Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Opinnäytetyö

Timo Härkönen

Qt-sovelluksen kehittäminen Maemo-alustalle

Työn ohjaaja Lehtori Jari Mikkolainen
Työn teettäjä Kilosoft Oy, valvojana diplomi-insinööri Mika Pihlamo
Tampere 3/2009

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tekijä	Timo Härkönen
Työn nimi	Qt-sovelluksen kehittäminen Maemo-alustalle
Sivumäärä	38
Valmistumisaika	05/2009
Työn ohjaaja	Lehtori Jari Mikkolainen
Työn teettäjä	Kilosoft Oy, valvoja DI Mika Pihlamo

TIIVISTELMÄ

Tämä työ käsittelee Qt-sovelluksen kehittämistä Maemo-alustalle. Työn tarkoituksena oli perehtyä Maemo-alustaan ja Qt-ohjelmointiin.

Työssä esitellään Maemo-kehitysalusta ja tutustutaan Qt-ohjelmointiin sekä käsitellään Maemo-alustalle tehtävässä Qt-sovelluskehityksessä huomioitavia seikkoja.

Työn sisällön perusteella aiheeseen tarkemmin perehtymättömän lukijan toivotaan saavan hyvän peruskäsityksen Maemo-alustasta ja Qt-ohjelmoinnista. Työn lukijalla oletetaan olevan hyvät perustiedot C++-ohjelmoinnista.

Työssä esiintyvissä esimerkeissä käytetään taustatyöksi tehtyä, langattomien tukiasemien paikantamiseen tarkoitettua, sovellusta. Sovellus esitellään työn lopuksi.

Tässä työssä ei käsitellä Qt-ohjelman kehittämisprosessia, joka käyttää Qt Designer -työkalua.

Avainsanat	Qt-ohjelmointi, Maemo, Linux, graafiset käyttöliittymät, Debian-paketointi

TAMK University of Applied Sciences
Programme in information technology
Software engineering

Author(s)	Timo Härkönen
Name of the report	Developing a Qt Application to the Maemo platform
Number of pages	38
Graduation time	May 2009
Thesis supervisor	Jari Mikkolainen
Commissioned by	Kilosoft Oy, supervisor Mika Pihlamo

ABSTRACT

This thesis covers the basics knowledge needed to develop a Qt application to the Maemo platform. The motivation behind the thesis was to learn more about Qt programming and the Maemo platform.

The Qt application used in this thesis' examples was developed as background work for this thesis. The application helps a user to find the closest wireless access points to his/her current location. The application downloads access point data from the web and shows the points closest to user, using the device's build-in GPS. The application is presented at the end of this thesis.

People with good C++ programming skills should get a good introduction to the subject in question by reading this thesis.

This thesis does not covet Qt application development with Qt Creator.

Keywords: Qt programming, Maemo, Linux, Graphical user interfaces, Debian packaging

Alkusanat

Tämä opinnäytetyö on tehty lokakuun 2008 ja maaliskuun 2009 välisenä aikana. Aiheen valintaan vaikutti suuresti henkilökohtainen kiinnostukseni Maemo-alustaa ja graafisten käyttöliittymien ohjelmointia kohtaan. Pienen harkinnan ja vaihtoehtojen tarkastelun jälkeen työn aiheeksi valikoitui Qt-sovelluksen kehittäminen Maemo-alustalle. Aihetta voidaan pitää onnistuneena valintana, sillä sen tekeminen tuntui mielekkäältä ja se lisäsi mielenkiintoani tutustua aiheeseen entistä syvällisemmin.

Haluaisin esittää kiitokset työtä valvoneelle diplomi-insinööri Mika Pihlamolle saamastani ohjauksesta ja rakentavasta palautteesta.

Tampereella 24. maaliskuuta 2009

Timo Härkönen

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Sisällys

Termit ja lyhenteet

ARM	Sulautetuissa järjestelmissä yleisesti käytetty prosessoriarkkitehtuuri
Armel	ARM-prosessoriarkkitehtuurin emulaattori
Debian	GNU/Linux-käyttöjärjestelmä
GNOME	Avoimen lähdekoodin työpöytäympäristö
GNU	Avoimen lähdekoodin projekti, joka kehittää työkaluja, kirjastoja ja muita ohjelmistokomponentteja.
GNU Autotools	Kokoelma avoimen lähdekoodin työkaluja ohjelman käännösprosessin automatisoimiseksi
GObject	Glib Object system. Tarjoaa olio-ohjelmointikehyksen C-ohjelmointikielelle.
GTK+	Kokoelma työkaluja ja kirjastoja alustariippumattomaan graafisten käyttöliittymien kehitykseen
Hildon	Linux-pohjaisille mobiililaitteille kehitetty käyttöliittymäkehys
Internet-tabletti	Nokia Oyj:n valmistama Maemo-alustaa käyttävä päätelaite
Linux	Avointa lähdekoodia oleva käyttöjärjestelmän ydin
Maemo	Mobiililaitteille tehtävään sovelluskehitykseen tarkoitettu alusta
Natiivisovellus	Ainoastaan kohdealustan omia kirjastoja käyttäen kehitetty sovellus, joka voidaan suorittaa ilman lisäkirjastojen asentamista.
pkg-config	Apuohjelma käännöksessä käytettävien kirjastojen selvittämiseen
Qt	Kokoelma työkaluja ja kirjastoja alustariippumattomaan graafisten käyttöliittymien kehitykseen
Scratchbox	Eristetty kehitysympäristö lähdekoodin kääntämiseen eri kohdearkkitehtuureille
Signaalit ja lokerot	Signals and Slots, Qt:n käyttämä olioiden välinen viestinvälitystekniikka
X11	GNU/Linux-käyttöjärjestelmissä käytetty ikkunointijärjestelmä, X Window Server versio 11
x86	PC-tietokoneissa yleisesti käytetty prosessoriarkkitehtuuri

1 Johdanto

Tässä työssä käsitellään Qt-sovelluksen kehittämistä Maemo-alustalle. Työ jakautuu neljään toisiinsa sidoksissa olevaan osaan. Työn osat muodostavat kokonaisuuden, jonka avulla aiheeseen tarkemmin perehtymättömän lukijan toivotaan saavan hyvän peruskäsityksen Qt-sovelluksen kehittämisestä Maemo-alustalle.

Työn toisessa luvussa tutustutaan Maemo-alustan perusrakenteeseen ja alustalle tehtävään ohjelmistokehitykseen. Maemo on Nokian Linux-pohjaisille kämmentietokoneille kehittämä avoin kehitysalusta, joka pohjautuu vahvasti Debian GNU/Linux-käyttöjärjestelmäjakeluun. Maemo-alustaa hyödyntäviä laitteita ovat Nokia Oyj:n valmistamat Internet-tabletit N770, N800 ja N810.

Kolmannessa luvussa esitellään graafinen ohjelmistokehityskirjasto Qt ja sen tuomat lisäyksen C++-ohjelmointiin. Qt on alustariippumaton graafinen ohjelmointikirjasto, joka tuo muutamia merkittäviä lisäominaisuuksia C++-ohjelmointiin. Qt lisää C++:aan esimerkiksi tekniikan olioiden väliseen kommunikaatioon ja valmiit luokat XML-muotoisen tiedon ja tietokantojen käsittelemiseen. Qt on tehty ohjelmoijan kannalta helpoksi omaksua ja sen tarjoamat ratkaisut ovat suoraviivaisia käyttää.

Työn neljännessä luvussa käydään lävitse huomioonotettavia seikkoja kehittäessä Qt-sovellusta Maemo-alustalle. Esitettyjä seikkoja ovat käyttöliittymän erityispiirteet, syötteen antaminen sekä ohjelman paketointi.

Työn viidennessä luvussa esitellään taustatyöksi toteutettu sovellus, Hotspot Radar.

2 Maemo

Maemo on Nokia Oyj:n kehittämä avoimen lähdekoodin ohjelmistokehitysalusta mobiileille Linux-päätelaitteille. Maemo sisältää ohjelmat ja työkalut, joita tarvitaan alustaa hyödyntäville päätelaitteille tehtävässä ohjelmistokehityksessä. Alusta sisältää ohjelmia lukuisista avoimen lähdekoodin projekteista, kuten Linux-ydin, Debian, GNU ja GNOME. Maemo alustaa käytetään Nokia Oyj:n valmistamissa Internettableteissa. /8/

Internet-tabletit ja niiden kehitysalustana käytettävä Maemo rakentuvat lähes täysin avoimen lähdekoodin ohjelmakoodista. Osa Maemo-alustan sisältämästä ohjelmakoodista on kuitenkin suljettua lähdekoodia. Suljettua lähdekoodia Maemo-alustassa ovat esimerkiksi sen käyttämä mediasoitin ja selaimen käyttöliittymäkirjasto. /11/

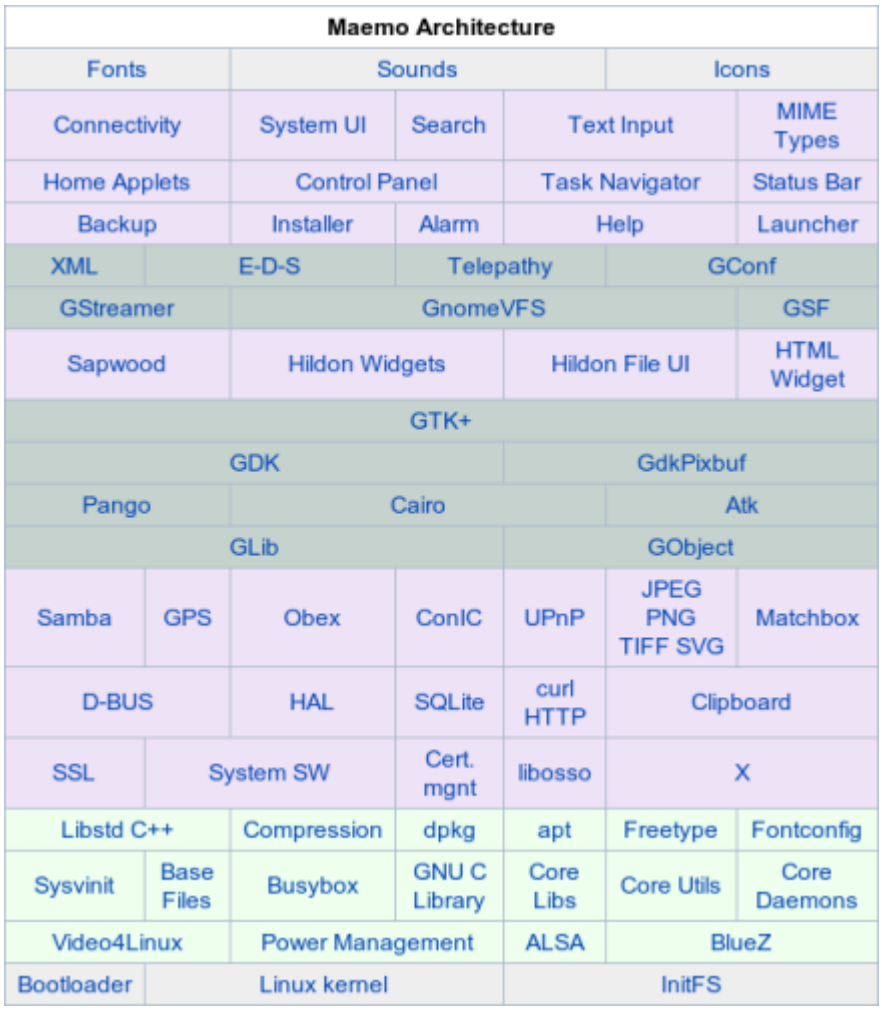
Suljetuista osistaan huolimatta Maemo-alustaa voidaan pitää avoimena kehitysalustana. Avoimen lähdekoodin määritelmä on esitetty liitteessä 1 /7/.

