TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tutkintotyö

Henri Auer

OGRE 3D-grafiikkamoottori Qt-ympäristössä

Työn ohjaaja lehtori Jari Mikkolainen
Työn teettäjä TAMK
Tampere 12/2009

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Tekijä Auer, Henri
Työn nimi OGRE 3D-grafiikkamoottori Qt-ympäristössä
Sivumäärä 37 sivua + 31 liitesivua
Työn ohjaaja Jari Mikkolainen
Työn teettäjä TAMK
Joulukuu 2009
Hakusanat OGRE, Qt, grafiikkamoottori, Maemo

TIIVISTELMÄ

Tässä työssä tutkitaan OGRE 3D -grafiikkamoottorin hyödyntämistä Qt-kehitysympäristössä. Kehitysympäristöjen asennus ja käyttö sekä työn ohella tehty esimerkkiprojekti toteutetaan Linux-ympäristössä. Esimerkkiprojekti toteutetaan sekä pöytätietokoneille että Maemo-pohjaisille kannettavilla laitteille. Työssä käydään läpi myös OGRE:n ja Qt:n yleistietoa ja niiden rakennetta sekä esitetään esimerkkiprojektin lähdekoodi ja muut tarpeelliset tiedostot, joita projektissa tarvitaan.

Työssä keskitytään vain niihin asioihin, jotka liittyvät OGRE- ja Qt-kehitysympäristöjen asentamiseen ja käyttämiseen sekä siihen, kuinka OGRE:n 3D-ominaisuudet saadaan käyttöön Qt:lla toteutetussa projektissa.

Lukijalta, joka haluaa selvittää, kuinka esimerkkiprojekti on toteutettu, edellytetään Qt C++:n ja OGRE:n perusymmärrystä. Itse esimerkkisovelluksen lähdekoodia käydään läpi vain pintapuolisesti niiltä osin, jotka ovat 3D-grafiikkamoottorin integroinnin kannalta tärkeitä.

TAMK UNIVERSITY OF APPLIED SCIENCES
Programme in Information Technology
Software engineering
Author Auer, Henri
Name of thesis OGRE 3D-graphics engine in Qt environment
Page count 37 pages + 31 appendices
Thesis supervisor Jari Mikkolainen
Commissioning Company TAMK
December 2009
Keywords OGRE, Qt, graphics engine, Maemo

ABSTRACT

In this thesis is examined how the OGRE 3D-graphics engine could work in a Qt-application. There is discovered the installation and usage of software development kits and an example project which is made with them. The example project is designed to work on both Linux operation system and Maemo based mobile devices. Thesis also covers the basic introduction of the Qt and the OGRE and it shows the source code of the example project and necessary files which are needed by the project.

This thesis includes only installation and usage of OGRE and Qt software development kits and explains how the OGRE 3D graphics engine is used in Qt projects.

To understand how the example project is designed the reader must have basic knowledge in Qt C++ and OGRE. The source code of example project is explained only on those parts which are needed by integration of 3D-graphics engine.

ALKUSANAT

Tämän työn aihe tuli esiin kesällä 2009, kun aloin miettiä kuinka kolmiulotteisia peli- tai simulaatioympäristöjä pystyisi toteuttamaan Qt-sovelluksissa. Qt-kehitysympäristö tukee täysin 3D-grafiikkaa, mutta laajempien ja monimutkaisempien 3D-sovellusten toteutus ilman toimivaa 3D-grafiikkamoottoria olisi liian suuritöinen projekti.

Tutkittuani valmiita 3D-moottoreita, jotka voisivat toimia yhdessä Qt-kehitysympäristön kanssa, kävi ilmi, että Qt:n ja OGRE 3D -grafiikkamoottorin yhteistyöstä löytyy aiemmin toteutettuja projekteja. Koska OGRE 3D -grafiikkamoottori toimii myös iPhone:ssa ja Linuxissa, päätin kokeilla Qt:n ja OGRE:n yhteistyön toimivuutta myös Nokian kehittämässä Maemo-Linuxissa.

Ajankohtaisesti tämä aihe on erinomainen, koska työn valmistuessa Nokia Oyj julkaisee samana vuonna Maemo-pohjaisen N900-puhelimen, jossa tämän työn tuloksien pitäisi toimia. Samoin OGRE:n 3D-tuki OpenGL ES:lle, jota käytetään sulautetuissa laitteissa, toteutettiin aikaisemmin samana vuonna. Tämä opinnäytetyö on tehty elokuun 2009 ja joulukuun 2009 välisenä aikana.

