kääntömatriisin (worldCoord) pohjalta. Uudelleen muodostettu matriisi vielä käännetään ylösalaisin ja oikealta vasemmalle kutsumalla invert()-metodia. (Android-Augment-Reality-Framework: How it works 2012.)

Lopullinen kääntömatriisi on näin muodostettu ja se tallennetaan ohjelman muistiin.

```
//Identity matrix
// [ 1, 0, 0 ]
// [ 0, 1, 0 ]
// [ 0, 0, 1 ]
magneticCompensatedCoord.toIdentity();

synchronized (mageticNorthCompensation) {
    //Cross product the matrix with the magnetic north compensation
    magneticCompensatedCoord.prod(mageticNorthCompensation);

}

//Cross product with the world coordinates
magneticCompensatedCoord.prod(worldCoord);

//Invert the matrix
magneticCompensatedCoord.invert();
```

Kuvio 23. Lopullisen kääntömatriisin muodostus

4.7.1 Laitteen asennon määrittäminen

Android-ohjelmointialustassa käytetään termejä azimuth, pitch ja roll kuvaamaan laitteen asentoa kolmiulotteisessa tilassa. On huomioitavaa, että termit eroavat lentodynamiikan määrittelemästä termistöstä. Azimuth vastaa kääntymistä pituusakselin, roll pystyakselin ja pitch poikkiakselin suuntaan. (Position Sensors [Viitattu 12.05.2013].)

Sovellus käyttää runkokoodin mukana tullutta luokkaa apuna laitteen asennon määrittämisessä. Luokka laskee laitteen orientaation annetun kääntömatriisin arvojen pohjalta. Kääntyminen kunkin akselin suuntaan (azimuth, pitch ja roll) tallennetaan lopuksi ohjelman muistiin. Yleisemmin käytetty tapa on käyttää Android-ohjelmointialustan mukana tulevaa SensorManager-luokan getOrientation()-metodia. Metodi toimii samalla periaattella kuin edellä mainittu luokka, mutta saattaa olla hieman epätarkempi.

4.7.2 Ikonit

Ikonien sijainti näytöllä määritellään puhelimen orientaation, näytön ominaisuuksien ja kääntömatriisin perusteella. Toteutus pohjautuu runkokoodin

käyttämiin luokkiin, joista olennaisimpina mainittakoon luokat: Marker, CameraModel, AugmentedView ja PhysicalLocation. (Android-Augment-Reality-Framework: How it works 2012.)

AugmentedView-luokka perii Android-alustan käyttöliittymien yhteydessä käytettäväksi tarkoitetun View-luokan. Augmented View -luokka sisältää pääluokalta perityn onDraw() metodin, jonka yhteydessä piirrettään kohteiden ikonit puhelimen näytölle. Metodin kutsu tapahtuu automaattisesti aina näytön päivityksen yhteydessä. Kameran ominaisuudet ovat määritelty sovelluksen käynnistyessä CameraModel-luokan ilmentymään. Olio sisältää tiedon katselukulmasta ja näytön piirtoalueesta. Se sisältää myös projectPoint()-metodin, jonka avulla lasketaan kolmiulotteisessa koordinaatistossa sijaitsevan pisteen sijainti laitteen näytöllä. (Android-Augment-Reality-Framework: How it works 2012.)

Etsityt kohteet tallennetaan aina Marker-luokan olioina ohjelman muistiin. Laitteen sijainnin päivittyessä jokainen tallennettu olio päivitetään kutsumalla calcRelativePosition()-funktiota. Funktion tarkoitus on laskea kunkin kohteen etäisyys käyttäjän sijaintiin nähden kaikilla kolmella akselilla. (Android-Augment-Reality-Framework: How it works 2012.)

```
for(Marker ma: markerList.values())
{
    ma.calcRelativePosition(newLocation);
}
```

Kuvio 23. Markkereiden päivitys

Seuraavaksi käyttäjän sijaintia vastaavien koordinaattien pohjalta muodostetaan vektori (Kuvio 23, locationXyzRelativeToPhysicalLocation-muuttuja) käyttämällä PhysicalLocation-luokan convLocationToVector()-funktiota. Funktio tarvitsee parametreikseen kohteen sijainnin, käyttäjän sijainnin sekä ulostulona toimivan vektorimuuttujan. (Android-Augment-Reality-Framework: How it works 2012.)

```
// Compute the relative position vector from user position to POI location
PhysicalLocation.convLocationToVector(location, physicalLocation,
                                locationXyzRelativeToPhysicalLocation);
```