Maemo alustaa käyttävä päätelaite Nokia N810 on esitetty kuviossa 1.

Kuvio 1: Nokia N810 Internet-tabletti /10/

2.1 Arkkitehtuuri

Maemo rakentuu käyttöjärjestelmän ytimen osalta Linux-ytimen ja GNU-projektin kirjastojen varaan. Käyttöliittymä on rakennettu pääasiasiallisesti käyttäen GNOME- ja GTK+ -kirjastoja. Maemo-alustan käyttämiä kirjastoja ja alustan arkkitehtuuria on havainnollistettu kuviossa 2.



Kuvio 2: Maemo-alustan arkkitehtuuri (Maemo Technology Overview)

Maemo rakentuu Linux-ytimen päälle. Linux-ytimen yläpuolella arkkitehtuurin toisessa kerroksessa ovat virranhallinta-, Bluetooth- ja C-kirjastot sekä muut alustan tarvitsemat matalan tason kirjastot. Kolmannessa kerroksessa ovat erilaisten yhteyksien hallintaan liittyvät osat, kuten prosessien väliseen viestinvälitykseen käytettävä D-BUS. Neljännessä kerroksessa ovat GTK+ -kirjastot, joita käytetään graafisten elementtien esittämiseen. Viidennessä kerroksessa on multimediaan ja tiedostojen käsittelyyn liittyviä rajapintoja, kuten tiedostojärjestelmärajapinta GnomeVFS- ja multimediatiedostojen toistamiseen käytetty GStreamer-rajapinta. Kuudennessa kerroksessa on GTK+ -kirjastoihin tehdyt laajennukset, kuten Hildon-käyttöliittymäelementit ja käyttöliittymän teemamoottori Sapwood. Seitsemännessä kerroksessa on alustan näyttäjälle näkyviä ohjauskomponentteja, kuten ohjauspaneeli. Arkkitehtuurin ylimmällä tasolla ovat alustan käyttämät ikonit, äänet ja fontit.

Alustan arkkitehtuuriin sisältyvistä osista Linux-ydin, GTK+ ja Hildon on esitelty tarkemmin seuraavissa alaluvuissa.

2.1.1 Linux-ydin

Ydin on käyttöjärjestelmän sydän. Se tarjoaa alimman tason abstraktion muistin, laitteiden ja prosessorin käsittelyyn.

Linux-ydin on arkkitehtuuriltaan monoliittinen. Monoliittinen ydin sisältää tarvittavat laiteajurit ja moduulit. Moduuleja voidaan ladata ytimeen ajonaikaisesti. Ydin toimii omassa virtuaalisessa muistitilassaan (kernel space) erillään käyttäjän käynnistämistä ohjelmista. Käyttäjän käyttämät ohjelmat toimivat omassa tilassaan (user space). Ohjelmat pyytävät ytimeltä palveluita käyttämällä systeemikutsuja.

2.1.2 GTK+

GTK+ on kokoelma C-ohjelmointikielellä toteutettuja työkaluja ja kirjastoja graafisten käyttöliittymien kehittämiseen. GTK+ käyttää GObject-oliojärjestelmää, ja se sisältää useita valmiita käyttöliittymäkomponentteja. GTK+ kehitettiin alun perin GIMP-kuvankäsittelyohjelman ikkunointijärjestelmäksi, josta GTK:n nimi, Gimp Tool Kit, on peräisin. /14/

GTK+ on tapahtumapohjainen järjestelmä. Ohjelman ollessa suorituksessa, se odottaa tapahtumasilmukassa jonkin toimenpiteen aiheuttamia tapahtumia. Kun käyttäjä esimerkiksi painaa hiiren painiketta, tapahtumasilmukka herää ja lähettää tapahtuman yhdelle tai useammalle käyttöliittymäkomponentille. Käyttöliittymäkomponentti puolestaan lähettää tapahtuman vastaanottaessaan signaalin tapahtuman käsittelevälle takaisinkutsufunktiolle. Takaisinkutsufunktio käsittelee tapahtuman, ja ohjelman suoritus palaa tapahtumasilmukkaan odottamaan uusia tapahtumia. /8/

2.1.3 Hildon

Hildon on GTK+:aa laajentava ohjelmistokehys, jota käytetään pienikokoisten kannettavien laitteiden käyttöliittymissä. Hildon sisältää käyttöliittymäkomponentteja, kosketusnäyttöä tukevan syötteenlukujärjestelmän, työpöytäympäristön ja työpöydän pikkuohjelmat. Hildon sisältää myös varmuuskopiointi- ja apukehyksen sekä sovellusten hallinnan. /5/

Hildon sisältää esimerkiksi GTK:n pääikkunan ylikuormittavan HildonWindow-elementin, jota käytetään Maemo-sovellusten pääikkunana. HildonWindow antaa sovellusikkunalle Hildon-työpöydän ulkonäön ja toiminnallisuuden.

Hildon kehyksen sisältämä syötteen lukumekanismi mahdollistaa käyttäjän syötteen lukemisen kosketusnäyttönäppäimistöltä ja kosketusnäytön käsikirjoittamisen tunnistuksen. /16/

2.2 Ohjelmistokehitys

Maemo-alustalle tehtävä ohjelmistokehitys on hyvin samankaltaista kuin tavalliselle Linux-alustalle tehtävä ohjelmistokehitys. Kehitysympäristö sisältää samat työkalut ja kirjastot kuin Debian GNU/Linux-jakelu.

Alustalle tehtävään ohjelmistokehitykseen käytetään pääasiallisesti C-ohjelmointikieltä. Alustalle on mahdollista kehittää myös C++-, Qt- tai Python-ohjelmia. Tuki muille ohjelmointikielille on asennettava kehitysympäristöön erikseen. /5/

Maemo-alustan natiivisovellukset kirjoitetaan C-ohjelmointikielellä käyttäen GTK+- ja Hildon-kirjastoja. Suurin osa GTK+-sovelluksista toimii alustalla ilman suuria muutoksia. Merkittävin muutos on GTK+:n pääikkunan korvaaminen Hildon-ikkunalla. /15/

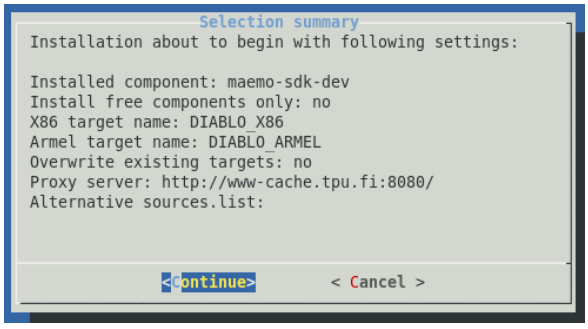
Alustalle kehitetyt sovellukset ristiinkäännetään alustan arkkitehtuurille Scratchbox-hiekkalaatikkoympäristössä. Ristiinkääntäminen on välttämätöntä, koska kehitysympäristön sisältämät PC-tietokoneet ja Maemo-alustaa käyttävät päätelaitteet käyttävät eri prosessoriarkkitehtuuria eivätkä niiden käyttämät binäärikirjastot ole keskenään yhteensopivia.

2.2.1 Kehitysympäristö

Maemo-alustalle tehtävä ohjelmistokehitys tehdään Linux-työasemalle asennettavassa Scratchbox-hiekkalaatikkoympäristössä. Hiekkalaatikkoympäristö sisältää päätelaitteen ympäristön sekä tarvittavat ohjelmistokehitystyökalut. Scratchbox mahdollistaa ohjelmakoodin kääntämisen muille kuin isäntäjärjestelmän arkkitehtuurille. /5/

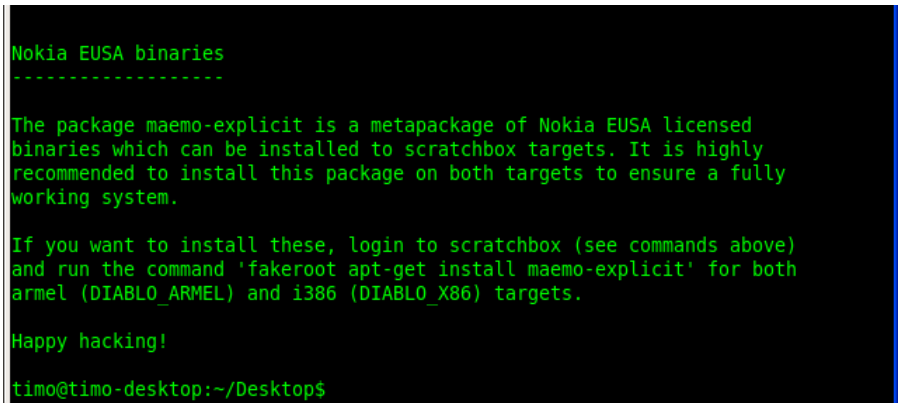
Maemo-kehitysympäristö sisältää työkalut ohjelmakoodin kääntämiseksi ARM- ja x86-prosessoriarkkitehtuureille. x86-kohdearkkitehtuuria käytetään kehitettävän ohjelman suorittamiseen PC-tietokoneella ilman, että päätelaitteen arkkitehtuuria tarvitsee emuloida. x86-kohdearkkitehtuurille käännetyillä ohjelmilla on parempi suorituskyky ja tuki isäntäjärjestelmän työkaluille. ARM-kohdearkkitehtuuria käytetään kääntämään kehitettävästä ohjelmasta päätelaitteessa suoritettava versio. /5/

Kehitysympäristön asennettavia kohdearkkitehtuureja on havainnollistettu kuviossa 3.



Kuvio 3: Kehitysympäristön asennuksen valintojen yhteenveto.