Haluan kiittää ohjaajaani Jari Mikkolaista opinnäytetyöni kehittävästä palautteesta ja ohjauksesta. Lisäksi haluan kiittää Marja-Liisa ja Jukka Mannista työn kirjallisen osuuden kehittämisestä. Lopuksi haluan vielä kiittää OGRE- ja Maemo-yhteisöjen avusta sovelluksen teknisessä toteutuksessa.

Tampereella 17.12.2009

Henri Auer

SISÄLLYSLUETTELO

LYHENTEET JA MERKIT

GPL	*GNU General Public License. GNU yleinen lisenssi.*
LGPL	*GNU Lesser General Public License. GNU vähemmän yleinen lisenssi.*
OUL	*OGRE Unrestricted License. OGRE rajoittamaton lisenssi.*
X11	*X Window System Version 11. X ikkunointijärjestelmä versio 11.*
OGRE	*Object-Oriented Graphics Rendering Engine. Oliopohjainen moottori grafiikan toistamiseen.*
SDK	*Software development kit. Ohjelmistokehitystyökalu.*
Maemo	*Debian-based Linux developed by Nokia Oyj. Nokia Oyj:n kehittämä Debian-pohjainen Linux-käyttöjärjestelmä.*
OpenGL ES	*OpenGL for Embedded Systems. OpenGL sulautettuihin järjestelmiin.*
Arm	*32-bit architecture designed for embedded systems. 32-bittinen arkkitehtuuri suunniteltu sulautetuille järjestelmille.*
Armel	*An ARM architecture emulator. ARM-arkkitehtuurin emulaattori.*
Scratchbox	*Scratchbox is a cross compilation toolkit for embedded Linux application development. Scratchbox on ristikäännös-työkalu, joka helpottaa Linuxin sulautettujen järjestelmien sovelluskehitystä.*
Xephyr	*X server which targets a window on a host X Server as its framebuffer. X-palvelin, joka toimii omassa ikkunassaan jo käynnistetyn X-palvelimen sisällä.*

1 JOHDANTO

Tämän työn aiheena on käsitellä OGRE 3D -grafiikkamoottorin toimintaa Qt-kehitysympäristössä. Työ on jaettu kahteen osaan, joista ensimmäinen keskittyy kehitysympäristön toteutukseen perinteisessä Linux-ympäristössä ja jälkimmäinen osa keskittyy siihen, kuinka kehitysympäristö saadaan toimimaan, kun sovelluksia kehitetään Nokia Oyj:n kehittämille Maemo-pohjaisille laitteille.

Luvussa 2 käydään läpi teoriaa ja taustatietoa Qt-kehitysympäristöstä; minkälaisista osista kehitysympäristö koostuu ja mille laitteille ja käyttöjärjestelmille se on suunnattu. Qt:n avulla on tarkoitus toteuttaa tämän työn ohella tehtävä graafinen sovellus, joka toimii sekä pöytäkoneilla että kannettavilla Maemo 5-käyttöjärjestelmään pohjautuvilla laitteilla.

Luvussa 3 selvitetään OGRE 3D -grafiikkamoottorin taustoja, rakennetta sekä sitä, minkälaisia asioita sen avulla voidaan toteuttaa. OGRE 3D -grafiikkamoottorin avulla toteutetaan ohella tehtävän sovelluksen 3D-grafiikan piirtäminen.

Luvussa 4 esitetään, miten Ubuntu 9.10 -käyttöjärjestelmään asennetaan kehitysympäristö, joka suoriutuu ohella tehtävän sovelluksen toteutuksesta. Luvussa kerrotaan Qt- ja OGRE-kehitysympäristöjen sekä tarvittavien työkalujen asennus.

Luvussa 5 selvitetään sisältö ja kehittäminen esimerkkiprojektista, jonka tarkoituksena on pystyä toteuttamaan graafinen Qt-kirjastoilla toteutettu sovellus, jossa käytetään OGRE 3D-grafiikkamoottoria. Samoin selvitetään projektin tarvitsemat asetukset ja tiedostot sekä se, kuinka Qt:n ja OGRE:n yhteistyö toimii lähdekooditasolla.