Kuvio 24. Sijaintivektorin muodostaminen

Etäisyyttä laskiessa on huomioitava myös laitteen orientaatio eli käyttäjän katselukulma. Tämä tapahtuu Marker-luokan populateMatrices()-metodissa, jossa lasketaan kohteelle uusi suhteellinen sijainti. Sijainnista muodostetaan uusi vektori aiemmin määriteltyjen sijaintivektorin ja kääntömatriisin pohjalta. Vektori alustetaan ensin lähtemään origosta (kuvio 25, symbolVector-muuttuja) käyttämällä set()-funktiota (yhdessä symbolVector-muuttujan kanssa) ja siihen lisätään edellä laskettu sijaintivektori add()-funktion avulla. Vektori kerrotaan vielä kääntömatriisin kanssa kutsumalla prod()-metodia. (Android-Augment-Reality-Framework: How it works 2012.)

```
// Find symbol position given the rotation matrix
tmpSymbolVector.set(symbolVector);
tmpSymbolVector.add(locationXyzRelativeToPhysicalLocation);
tmpSymbolVector.prod(ARData.getRotationMatrix());
```

Kuvio 25. Sijaintivektorin ja kääntömatriisin tulon muodostaminen

Laskettu vektori (Kuvio 26, muuttuja tmpSymbolVector) projektoidaan lopulta laitteen näytölle kutsumalla CameraModel olion projectPoint()-metodia. (Android-Augment-Reality-Framework: How it works 2012.)

```
cam.projectPoint(tmpSymbolVector, tmpVector, addX, addY);
symbolXyzRelativeToCameraView.set(tmpVector);
cam.projectPoint(tmpTextVector, tmpVector, addX, addY);
textXyzRelativeToCameraView.set(tmpVector);
```

Kuvio 26. Vektorin projektointi laitteen näytölle

5 Päätäntö

5.1 Tulokset

Tavoitteiden mukaisen Augmented Reality -sovelluksen kehitystyö onnistuttiin aloittamaan menestyksekkäästi. Ohjelmisto on yhä kehitysvaiheessa ja vaatii lisää kehitystyötä ennen lopullista valmistumistaan. Testauksen aikana kohteiden sijainnit kovakoodattiin ohjelman muistiin, koska sovellus ei vielä sisällä tukea sijaintipalveluja tarjoaville rajapinnoille. Sovellus löysi testausten aikana etsityt kohteet ja käyttöliittymä toimi tavoitteiden mukaisesti. Testaukset osoittivat myös, että sovelluksen toiminta on täysin riippuvainen GPS–signaalin laadusta. Sisätiloissa paikannus on huomattavasti ulkotiloja epätarkempaa, mikä oli jo etukäteen odotettavissa. Myös magnetometrin toiminta paljastui melko häiriöherkäksi etenkin sähköisten laitteiden lähettyvillä. On myös huomioitavaa, että käyttäjä tullaan opastamaan kohteille tietokannasta haettujen koordinaattien perusteella. Suurin osa tietokannoista ei sisällä tietoa rakennusten korkeudesta tai pinta-alasta, mikä saattaa tulevaisuudessa aiheuttaa väärinkäsityksiä kohteita lähestyessä.

5.2 Yhteenveto

Lisätty todellisuus on vielä melko tuntematon käsite mobiilisovellusten käyttäjien ja kehittäjienkin keskuudessa. Muutamat sovellukset ovat kuitenkin onnistuneet herättämään suurempaakin kiinnostusta, mutta mikään varsinainen ilmiö ei vielä ole kyseessä. Tekniikan rajoituksista huolimatta käyttömahdollisuudet on jo ehditty huomioimaan laajasti etenkin pelimaailmassa ja yrityskäytössäkin. Onkin hyvin todennäköistä, että lähitulevaisuudessa lisätty todellisuus tekee lopullisen läpimurron etenkin mobiilisovellusten keskuudessa.