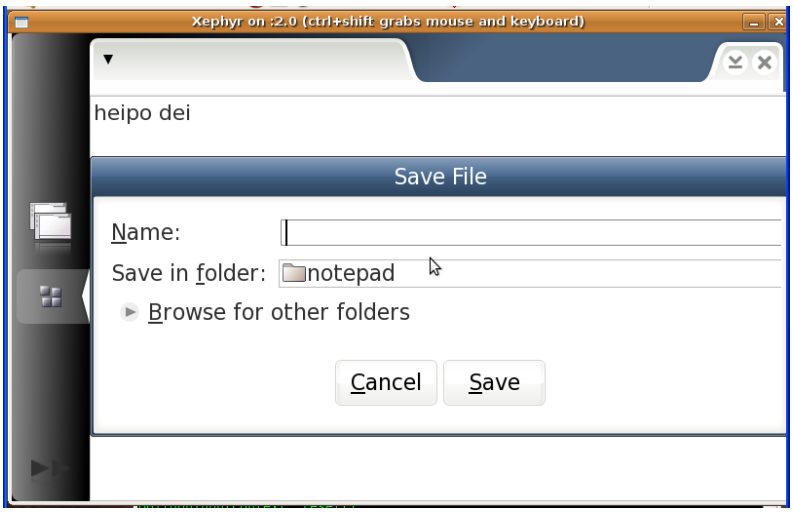
Kehitysympäristöön sisältyy myös Nokian suljettuja ohjelmistokomponentteja. Näitä komponentteja ei asenneta kehitysympäristöön oletuksena. Komponentit on asennettava erikseen, jotta kehitysympäristöstä saadaan täysin toimiva. Asennuskomentosarja kehottaa asennuksen päätteeksi asentamaan komponentit. Kehotus on esitetty kuviossa 4.



Kuvio 4: Kehotus asentaa Nokian suljetut komponentit kehitysympäristöön.

Kehitettävää ohjelmaa voidaan ajaa PC-tietokoneella. Hildon-työpöytäympäristö tarvitsee kuitenkin toissijaisen ikkunointipalvelimen, johon on asetettu päätelaitetta vastaava resoluutio ja värisyvyys. Päätelaitteen näytön resoluutio on 800x480 ja värisyvyys 16 bittiä kuvapistettä kohden. /5/

Maemo-ohjelmistokehityksessä päätelaitteen näkymän simuloimiseen käytetään Xephyr-ikkunointipalvelinta, johon Hildon sovelluskehys käynnistetään. Kun kehitysympäristöstä käynnistetään ikkunointia tarvitsevia ohjelmia, näytetään ne Xephyrin sovellusikkunassa. Maemo-alustalla kehitetyn ohjelman suorittamista Xephyr-ikkunointipalvelimen avulla on havainnollistettu kuviossa 5.



Kuvio 5: Sovelluksen suorittaminen Xephyr-ikkunointipalvelimessa.

3 Qt

Qt on alustariippumaton graafinen sovelluskehityskirjasto, jonka avulla saman ohjelmakoodin saa käännettyä Windows-, Mac OSX- ja Linux-käyttöjärjestelmille. Qt:n on kehittänyt norjalainen ohjelmistoyritys Trolltech. Nokia Oyj osti Trolltech:n 29. syyskuuta 2008 tukemaan alustariippumatonta sovelluskehitystään ja nimesi yrityksen uudelleen. Yritys tunnetaan nykyään nimellä Qt Software. /18/

Qt toimii standardin C++-kielen laajennuksena. C++:n lisäksi Qt tukee virallisesti Java- ja JavaScript-ohjelmistokehitystä. Kolmannet osapuolet tarjoavat lisäksi tuen monelle muulle ohjelmointikielelle, kuten Python, PHP ja .Net. Qt sisältää kirjastot mm. graafisten käyttöliittymien, tietoverkkojen, tietokantojen, 3D-grafiikan kehittämiseen. Qt sisältää oman oliomallinsa, merkkijonojen käsittelyn ja muut perusohjelmoinnissa tarvittavat komponentit. Qt:n merkittävin lisäys standardiin C++:aan on olioiden välisessä kommunikoinnissa käytettävä signaali-lokero -mekanismi. /1/

3.1 Qt:n luokkamalli

Qt:ssa kaikki luokat periytyvät suoraan tai epäsuorasti QObject-luokasta. Luokista luodut oliot järjestävät itsensä hierarkkisiksi oliopuiksi, siten että dynaamisesti luoduilla olioilla on niiden muistinvarauksen vapauttamisesta vastaava vanhempi olio. Näin luodusta oliosta tulee vanhemman olion lapsiolio. Lapsiolioilla voi olla omalla vastuullaan olevia lapsia. Kun olio tuhoutuu, sen vastuulla olevat lapsioliot tuhotaan ja niiden muistinvaraukset vapautetaan automaattisesti. /6/

C++:aan tehtyjen laajennusten vuoksi Qt:ssa olioita käsitetään identiteetteinä, ei arvoina. Arvot kopioidaan tai sijoitetaan muuttujiin. Näin ei voida menetellä käsiteltäessä olioita identiteetteinä. Qt:n oliot sisältävät tietoa mm. sijainnistaan oliohierarkiassa ja olioon kytketyistä signaaleista. Oliosta ei voida tehdä identtistä kopiota. Tämän sijasta oliot kloonataan ja sille luodaan uusi identiteetti. Kaikissa QOject-luokasta suoraan tai epäsuorasti periytyvissä luokissa kopiorakentaja ja sijoitusoperaattori on poistettu käytöstä. /6/

3.1.1 Signaalit ja lokerot

Qt:n merkittävin lisäys perinteiseen C++:n luokkakäsitteeseen ovat signaalit ja lokerot. Signaalit ja lokerot ovat Qt:n tapa hoitaa olioiden välinen kommunikaatio.

Signaalit ainoastaan esitellään luokan esittelytiedostossa eikä niille kirjoiteta toteutusta. Signaaleilla ei voi olla paluuarvoa, joten niiden paluuarvon tyypiksi on määriteltävä void.

Lokerot esitellään luokan esittelytiedostossa ja niille kirjoitetaan toteutus toteutustiedostoon. Lokeroille voidaan määrittää samat näkyvyysmääreet kuin normaaleille C++:n metodeille. Näkyvyyssäännöillä on ainoastaan merkitystä, kun lokeroita käytetään normaalien metodien tapaan. Lokerot eroavat normaaleista C++-funktioista siten, että niihin voidaan kytkeä signaaleja. Esimerkissä 1 on esitetty yksinkertainen signaaleja ja lokeroita sisältävän Qt-luokan esittelytiedosto.

Esimerkki 1: Qt-luokan esittelytiedosto

```
class Class : public QWidget
{
    Q_OBJECT
    Q_PROPERTY(QColor color READ getColor WRITE setColor)
public:
    Class(QObject *parent = 0);
    QColor getColor() const;
public slots:
    void setColor(const QColor &newColor);
signals:
    void colorChanged(const QColor);
private:
    QColor color;
};
```

Esimerkissä 1 käytetään kahta makroa, Q_OBJECT ja Q_PROPERTY. Q_OBJECT -makro on pakollinen kaikissa luokissa, joissa määritellään signaaleja tai lokeroita. Makroa käytetään kertomaan, että siitä on käännöksen yhteydessä luotava meta-luokka. Q_PROPERTY -makron avulla kerrotaan luokan ominaisuuksista. Ominaisuudet ovat

luokan tietojäseniä, joita voidaan lukea ja asettaa Meta-luokkien avulla. /6/ Meta-luokista ja Qt-ohjelman käännösprosessista on kerrottu tarkemmin luvussa 3.2.

Signaalit kytketään lokeroihin käyttämällä QObject-luokan metodia connect. Yksi signaali voidaan kytkeä kuinka moneen lokeroon tahansa ja yhteen lokeroon voidaan kytkeä kuinka monta signaalia tahansa. Lisäksi signaalit voidaan kytkeä toisiin signaaleihin.

Signaali kytketään lokeroon esimerkin 2 mukaisesti, jossa update-olion signaali updateInProgress(bool) kytketään this-olion setUpdateFlag(bool) lokeroon.

Esimerkki 2: Signaalin kytkeminen lokeroon.

```
QObject::connect(
              update,
              SIGNAL(updateInProgress(bool)),
              this,
              SLOT(setUpdateFlag(bool))
              );
```

Qt:n valmiit komponentit sisältävät valmiiksi määriteltyjä signaaleja ja lokeroita. Komponentit lähettävät signaaleja eri tapahtumien yhteydessä, kuten napin painamisen, ikkunan sulkemisen tai tiedonsiirron valmistumisen yhteydessä. Itse määritetyt signaalit lähetetään esimerkissä 3 esitetyllä tavalla.

Esimerkki 3: Signaalin lähettäminen

```
emit updateInProgress(false);
```

Kun signaali lähetetään, välitetään se kaikkiin siihen kytkettyihin lokeroihin. Lokeroiden suoritusjärjestykseen ei voida vaikuttaa. /1/

3.1.2 Tapahtumat

Qt on monen muun graafisen ohjelmointikirjaston tapaan tapahtumapohjainen. Tapahtumapohjainen sovellus odottaa tapahtumasilmukassa, kunnes jokin tapahtuma ilmenee. Tapahtumat syntyvät esimerkiksi käyttäjän antamista syötteistä, ohjelman suorituksen alkamisesta tai päättymisestä. Kun käyttäjä esimerkiksi painaa jotain näppäimistön pai-

niketta, syntyy näppäimistötapahtuma, ja tapahtumasilmukka kutsuu tämän tapahtuman käsittelevää ohjelman osaa.

Tapahtumien käsittelyfunktiot ovat Qt:ssa virtuaalisia, joten ne voidaan ylikuormittaa periyttämistä käyttämällä ja toteuttaa vaihtoehtoinen käsittely halutuille tapahtumille. Esimerkissä 4 on esitetty ylikuormitettu näppäimistötapahtuma, joka erottaa näppäimistötapahtumista F6-näppäimen painalluksen.

Esimerkki 4: Tapahtuman ylikuormittaminen perityssä luokassa

```
/** esittelytiedosto **/
…
protected:
    void keyPressEvent(QKeyEvent *event);
…

/** toteutustiedosto **/
void Class::keyPressEvent(QKeyEvent *event);
{
    if(event->key() == Qt::key_F6) {
        qDebug() << ”F6 pressed”;
    }
}
```

3.1.3 Automaattinen optimointi

Qt käyttää automaattisesti joitain ohjelman suorituskykyä parantavia optimointikeinoja. Esimerkiksi muuttujien sijoituksen yhteydessä Qt käyttää ”kopioi kirjoittaessa (copy on write)” -menetelmää, jossa sijoitettu muuttuja kopioidaan vasta, kun sen arvo muuttuu alkuperäisestä. Sijoitettu muuttuja pitää sisällään viittauksen alkuperäiseen muuttujaan sijoitetun arvon muuttumiseen saakka. Kun arvo muuttuu, sijoitettu muuttuja kirjoitetaan. Tästä syystä Qt-ohjelmissa ei parametrien välittämisestä viittauksina funktioille, jotka eivät muuta parametrin sisältöä, ole suoritusta enempää optimoivaa vaikutusta, koska tämä optimointi tehdään jo automaattisesti. /2/

3.1.4 Luokkakirjasto

Qt sisältää monia ohjelmoijan työtä helpottavia luokkia. Qt sisältää esimerkiksi helppokäyttöiset tietokanta- ja verkkoyhteyksien käsittelemiseen tarkoitetut luokat. Qt:n luokkia on helppo laajentaa tai muokata käyttämällä periyttämistä.