Luvuissa 6 ja 7 käydään läpi samat asiat kuin luvuissa 4 ja 5, mutta Maemo-kehityksen kannalta. Luvussa 6 kerrotaan projektin vaatiman kehitysympäristön asentaminen Ubuntu-käyttöjärjestelmään. Luvussa 7 selvitetään sovelluksen kehitys Maemo-laitetta varten.

2 QT-OHJELMISTOKEHITYSYMPÄRISTÖ

Tässä luvussa esitellään Qt:n taustaa ja rakennetta yleisellä tasolla. Selvitetään mihin Qt:ta voidaan käyttää ja minkälaisissa kehitysympäristöissä sen avulla voidaan kehittää sovelluksia.

2.1 Yleistietoa Qt:sta

Qt on useissa käyttöjärjestelmissä toimiva ohjelmistokehitysympäristö, joka on erityisesti suunnattu graafisten käyttöliittymien kehitykseen. Yleisnäkemys sen kokonaisuudesta esitetään kuvassa 1. Siihen kuuluu useissa käyttöjärjestelmissä toimivat kirjastot, sisäänrakennetut kehitystyökalut ja editori, joka tunnetaan nimellä QtCreator. (Qt – A cross platform application and UI framework 2009.)

Kuvasta 1 käy ilmi, että Qt:n avulla voidaan toteuttaa sovelluksia Windows-, Mac OS X- ja Linux-käyttöjärjestelmille sekä sulautetuille laitteille, jotka käyttävät Linux-, WinCE- tai Symbian S60-käyttöjärjestelmiä. Kuvassa esiintyvät kirjastomoduulit on selitetty luvussa 2.2.

Kuva 1 Qt-ohjelmistokehitystyökalun kuvaus (Qt – A cross platform application and UI framework 2009.)

Qt:n kehittäjä on Nokia Oyj:n perustama Qt Software, joka alun perin oli nimeltään Trolltech. Muutos yrityksen nimeen tehtiin vuonna 2008, jolloin Nokia Oyj osti Trolltechin itselleen. Qt-ohjelmistokehitysympäristö toimii tällä hetkellä ainakin X11-, Max OS X-, Windows-käyttöjärjestelmillä ja joillakin sulautetuilla käyttöjärjestelmillä kuten Symbianilla ja Maemolla. (Qt – Wikipedia 2009.)

Qt:n avulla suunnittelija voi saada toimimaan yhden ratkaisunsa eri käyttöjärjestelmillä ja laitteilla ilman lähdekoodin uudelleen kirjoitusta. Se tukee C++- ja Java-ohjelmointikieliä, mutta epävirallisesti myös Python- ja Ruby-kieliä. (Qt – A cross platform application and UI framework 2009.)

Qt:n avulla sovellusten kehittäminen Nokian matkapuhelimiin on helpompaa ja halvempaa kuin natiivin Symbian C++:n avulla. Matkapuhelimien erikoisominaisuuksia kuten kameraa ja tekstiviestejä varten voidaan käyttää Qt Mobility-pakettia. (Qt auttaa jo Symbianiin tuskastuneita 2009.)

Tällä hetkellä uusimman Qt 4.6 -version avulla voidaan käyttää hyödyksi uusimpia laitteiden ominaisuuksia kuten, monikosketusta. Uusi Qt-versio on erityisesti optimoitu Maemo 6 -käyttöjärjestelmää varten, joka aiotaan julkaista vuoden 2010 kuluessa. Tämä versio tukee myös uusia käyttöjärjestelmiä, joille ei aikaisemmin ollut tukea. Näitä käyttöjärjestelmiä ovat Windows 7, Mac OS X 10.6, Solaris 10, HP-UX ja AIX 6. (Qt 4.6 Technology Preview 2009.)

2.2 Qt:n kirjastomoduulit

Qt-kirjasto koostuu useista eri moduuleista. Näitä moduuleita ovat:

- QtCore, joka sisältää ei-graafiset luokat, joita muut moduulit käyttävät toimiakseen.
- QtGui, joka koostuu graafisista käyttöliittymäkomponenteista.
- QtMultimedia, joka auttaa käyttämään matalan tason multimediaominaisuuksia.