Aihepiirinä lisätty todellisuus on monipuolinen ja laaja kokonaisuus, mistä johtuen työ rajattiin lähinnä mobiilisovellusten näkökulmaan. Alan kirjallisuutta aihepiiristä ei vielä ole runsaasti saatavilla ja useat julkaisut käsittelevät aihetta melko suppeasti. Sovelluksen pohjana käytetyn runkokoodin dokumentointi oli myös

vähäistä ja sen käyttäminen vaati melkoisesti syventymistä aiheeseen. Myös koodin rakenne ja toiminta oli ymmärrettävä kokonaisuudessaan ennen oman toiminnallisuuden kehittämistä.

Sovelluksen kehittäminen oli haastavaa, mutta samaan aikaan myös palkitsevaa. Augmented Reality -sovellusten kehitystyö vaatii ohjelmointitaitojen lisäksi perusymmärrystä seurantaan käytettävien laitteiden toiminnasta. Työn hankalin osuus oli perehtyminen ennestään tuntemattomaan Android-ohjelmointialustaan ja Augmented Realityn toimintaperiaatteisiin. Lisäksi kehitysympäristö ja tarvittava laitteisto vaati totuttelemista.

LÄHTEET

Activities. Ei päivitystä. [WWW-dokumentti]. Google. [Viitattu 12.05.2013]. Saatavissa: http://developer.android.com/guide/components/activities.html

AndAR - Android Augmented Reality. Ei päivitystä. [WWW-dokumentti]. Google Code Project Hosting. [Viitattu 12.05.2013]. Saatavissa: http://code.google.com/p/andar/

Android. Ei päivitystä. [WWW-dokumentti]. linux.fi. [Viitattu 12.05.2013]. Saatavissa: http://linux.fi/wiki/Android

Mikä on Android. Ei päivitystä. [WWW-dokumentti]. Android suomi. [Viitattu 21.05.2013]. Saatavissa: http://blog.androidsuomi.fi/mika-on-android/

Android-Augment-Reality-Framework: How it works. 2012. [WWW-dokumentti]. Google Code Project Hosting. [Viitattu 12.05.2013]. Saatavissa: http://code.google.com/p/android-augment-reality-framework/wiki/HowItWorks

Application Fundamentals. Ei päivitystä. [WWW-dokumentti]. Google. [Viitattu 12.05.2013]. Saatavissa: http://developer.android.com/guide/components/fundamentals.html

ARQuake: Interactive Outdoor Augmented Reality Collaboration System. Ei päivitystä. [WWW-dokumentti]. Wearable Computer Lab. [Viitattu 12.05.2013]. Saatavissa: http://wearables.unisa.edu.au/projects/arquake/

Benson, T. 2008. Aircraft Rotations. [WWW-dokumentti]. National Aeronautics and Space Administration (NASA). [Viitattu 12.05.2013]. Saatavissa: http://www.grc.nasa.gov/WWW/k-12/airplane/rotations.html

Bimber, O. & Ramesh, R. 2005. Spatial Augmented Reality: Merging Real and Virtual Worlds. 1 painos. Wellesley, Massachusetts: A K Peters.

Bonetti, P. 2012. Nokia City Lens comes out of beta. [WWW-dokumentti]. Conversations by Nokia. [Viitattu 12.05.2013]. Saatavissa: http://conversations.nokia.com/2012/09/10/nokia-city-lens-comes-out-of-beta/

Carmigniani, J. & Furth, B. 2011. Handbook of Augmented Reality. Augmented Reality: An Overview. 1 painos. New York: Springer.

Livingston, M., Rosenblum, L. Brown, D., Schmidt, G., Julier, S., Baillot, Y., Swan, J., Ai, Z & Maassel, P. 2011. Handbook of Augmented Reality. Augmented Reality: Military Applications of Augmented Reality. 1 painos. New York: Springer.

Kan, T., Teng, C & Chen, M. 2011. Handbook of Augmented Reality. Augmented Reality: WR Code Based Augmented Reality Applications. 1 painos. New York: Springer.