Qt:n luokat on jaettu useampaan moduuliin. Jotta jonkin moduulin sisältämiä luokkia voidaan käyttää, on moduuli sisällytettävä ohjelman projektitiedostoon. Qt:n moduulit on esitelty taulukossa 1.

Taulukko 1: Qt:n moduulit /9/

Moduuli	**Käyttötarkoitus**
QtCore	Muiden moduulien käyttämät ei-graafiset ominaisuudet
QtGui	Graafiset käyttöliittymäkomponentit
QtNetwork	Verkkoyhteyksien käsittely
QtOpenGL	OpenGL-ohjelmoinnissa käytettävät luokat
QtScript	Qt:n komentosarjojen käyttämiseen tarkoitetut luokat
QtSql	Tietokantojen käsittely
QtSvg	SVG-tiedostojen esittämiseen käytetyt luokat
QtWebKit	www-sisällön esittämiseen ja käsittelyyn tarkoitetut luokat
QtXml	XML-tiedostojen käsittely
QtXmlPatterns	Muokattujen tietomallien käsittely
Phonon	Multimediakehyksen luokat
Qt3Support	Qt versio 3 yhteensopivuuden mahdollistavat luokat
QtDBus	Unix-alustalla suoritettavien prosessien välinen kommunikaatio

Esimerkissä 5 esitetään Internet-sivun sisällyttäminen sovellukseen, jolla havainnollistetaan Qt:n valmiiden luokkien helppokäyttöisyyttä.

Esimerkki 5: Internet-sivun esittäminen sovelluksessa.

```
QWebView *view = new QWebView(this);
view->load(QUrl(“www.maemo.org”));
view->show();
```

3.2 Qt-ohjelman käännösprosessi

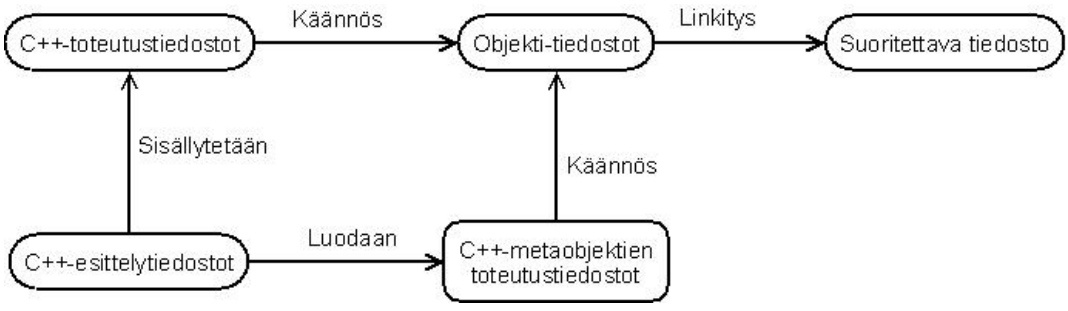
Qt-ohjelmien kääntämisessä suoritettaviksi tiedostoiksi käytetään qmake-työkalua. Qmake luo projektitiedoston sisältämien ohjeiden perusteella isäntäjärjestelmäkohtaisen komentosarjan, jolla ohjelma käännetään. Qmake käyttää apunaan moc-työkalua, joka huolehtii tarvittavien meta-objektien luomisesta. Esimerkissä 6 on esitetty yksinkertainen projektitiedosto. Tiedostossa määritellään projektissa käytettävät lähdekooditiedostot ja käännöksessä luotavan binääritiedoston nimi.

Esimerkki 6: Qt:n projektitiedosto.

```
TEMPLATE = app
TARGET = clock
DEPENDPATH += .
INCLUDEPATH += .

#input
HEADERS += clock.h
SOURCES += clock.cpp main.cpp
```

Qt-ohjelman käännösprosessi eroaa normaalista C++-ohjelman kääntämisestä meta-objektien automaattisen luomisen osalta. Meta-objektit luodaan luokista, joiden esittelytiedostoihin on liitetty Q_OBJECT -makro. Qt:n käännösprosessi on esitetty kuviossa 6.



Kuvio 6: Qt-ohjelman käännösprosessi /6/

3.2.1 MOC

MOC eli Meta Object Compiler on työkalu, joka hallitsee Qt:n C++-laajennuksia. MOC lukee esittelytiedoston ja luo siitä standardille C++-kääntäjälle sopivan luokan meta-tiedot sisältävän lähdekooditiedoston, mikäli esittelytiedostoon on sisällytetty Q_OBJECT -makro. Lisäksi MOC käsittelee muitakin makroja, kuten esimerkiksi Q_PROPERTY, Q_ENUMS, ja Q_CLASSINFO.

Q_PROPERTY -makroa käytetään asettamaan luokan jokin ominaisuus ja sen lukemiseen ja kirjoittamiseen käytettävät funktiot. Q_ENUMS -makroa käytetään ilmaisemaan enumeraatiota, joita voidaan käyttää Qt:n ominaisuusjärjestelmässä. Q_CLASSINFO -makron avulla luokkaan voidaan tallentaa nimi-arvopareina lisätietoa sen ominaisuuksista.

Mikäli ohjelman kääntämiseen käytetään qmake-työkalua, ei MOC:a tarvitse kutsua erikseen. Qmake kutsuu työkalua tarvittaessa automaattisesti. /6/

3.2.2 UIC

UIC eli User Interface Compiler on työkalu, joka luo Qt Designer -työkalulla luoduista XML-muotoisista ui-tiedostoista vastaavat C++-esittelytiedostot. Mikäli ohjelmassa käytettään ui -tiedostoja ja kääntäminen suoritetaan käyttämällä qmake-ohjelmaa, ei työkalua tarvitse kutsua erikseen. /1/

Qt Designerin avulla tehtävää ohjelmistokehitystä ei käsitellä tarkemmin tässä työssä.

4 Qt-sovelluksen kehittäminen Maemo-alustalle

Valmis Qt-ohjelma saadaan toimimaan Maemo-alustalla kääntämällä ohjelmakoodi kehitysympäristössä ARM-kohdearkkitehtuurille. Kehitettäessä ohjelmia Maemo-alustalle on otettava huomioon joitain alustan erityispiirteitä, kuten päätelaitteen kosketusnäyttö ja suuremmaksi skaalautuvat tekstit ja muut graafiset komponentit.

4.1 Maemo Qt

Qt:n Maemo-versiota kehitetään yhteisön voimin. Nokia Oyj:n virallisesti tukema versio Maemo-alustalle on tulossa Maemo-alustan versioon, joka tunnetaan koodinimellä Harmattan. /8/

Maemo-alustan ominaisuuksiin sopeutettu Qt:n versio pitää sisällään samat ohjelmointirajapinnat kuin muillekin alustoille kohdennettu Qt. Maemo-alustaa varten Qt:n ohjelmointirajapintaa on hieman laajennettu. Laajennukset pitävät sisällään esimerkiksi keinon kaksoispainalluksen alueen koon muokkaamiseen, kosketusnäyttöön kohdistuvan painalluksen voimakkuuden lukemiseen, syötteen lukemisen käyttäen Hildon-kehystä ja päätelaitteen erikoisnäppäimien käyttämisen. /11/

Tätä kirjoittaessa rajapintaan tehdyt laajennukset eivät olleet vielä lopullisia ja ne ovat voineet muuttua. Tiedossa olleet laajennukset on esitelty seuraavassa luvussa.

4.1.1 Rajapintaan tehdyt laajennukset

Kaksoispainalluksen lukemiseen kosketusnäytöltä voidaan vaikuttaa asettamalla kaksoispainalluksen säde halutun kokoiseksi. Kaksoispainalluksen säteellä tarkoitetaan painallusten välistä maksimietäisyyttä, jotta painallukset tulkitaan kaksoispainallukseksi. Oletuksena säde on 20 kuvapistettä. Säteen kokoa muutetaan käyttämällä QApplication-luokan staattista metodia setDoubleclickRadius(int). Nykyinen arvo voidaan lukea käyttämällä QApplication-luokan staattista metodia doubleClickRadius().

Kaksoispainalluksen säteen asettamista on havainnollistettu esimerkissä 7.

Esimerkki 7: Kaksoispainalluksen säteen muuttaminen

```
QApplication::setDoubleClickRadius(30);
```

QTableEvents-luokan avulla voidaan lukea kosketusnäytöltä siihen kohdistuvan painalluksen voimakkuus. Voimakkuus luetaan ylikuormittamalla tabletin tapahtumia käsittelevä virtuaalinen tabletEvent-funktio. Oletuksena funktio ei käsittele tapahtumia ollenkaan. /6/

Painalluksen voimakkuuden lukemista on havainnollistettu esimerkissä 8.

Esimerkki 8: Painalluksen voimakkuuden lukeminen

```
/**  esittelytiedosto **/
…
protected:
   void tabletEvent(QTabletEvent *event);
…

/** toteutustiedosto **/

Class::tabletEvent(QTabletEvent *event);
{
    qreal pressure = event->pressure();
}
```

Käyttöliittymäkomponenttien lukeman syötteen muoto asetetaan automaattisesti Qt:n omille syötettä vastaanottaville käyttöliittymäkomponenteille, kuten tekstinsyöttökentille. Syötteen muotoa voidaan muuttaa käyttämällä QInputContext luokan metodia setInputMode(int mode).

Syötteen muodon muuttamista on havainnollistettu esimerkissä 9, jossa tekstirivin syöttämiseen tarkoitettuun käyttöliittymäkomponenttiin asetetaan käyttöön ennakoiva tekstinsyöttö.

Esimerkki 9: Käyttöliittymäkomponentin syötteen muodon muuttaminen

```
int mode = HILDON_GTK_INPUT_MODE_DICTIONARY
QInputContext con = lineEdit->inputContext();
con->setInputMode(mode);
```

Syötteen muoto annetaan enumeraatioiden avulla. Syötteen muodot ilmaisevat enumeraatiot on esitetty taulukossa 2. Taulukossa esitetyt enumeraatiot on lyhennetty poistamalla kaikissa syötteen muodon enumeraatioissa esiintyvä alkuosa HILDON_GTK_INPUT_MODE

Taulukko 2: Syötteen muodon enumeraatiot /11/

Enumeraatio	**Selitys**
_ALPHA	Aakkoset ja välilyönnit
_NUMERIC	Numerot ja väliviiva
_SPECIAL	Erikoismerkit
_HEXA	Numerot ja kirjaimet a-f sekä A-F
_TELE	Numerot ja merkit pwPW/().-+*#?,
_FULL	Syötteen muotoa ei ole rajoitettu.
_MULTILINE	Syöte jakaantuu useammalle riville.
_INVISIBLE	Syötettä ei näytetä. Käytetään esimerkiksi salasanakentissä.
_AUTOCAP	Muuttaa automaattisesti lauseen ensimmäisen merkin isoksi kirjaimeksi.
_DICTIONARY	Aktivoi ennakoivan tekstin syöttämisen.

Taulukossa 2 esitettyjä enumeraatioita voidaan yhdistää käyttämällä bittimuotoista tai-operaattoria |.

4.2 Alustan omat ohjelmointirajapinnat

Qt ei sisällä alustan omien rajapintojen käyttämisen mahdollistavia kirjastoja. Alustan omia palveluita käyttääkseen on kirjoitettava rajapintaluokat alustan omiin C-kielisiin ohjelmointirajapintoihin. Esimerkiksi GPS:n tai kameran käyttämiseksi Qt:sta käsin on kutsuttava alustan C-kielellä toteutettuja funktioita.

C-kielellä kirjoitettua ohjelmaa käyttävä C++-kielinen rajapintaluokka toteutetaan käyttämällä apuna varattua sanaa extern. Extern ”C” kertoo C++-kääntäjälle, että ohjelmakoodissa käytetään C-kielisiä funktioita ja että se on linkitettävä C-tyylisesti. Mikäli tätä ei kerrota kääntäjälle, ohjelma ei toimi. Tämä johtuu siitä, että C- ja C++-funktiokutsut linkitetään eri tavoin. C++-kääntäjä nimeää funktiot eri tavoin kuin C-kääntäjä, ja tämä johtaa käännetyn ohjelman linkityksen epäonnistumiseen. /3, 4/

Maemo-alustan sisältämät kirjastot on toteutettu niin, että niitä voidaan käyttää ilman extern-määreen lisäämistä C++-ohjelmakoodiin. Kirjastoissa määre on upotettu C++-ohjelmointikielen käytön ilmaisevan symbolisen vakion __CPLUSPLUS avulla ehtojen sisään siten, että määre otetaan käyttöön, kun kirjastoa käytetään C++-ohjelmakoodista. /17/

Esimerkissä 10 on esitetty Qt:lla toteutettu ohjelma, joka käyttää Hildon-kehyksen sisältämää funktiota. Symbolinen vakio Q_WS_HILDON on määritelty, kun ohjelma käännetään Maemo-alustan kehitysympäristössä. Vakiota voidaan käyttää erottamaan muusta lähdekoodista vain Maemo-alustalle käännettävät ohjelmakoodin osat.

Esimerkki 10: Hildon-kehyksen funktioiden käyttäminen Qt C++-ohjelmasta.

```
#include <QApplication>
#ifdef Q_WS_HILDON
#include <hildon/hildon-banner.h>
#endif

int main(int argc, char *argv[])
{
    QApplication app(argc,argv);
    hildon_banner_show_information(NULL, NULL,”hello”);
    return app.exec();
}
```

Edellä esitetyn lisäksi Qt:n projektitiedostoon on lisättävä ohjelman käännöksessä tarvittavat kirjastot. Esimerkissä 11 tarvittavat kirjastot on lisätty käyttämällä apuna pkg-config-työkalua. pkg-config-työkalun avulla voidaan ohjelman käännös- ja linkitysprosessissa tarvittavat kirjastot ja parametrit välittää kääntäjälle tarvittavan kirjaston nimen perusteella.

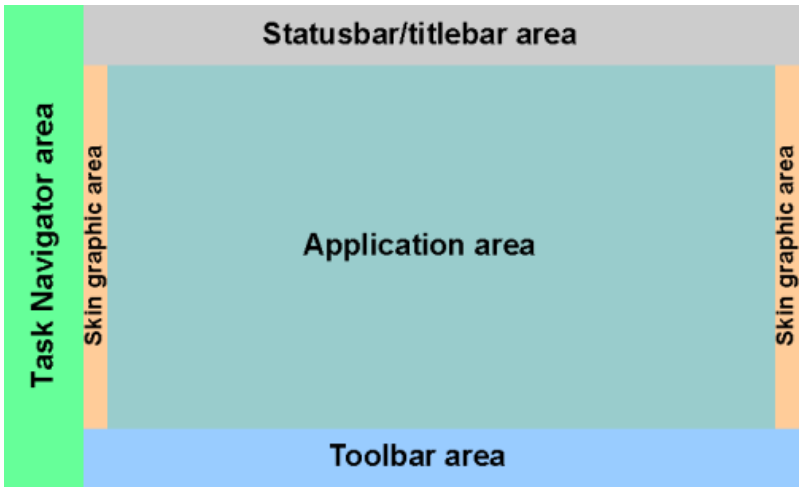
Esimerkki 11: Kirjastojen lisääminen projektitiedostossa pkg-config-työkalun avulla.

```
…
CONFIG += link_pkgconfig
PKGCONFIG += hildon-1
…
```

4.3 Käyttöliittymä

Maemo-alustalle kohdistettu Qt:n versio sisältää joitakin sovellusten ulkoasuun vaikuttavia muutoksia. Tehdyt muutokset saavat Maemo-alustalle käännetyt Qt-ohjelmat näyttämään ja tuntumaan natiiveilta Maemo-sovelluksilta.

Maemo-alustalla suoritettavien sovellusten käyttöliittymä poikkeaa jonkin verran perinteisistä PC-tietokoneella suoritettavista työpöytäsovelluksista. Esimerkiksi tilannepalkki on yhdistetty otsikkopalkkiin ja valikot ovat ruudun vasemmassa reunassa. Maemo-alustan käyttöliittymän tilankäyttöä päätelaitteen näytöllä on havainnollistettu kuviossa 7.



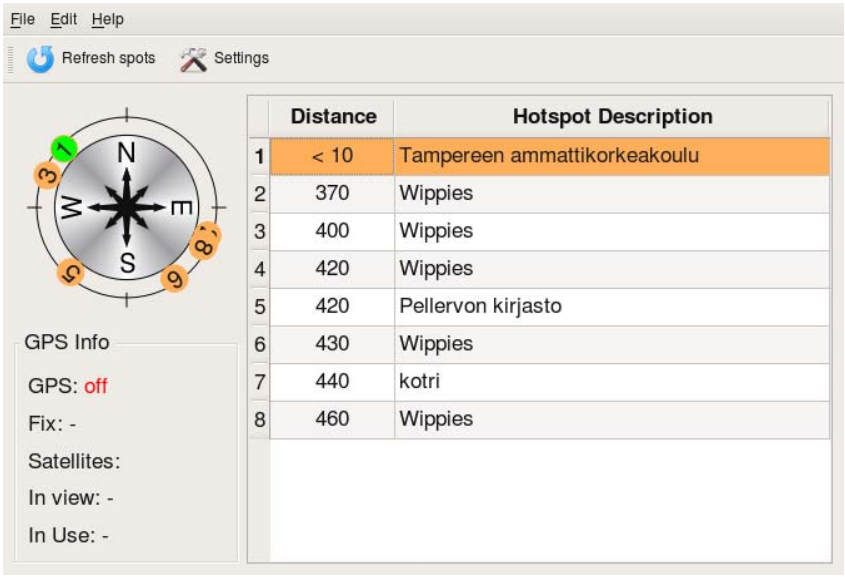
Kuvio 7: Maemo-käyttöliittymän tilankäyttö /15/

Koko näytön resoluutio on 800x480. Sovelluksen ollessa normaalitilassa, vasemman reunan valikot näkyvissä, sovellukselle varattu tila on 696x396 ilman työkalupalkkia ja 696x360 työkalupalkin kanssa. Kokonäyttötilassa sovellukselle varattu tila on 800x480 ilman työkalupalkkia ja 800x422 työkalupalkin kanssa. Muuttuva, sovellukselle varattava tila on syytä ottaa huomioon sovellusta suunniteltaessa. /15/

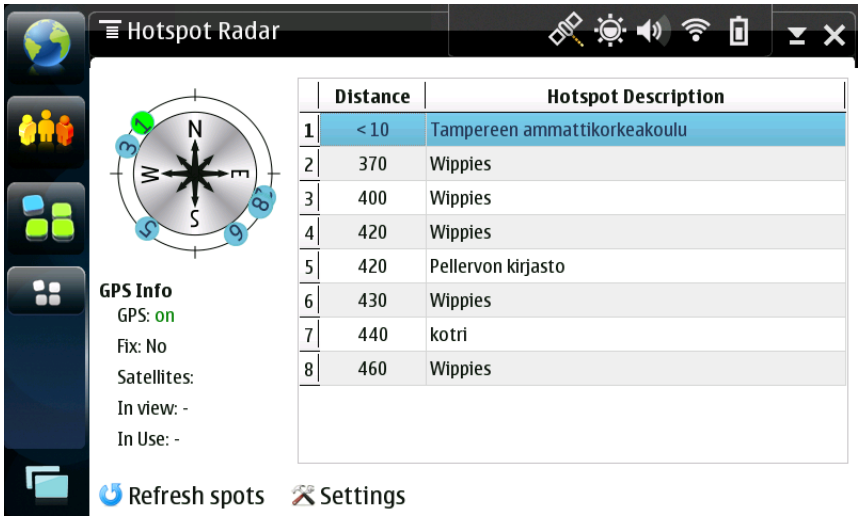
Kun Maemo-alustalle kehittävä sovellus rakennetaan QMainWindow-luokan ympärille, toimii Internet-tabletin kokonäytön valintanäppäin automaattisesti. Tämän lisäksi sovellus integroituu Maemo-alustan Hildon-kehykseen sekä näyttää ja tuntuu oikealta Maemo-sovellukselta.

4.3.1 Elementtien ulkoasu

Maemo-alustalla teksti- ja muiden käyttöliittymäelementtien ulkoasussa on huomattavia eroja normaaliin työpöytäympäristöön verrattuna, jotka on syytä ottaa huomioon käyttöliittymiä suunniteltaessa. Käyttöliittymien eroja on havainnollistettu kuvioissa 8 ja 9. Kuviossa 8 Qt-sovellus suoritetaan Ubuntu Linuxissa ja kuviossa 9 sama sovellus suoritetaan Maemo-alustalla.