GeomagneticField. Ei päivitystietoja. [WWW-dokumentti]. Google. [Viitattu 12.05.2013]. Saatavissa: http://developer.android.com/reference/android/hardware/GeomagneticField.html

GPS vs. aGPS: A Quick Tutorial. 2009. [WWW-dokumentti]. Windows Phone Central. [Viitattu 12.05.2013]. Saatavilla: http://www.wpcentral.com/gps-vs-agps-quick-tutorial

How does ARToolKit work?. Ei päivitystietoja. [WWW-dokumentti]. The Human Interface Technology Lab, University of Washington. [Viitattu 12.05.2013]. Saatavissa: http://www.hitl.washington.edu/artoolkit/documentation/userarwork.htm

HowToBuildApplicationsBasedOnAndAR. 2010. [WWW-dokumentti]. Google Code Project Hosting. [Viitattu 12.05.2013]. Saatavissa: http://code.google.com/p/andar/wiki/HowToBuildApplicationsBasedOnAndAR

Hyrrävoimat ja gyroskooppi. Ei päivitystä. [Videotallenne]. Opetus tv. [Viitattu 21.05.2013]. Saatavissa: http://opetus.tv/fysiikka/fy5/hyrravoimat-ja-gyroskooppi/

Interactive SAR. Ei päivitystietoja. [WWW-dokumentti]. Wearable Computer Lab. [Viitattu 12.05.2013]. Saatavissa: http://wearables.unisa.edu.au/projects/interactive-sar/

Introduction to ARToolKit. Ei päivitystä. [WWW-dokumentti]. The Human Interface Technology Lab, University of Washington. [Viitattu 12.05.2013]. Saatavissa: http://www.hitl.washington.edu/artoolkit/documentation/userintro.htm

Lego Digital Box Augmented Reality Kiosk. Ei päivitystä. [WWW-dokumentti]. Mentaio. [Viitattu 12.05.2013]. Saatavissa: http://www.metaio.com/kiosk/lego

Miettinen, S. 2006. GPS Käsikirja. 3. painos. Porvoo: WS Bookwell Oy.

Mixare – Open Source Augmented Reality Engine. Ei päivitystä. [WWW-dokumentti]. Mixare. [Viitattu 21.05.2013]. Saatavissaa: http://www.mixare.org/

NyARToolkit project. Ei päivitystä. [WWW-dokumentti]. The NyARToolkit project website. [Viitattu 12.05.2013]. Saatavissa: http://nyatla.jp/nyartoolkit/wp/?page_id=198

Position Sensors. Ei päivitystä. [WWW-dokumentti]. Google. [Viitattu 12.05.2013]. Saatavissa: http://developer.android.com/guide/topics/sensors/sensors_position.html

Rivington, J. 2013. Google Glass: what you need to know. [WWW-dokumentti]. TechRadar. [Viitattu 12.05.2013]. Saatavissa: http://www.techradar.com/news/video/google-glass-what-you-need-to-know-1078114

Sensorama Machine. Ei päivitystä. [WWW-dokumentti]. mortonheilig.com. [Viitattu 12.5.2013]. Saatavissa: http://www.mortonheilig.com/InventorVR.html

Studierstube Tracker. 2011. [WWW-dokumentti]. The Christian Doppler Labor for Handheld AR. [Viitattu 12.05.2013]. Saatavissa: http://handheldar.icg.tugraz.at/stbtracker.php

Sung, D. 2011. The history of augmented reality [WWW-dokumentti]. Pocket-lint. [Viitattu 12.05.2013]. Saatavissa: http://www.pocket-lint.com/news/108888-the-history-of-augmented-reality

Techopedia explains Accelerometer. Ei päivitystietoja. [WWW-dokumentti]. Techopedia. [Viitattu 21.05.2013]. Saatavissa: http://www.techopedia.com/definition/24430/accelerometer

Understanding Smart Phone Sensor Performance: Magnetometer. 2011. [WWW-dokumentti]. Sensor Platforms. [Viitattu 12.05.2013]. Saatavissa: http://www.sensorplatforms.com/understanding-smart-phone-sensor-performance-magnetometer-2/

Yadav, M. 2011. History of Android. [WWW-dokumentti]. Tech2Crack. [Viitattu 12.05.2013]. Saatavissa: http://www.tech2crack.com/history-android/

Young, D. 2010. Myron Krueger. [WWW-dokumentti]. Inventing Interactive. [Viitattu 12.05.2013]. Saatavissa: http://www.inventinginteractive.com/2010/03/22/myron-krueger/