Kuvio 8: Qt-sovelluksen suoritus Ubuntu Linuxissa



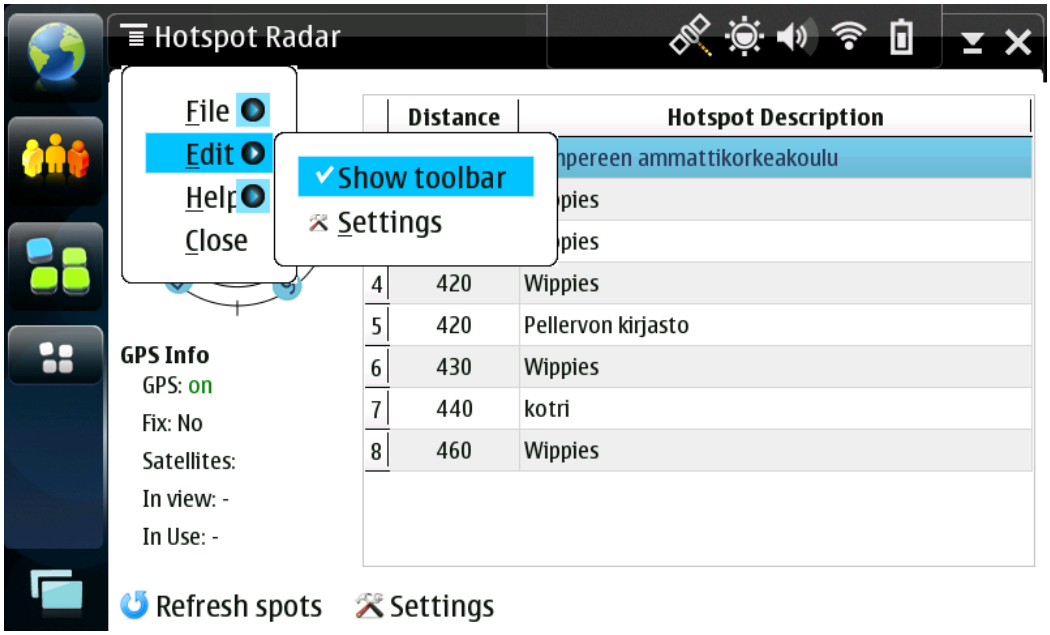
Kuvio 9: Qt-sovelluksen suoritus Maemo-alustalla

Kuvioissa näkyvä työkalupalkki on Maemo-alustalla aina kuvion 9 mukaisesti näytön alalaidassa. Muilla alustoilla työkalupalkin sijaintiin voidaan vaikuttaa ohjelmakoodissa

ja ajonaikaisesti raahaamalla palkki johonkin toiseen kohtaan tai sovellusikkunan ulkopuolelle. Työkalupalkkeja vertailemalla nähdään myös, että Maemo-alustalla tekstissä käytetään oletuksena huomattavasti suurempaa fonttikokoa. Muissa kuvioissa esiintyvissä tekstielementeissä käytetään erikseen määriteltyä fonttikokoa.

4.3.2 Valikot

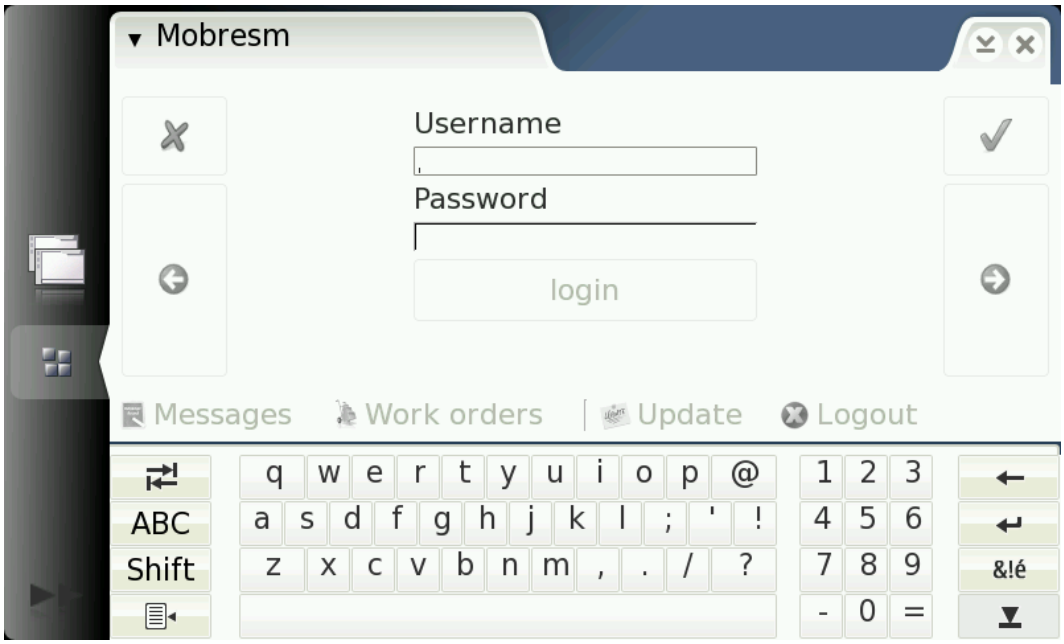
Maemo-alustalla sovelluksen valikot poikkeavat perinteisistä sovellusvalikoista. Valikot eivät näy suoraan sovelluksen yläpalkissa, vaan avautuvat yhden päävalikon alta. Valikkoja Maemo-alustalla on havainnollistettu kuviossa 10.



Kuvio 10: Sovellusvalikot Maemo-alustalla

4.3.3 Virtuaalinäppäimistö

Hildon-kehyksen sisältämä virtuaalinäppäimistö ponnahtaa oletuksena automaattisesti esiin käyttäjän valitessa syötettä vastaanottavan käyttöliittymäkomponentin. Kun virtuaalinäppäimistö on näkyvissä, suoritettavan sovelluksen käyttöliittymälle näytölle varattu tila pienenee. Virtuaalinäppäimistöä ja sen vaikutusta sovelluksen käyttöliittymään on havainnollistettu kuviossa 11.



Kuvio 11: Hildon-kehyksen virtuaalinäppäimistö

Virtuaalisen näppäimistön käyttöönotto syötekentän aktivoinnin yhteydessä voidaan estää komponenttikohtaisesti. Tekstinsyöttökentän yhteydessä virtuaalinäppäimistön aktivointi voidaan estää esimerkin 12 mukaisesti.

Esimerkki 12: Virtuaalinäppäimistön poistaminen käytöstä

```
QLineEdit *edit = new QLineEdit(this);
edit->setAttribute(Qt::WA_InputMethodEnabled, false);
```

4.3.4 Päätelaitteen erikoisnäppäimet

Päälaitteessa on joitain normaalista näppäimistöstä poikkeavia näppäimiä, kuten kokonäyttötilanäppäin. Toisaalta päätelaitteen näppäimistöstä puuttuu näppäimiä, joita normaali näppäimistö sisältää. Puuttuvia näppäimiä ovat mm. funktionäppäimet.

Kokonäyttö- ja sovelluksen pikavalikkonäppäimet on kytketty alustalla funktionäppäimiksi. F4-näppäin vastaa sovelluksen pikavalikkonäppäintä ja F6-näppäin vastaa kokonäyttönäppäintä.

Kuvan lähennys- ja loitonnusnäppäimen toiminta on määritelty QKeySequence-luokkaan. Lähennys on QKeySequence::ZoomIn ja loitonnus QKeySequence::ZoomOut.

Edellä esitetyt näppäimistön kytkennät ovat käytettävissä ainoastaan Qt:n pääikkuna-luokassa QMainWindow. /11/

4.4 Sovelluksen paketointi

Maemo-alustalle kehitetyt Qt-sovellukset paketoidaan käyttäen Debian projektin käyttämää Deb-paketointia, jota käytetään muidenkin alustalle kehitettyjen sovellusten paketointiin asentamiseen alustaa hyödyntäviin päätelaitteisiin. Deb-paketti sisältää sovelluksen suorittamiseen tarvittavat tiedostot sekä asennukseen tai muuhun toimintaan liittyvät komentosarjat.

Sovelluksen projektitiedostoon on määriteltävä sovelluksen asentamat tiedostot ja niiden sijainnit kohdejärjestelmässä /19/. Projektitiedostoon sovelluksen asennusta varten tehtäviä määrityksiä on havainnollistettu esimerkissä 13, jossa määritetään suoritettavan binääritiedoston ja sovelluksen alustan valikoihin sijoittavan työpöytätiedoston asennus.

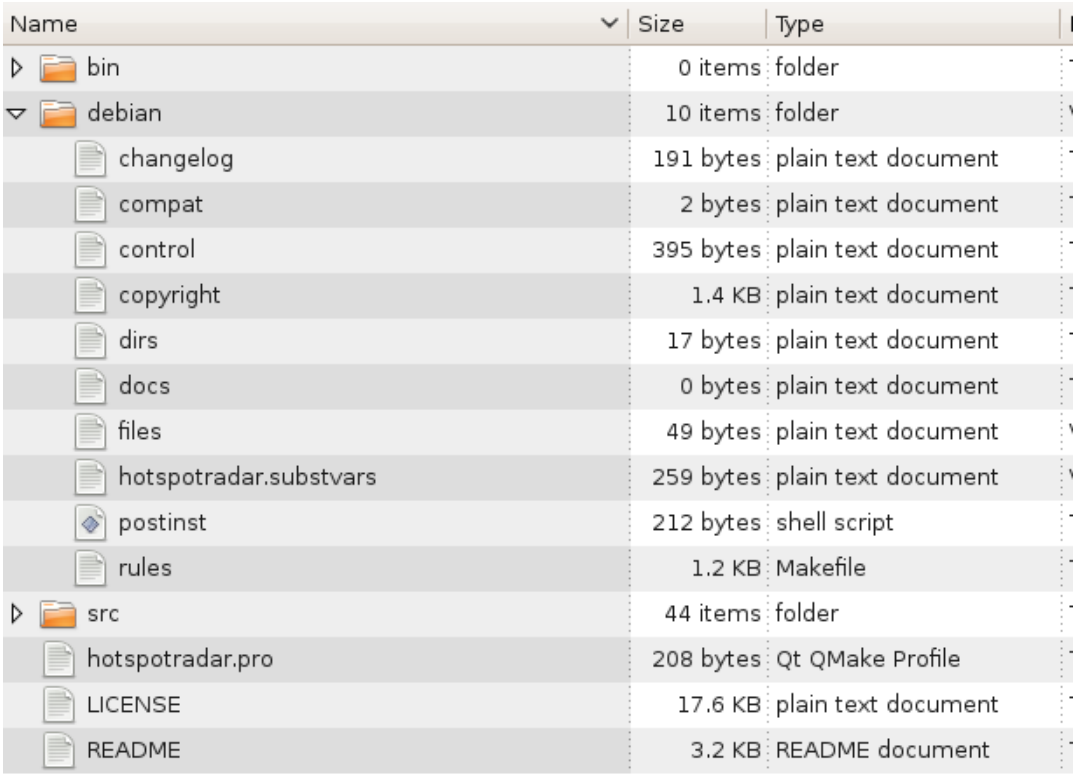
Esimerkki 13: Projektitiedostoon tehtävät asennusmääritykset

```
BINDIR = $$PREFIX/bin/
DATADIR = $$PREFIX/share/application/

desktop.path = $$DATADIR/hildon/
desktop.files = hotspotradar.desktop
INSTALLS += desktop

target.path = $$BINDIR
INSTALLS += target
```

Paketoitavalle sovellukselle luodaan hakemisto, jonka alle luodaan src-kansio. Src-kansioon sijoitetaan sovelluksen lähdekooditiedostot. Molempiin kansioihin kirjoitetaan oma Qt:n projektitiedosto. Ylemmän tason kansiossa olevassa projektitiedostossa ainoastaan kerrotaan alikansio, jossa lähdekoodit sijaitsevat. Src-kansiossa oleva projektitiedosto nimetään ”src.pro”, ja se sisältää kääntämisessä tarvittavat tiedot /11/. Paketoitavan sovelluksen sisältävän hakemiston rakennetta on havainnollistettu kuviossa 12.



Kuvio 12: Paketoitavan sovelluksen hakemistot

Kuviossa 12 näkyvä debian-kansio ja sen sisältämät tiedostot luodaan suorittamalla esimerkissä 14 esitetty komento.

Esimerkki 14: Ohjelman deb-paketoinnin valmisteleminen

```
dh_make -createorig –single –e joku@email.org –c gpl
```

Komento luo debian-kansion ja sen sisään paketoinnissa tarvittavien komentosarjojen rungot. Debian-kansiossa olevasta rules-tiedostosta poistetaan viittaukset configure-komennon käyttöön, sillä Qt ei käytä configure-komentoa sovelluksen kääntämisessä /11/. Qt-ohjelmalle kirjoitettua rules-tiedostoa on havainnollistettu esimerkissä 15.

Esimerkki 15: Deb-paketoitavan Qt-sovelluksen rules-tiedosto

```
#!/usr/bin/make -f
APPNAME := hotspotradar
builddir:
    mkdir -p builddir

builddir/Makefile: builddir
    cd builddir && qmake-qt4 PREFIX=/usr ../$(APPNAME).pro

build: build-stamp
```

```
build-stamp: builddir/Makefile
    dh_testdir
    cd builddir && $(MAKE)
    touch $@
clean:
    dh_testdir
    dh_testroot
    rm -f build-stamp
    rm -rf builddir
    dh_clean
install: build
    dh_testdir
    dh_testroot
    dh_clean -k
    dh_installdirs
    cd builddir && $(MAKE)\
    INSTALL_ROOT=$(CURDIR)/debian/$(APPNAME) install
binary-indep: build install
# We have nothing to do by default.
binary-arch: build install
    dh_testdir
    dh_testroot
    dh_installdocs
    dh_installexamples
    dh_installman
    dh_link
    dh_strip
    dh_compress
    dh_fixperms
    dh_installdeb
    dh_shlibdeps
    dh_gencontrol
    dh_md5sums
    dh_builddeb
binary: binary-indep binary-arch
.PHONY: build clean binary-indep binary-arch binary install con-
figure
```

Tämän jälkeen muokataan control-tiedostoa, johon määritellään sovelluksen tiedot ja riippuvuudet. Control-tiedoston sisältöä on havainnollistettu esimerkissä 16.

Esimerkki 16: Deb-paketoitavan sovelluksen control-tiedosto

```
Source: hotspotradar
Section: user/other
Priority: optional
Maintainer: timoph <email@address>
Build-Depends: debhelper (>= 5), libqt4-dev, libgpsbt-dev
Standards-Version: 3.7.2

Package: hotspotradar
Architecture: any
Depends: ${shlibs:Depends}
Description: Find wlan hotspot near your location
 Hotspot Radar shows Wippies and LangatonTampere
 hotspots nears your current location.
```

Mikäli sovellus halutaan näkyviin päätelaitteen sovellusvalikkoon, täytyy sovellukselle kirjoittaa desktop-tiedosto, joka sisältää tarvittavat tiedot sovelluksen näyttämiseksi sovellusvalikossa. Desktop-tiedoston sisältöä on havainnollistettu esimerkissä 17.

Esimerkki 17: Sovelluksen desktop-tiedosto

```
[Desktop Entry]
Encoding=UTF-8
Version=1.0
Type=Application
Name=Hotspot Radar
Exec=/usr/bin/hotspotradar
X-Osso-Type=application/x-executable
Terminal=false
```

Sovelluksen käyttäjän voidaan antaa valita minkä valikon pääotsikon alle sovellus sijoitetaan asennuksen yhteydessä. Valinnan tekeminen mahdollistetaan kirjoittamaan debian-kansioon asennuksen päätteeksi suoritettava postinst-tiedosto, jossa kutsutaan Maemo-alustan sisältämää komentoa sovelluksen sijainnin valitsemiseen. Postinst-tiedoston sisältöä on havainnollistettu esimerkissä 18.

Esimerkki 18: Valikkosijainnin valitsemisen kutsuminen postinst-tiedostosta

```
#!/bin/sh
maemo-select-menu-location hotspotradar.desktop tana_fi_extras
```

Kun Debian-paketoinnissa tarvittavat komentosarjat on muokattu paketoitavan sovelluksen vaatimaan muotoon, paketoidaan sovellus esimerkissä 19 esitetyllä komennolla.

Esimerkki 19: Sovelluksen paketoiminen

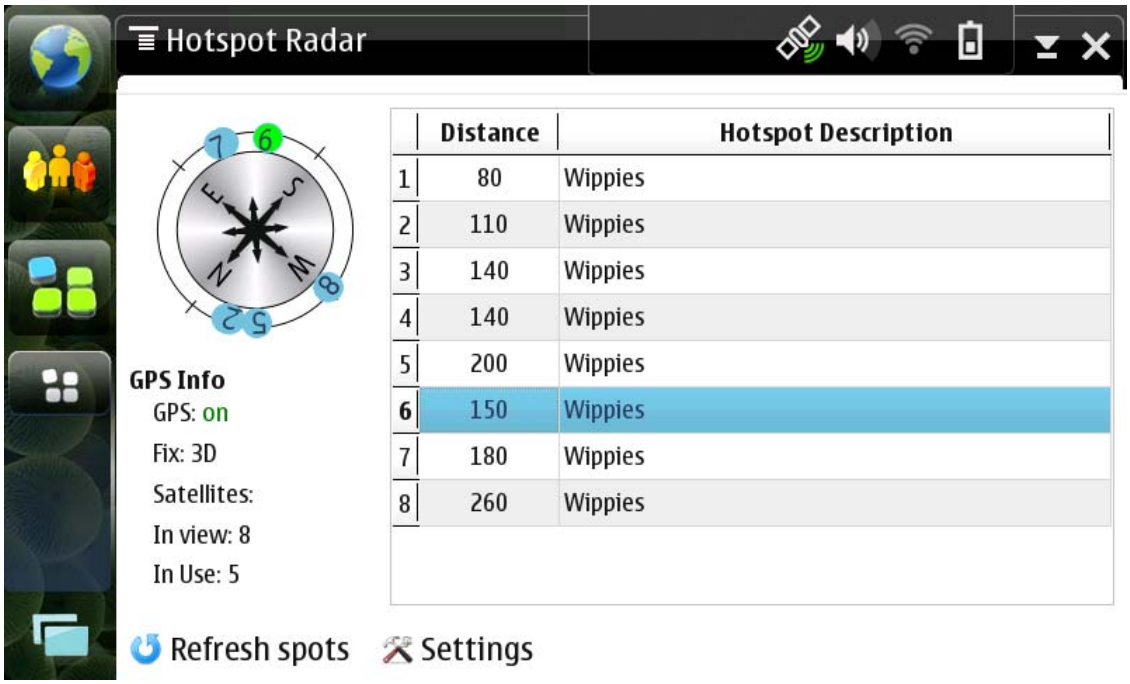
```
dpkg-buildpackage –rfakeroot -b
```

Komento luo Deb-paketin, jota käyttäen sovellus voidaan asentaa Maemo-alustaa käyttäviin päätelaitteisiin. /11/

5 Toteutetun sovelluksen esittely

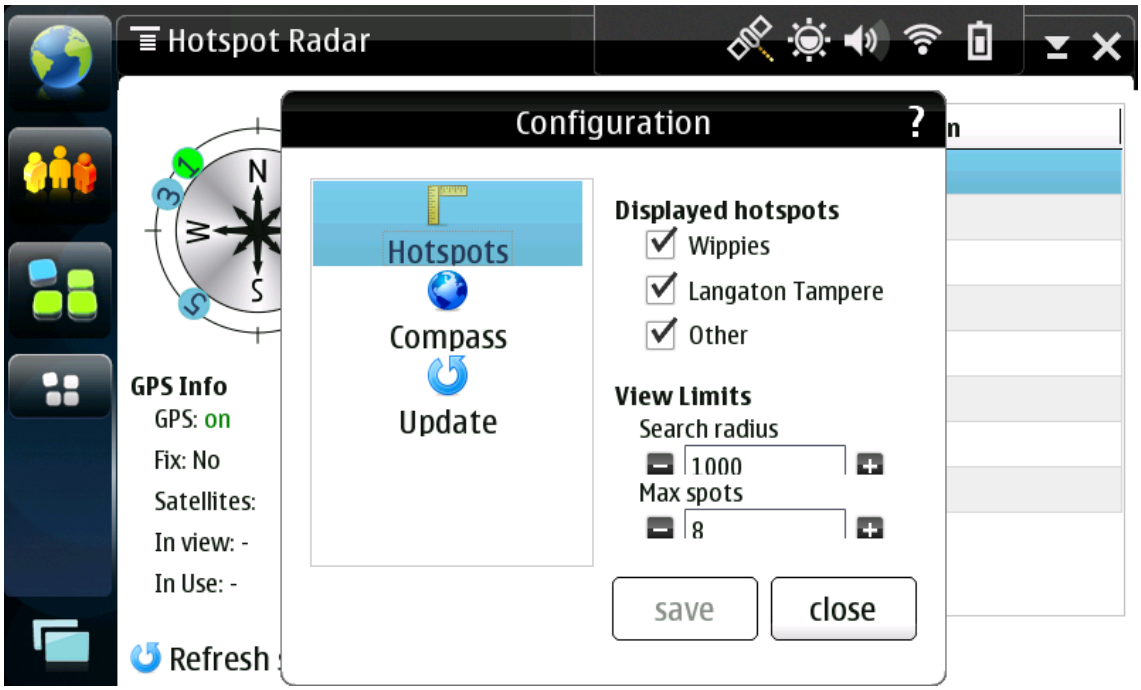
Tämän työn taustatyöksi toteutettiin Hotspot Radar -sovellus. Sovellus on tarkoitettu auttamaa Internet-tabletin käyttäjää löytämään hänen nykyistä sijaintiaan lähinnä olevat langattomat tukiasemat.

Sovelluksen päänäkymä sisältää listan lähimmistä tukiasemista, GPS-yhteyden tiedot sekä kompassina toimivan tukiasemien suunnan näyttävän käyttöliittymäelementin. Sovelluksen päänäkymä on havainnollistettu kuviossa 13.

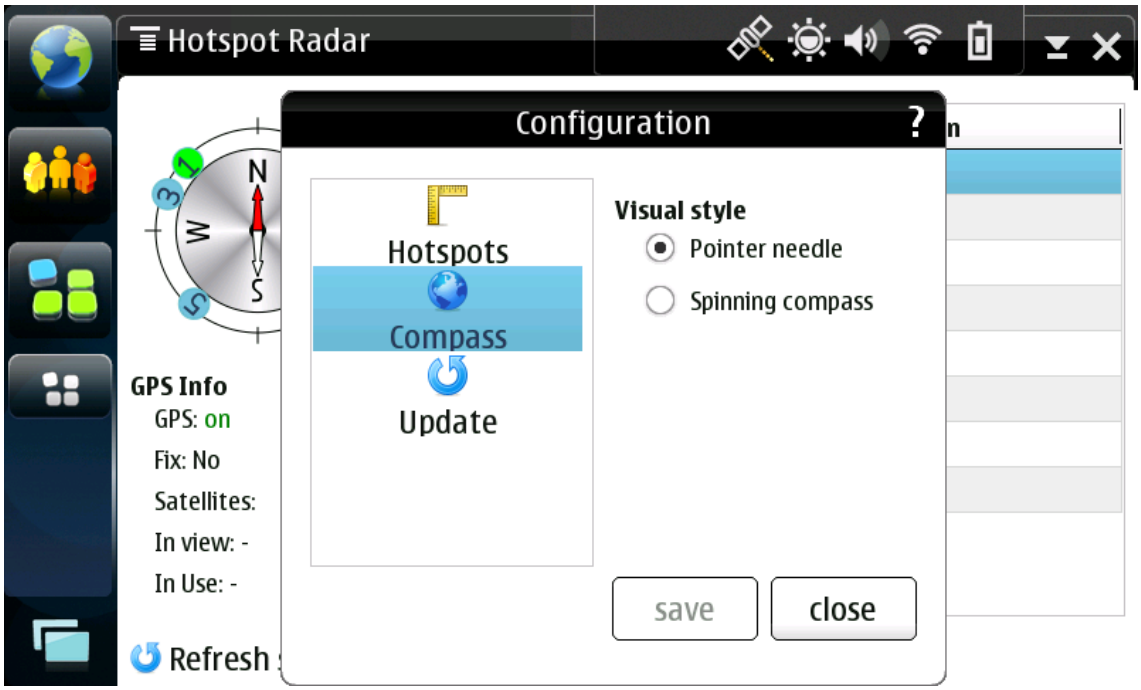


Kuvio 13: Hotspot Radar -sovelluksen päänäkymä

Sovelluksen asetuksista voidaan vaikuttaa näytettäviin tukiasemaryhmiin sekä haettavien tukiasemien maksimi määrään ja etäisyyteen. Lisäksi asetusdialogista voidaan vaikuttaa kompassina toimivan käyttöliittymäelementin ulkoasuun sekä päivittää sovelluksen tiedot langattomien tukiasemien sijainnista. Sovelluksen asetusdialogia on havainnollistettu kuvioissa 14 ja 15.



Kuvio 14: Hotspot Radar -sovelluksen asetusdialogi



Kuvio 15: Hotspot Radar -sovelluksen kompassin asetukset

Esitelty sovellus ja sen lähdekoodit ovat julkisesti saatavilla GNU GPL versio 2 lisensoituna www-osoitteesta http://hotspotradar.garage.maemo.org.

6 Yhteenveto

Maemo on avoin ohjelmistokehitysalusta mobiililaitteille, mikä tarjoaa hyvän mahdollisuuden kehittää omia ohjelmia tai itse alustaa haluamallaan tavalla. Alusta tukee ohjelmistokehitystä eri ohjelmointikielillä.

Nokia Oyj:n ohella Maemo-alustaa ja sille tarkoitettuja sovelluksia kehittää aktiivinen kehittäjä- ja käyttäjäyhteisö. Yhteisö ylläpitää alustan dokumentaatiota ja oppaita, tarjoaa sovelluskehittäjälle työkaluja projektien hallintaan, julkaisuun sekä kanavat kehittäjien väliseen kommunikaatioon. Työn aiheesta kiinnostuneille yhteisö on hyvä paikka tutustua alustaan ja sille tehtävään kehitystyöhän tarkemmin.

Qt laajentaa C++:n luokkakäsitettä esimerkiksi ominaisuusjärjestelmällä sekä signaaleilla ja lokeroilla, joita käytetään olioiden välisessä kommunikaatiossa. C++ -kieleen tehdyt laajennukset on toteutettu käyttämällä meta-luokkia. Kehittäjän kirjoittamasta lähdekoodista luodaan standardin C++-kääntäjän ymmärtämiä meta-luokkia Qt:n qmake- ja MOC -työkaluja käyttämällä.

Qt:n tarjoamat kirjastot ja laajennukset C++-ohjelmointikieleen ovat nopeita omaksua ja tehokkaita käyttää. Tekniikka soveltuu hyvin graafisten käyttöliittymien kehittämiseen alustasta riippumatta. Kehitettäessä sovellusta jollekin tietylle alustalle, kuten Maemo-alustalle, on otettava huomioon alustan erityispiirteet.

Nopeaa vauhtia yleistyvät kosketusnäytölliset mobiililaitteet asettavat käyttöliittymien suunnittelulle uusia haasteita ja mahdollisuuden uusien innovatiivisten käyttöliittymien kehittämiseen. Maemo-alusta antaa kaikille kehittäjille mahdollisuuden toteuttaa visioitaan kosketusnäytöllisen päätelaitteen käyttöliittymien suhteen.

Lähdeluettelo

Kirjalliset lähteet

/1/ Thelin, Johan 2007. Foundations of Qt Development. New York: Apress.

/2/ Blanchette, Jastim & Summerfield, Mark 2008. C++ GUI Programming with Qt4. Westford: Prentice Hall.

/3/ Stroustrup, Bjarne 2007. The C++ Programming Language Third Edition. Westford: Addison Wesley.

/4/ Von Hagen, William 2006. The Definite Guide to GCC. New York: Apress.

Sähköiset lähteet

/5/ Maemo Diablo Reference Manual for maemo 4.1. [www-sivu] [viitattu 5.2.2009]. http://maemo.org/maemo_release_documentation/maemo4.1.x/

/6/ Qt Reference Documentation. [www-sivu] [viitattu 4.2.2009]. http://doc.trolltech.com/4.4/index.html

/7/ Open Source Definition. [www-sivu] [viitattu 29.1.2009]. http://opensource.org/docs/osd/

/8/ Home of the Maemo Community. [www-sivu] [viitattu 29.1.2009]. http://www.maemo.org

/9/ Qt4 Hildon project. [www-sivu] [viitattu 3.2.2009]. http://qt4.garage.maemo.org

/10/ Forum Nokia. [www-sivu] [viitattu 28.1.2009]. http://forum.nokia.com

/11/ Maemo Community Wiki. [www-sivu] [viitattu 7.2.2009]. http://wiki.maemo.org

/12/ Maemo Quick Start Guide. [www-sivu] [viitattu 7.2.2009]. http://maemo.org/development/documentation/maemo-quick-start-guide.pdf

/13/ GLib Reference Guide. [www-sivu] [viitattu 8.2.2009]. http://library.gnome.org/devel/glib/

/14/ The GTK+ Project. [www-sivu] [viitattu 13.2.2009]. http://www.gtk.org

/15/ Maemo Technology Overview. [www-sivu] [viitattu 16.2.2008]. http://maemo.org/maemo_training_material/maemo4.x/html/maemo_Technology_Overview/index.html

/16/ GNOME Live. [www-sivu] [viitattu 26.2.2008]. http://live.gnome.org/Hildon

/17/ Maemo API Reference. [www-sivu] [viitattu 24.3.2009]. http://maemo.org/api_refs/

/18/ QtSoftware. [www-sivu] [viitattu 24.3.2009]. http://www.qtsoftware.com

/19/ Debian GNU/Linux FAQ. [www-sivu] [viitattu 24.3.2009]. http://www.debian.org/doc/FAQ/ch-pkg_basics

Liitteet

Avoimen lähdekoodin määritelmä

Avoimella lähdekoodilla tarkoitetaan ohjelmia tai kirjastoja, jotka täyttävät avoimen lähdekoodin määritelmän asettamat vaatimukset. Määritelmän asettamat vaatimukset ovat

1. Ohjelman tulee olla vapaasti levitettävissä. Ohjelman levitystä ei saa rajoittaa lisenssiehdoilla.
2. Ohjelman lähdekoodi on toimitettava ohjelman mukana ja myös sen on oltava vapaasti levitettävissä.
3. Ohjelman lisenssiehtojen on sallittava ohjelman muokkaaminen ja muokatun ohjelman levittäminen samoin ehdoin kuin alkuperäisen ohjelman.
4. Lisenssiehdoissa voidaan rajoittaa lähdekoodin levittämistä, mikäli ne sallivat korjaustiedostojen toimittamisen lähdekoodin mukana. Lisenssiehdoissa voidaan kieltää muokattujen ohjelmaversioiden levittäminen alkuperäisen ohjelman versionumerolla.
5. Lisenssiehdot eivät saa syrjiä yksilöitä tai ihmisryhmiä.
6. Ohjelman käyttöä ei saa rajoittaa millekään tietylle alalle.
7. Lisenssiehdot ovat kaikille ohjelman käyttäjille samat.
8. Lisenssiehtojen on koskettava kaikkia ohjelman osia. Oikeuksien on säilyttävä, vaikka osa ohjelmasta irrotettaisiin alkuperäisestä kokonaisuudesta.
9. Lisenssiehdot eivät saa rajoittaa muiden ohjelmien käyttöä. Ohjelmaa on voitava levittää yhdessä sellaisien ohjelmien kanssa, jotka eivät käytä samaa lisenssiä.
10. Lisenssin on oltava neutraali käytettävän teknologian suhteen.