- QtNetwork, joka sisältää verkkoliikenteeseen liittyvät komponentit.
- QtOpenGL, joka mahdollistaa 3D-grafiikan piirtämisen.
- QtOpenVG, joka mahdollistaa vektorigrafiikan käytön.
- QtScript, joka mahdollistaa skriptien käytön.
- QtScriptTools, joka sisältää lisättyjä skriptiominaisuuksia.
- QtSql, joka mahdollistaa tietokantojen käytön.
- QtSvg, joka auttaa piirtämään svg-grafiikkaa.
- QtWebKit, joka mahdollistaa web-teknologian.
- QtXml, joka auttaa xml-tiedostojen käsittelyä.
- QtXmlPatterns, joka on XQuery- ja Xpath-moottori xml-tekniikkaa varten.
- Phonon, joka mahdollistaa äänentoiston.
- Qt3Support, joka auttaa hyödyntämään vanhempia lähdekoodeja. (All Qt Modules 2009.)

Qt:sta on saatavilla useita eri lisenssejä. Niitä ovat kaupallinen-, LGPL-, ja GPL-lisenssi. (Qt Licensing 2009.)

3 OGRE 3D -GRAFIIKKAMOOTTORI

Tässä luvussa esitellään OGRE 3D -grafiikkamoottorin taustaa, arkkitehtuuria ja ominaisuuksia yleisellä tasolla.

3.1 Yleistietoa OGRE:sta

OGRE on skene-pohjainen ja joustava 3D-grafiikkamoottori. Skenellä, johon moottori pohjautuu, tarkoitetaan kokonaisuutta, joka voi sisältää useita graafisia ja hierarkkisia komponentteja. Nämä komponentit toistetaan näytölle skenen toistamiseen erikoistuneilla luokilla. Skeneen voidaan esimerkiksi liittää valoja, 3D-hahmoja ja maastoja, joiden sijainti- ja muut parametrit määritellään skenen tiedostoihin tai lähdekoodiin, jossa skene luodaan. OGRE on suunniteltu helpottamaan sovellusten kehittäjiä luomaan laitteistokiihdytettyjä 3D-sovelluksia. (OGRE – Wikipedia 2009.)

OGRE:lla on erittäin aktiivinen yrityksistä riippumaton yhteisö ja se valittiin kuukauden projektiksi Sourceforge.net-sivulla maaliskuussa 2005. OGRE 3D -grafiikkamoottoria on käytetty sellaisenaan joissakin kaupallisissa peleissä. Ensimmäinen OGRE:n versio, nimeltään Azathoth, julkaistiin helmikuussa 2005, ja nykyinen virallinen versio 1.6.4, nimeltään Shoggoth, julkaistiin syksyllä 2009. Tässä työssä käytetään kuitenkin OGRE:n uusinta epävirallista versiota 1.7, jota kutsutaan nimellä Cthugha. Tämä johtuu siitä, että OpenGL ES -tuki on toteutettu vasta uusimmassa kehitysversiossa. OpenGL ES -tukea tarvitaan Maemon kannettaviin laitteisiin. (OGRE – Wikipedia 2009.)

Tällä hetkellä OGRE on julkaistu kaksoislisenssin alla. Nämä lisenssit ovat LGPL ja OUL. OUL-lisenssi on maksullinen, mutta sen käyttäjän ei tarvitse ilmoittaa tekemistään muutoksista OGRE:n lähdekoodiin. Näin ollen projektiin kehitetyt muutokset pysyvät vain kehittäjän omana tietona. (OGRE – Wikipedia 2009.)

OGRE on periaatteessa vain grafiikkamoottori, kuten sen nimikin kertoo. Sen päätarkoituksena on antaa laadukas ja monipuolinen ratkaisu grafiikantoistoa

varten. Se ei siis ole kokonaisratkaisu pelien kehitykseen tai simulaatioihin, koska se ei esimerkiksi sisällä lainkaan äänentoistoa tai fysiikan laskentaa. Näitä ominaisuuksia saadaan kuitenkin käyttöön liittämällä lisäkirjastoja projektiin. OGRE-yhteisö on tehnyt lisäkirjastoja esimerkiksi törmäysten tunnistamiseen ja fysiikan laskentaan. (OGRE – Open Source 3D Graphics Engine 2009.)

OGRE on suunniteltu olio-pohjaiseksi 3D-grafiikkamoottoriksi liitännäisteknologian kanssa. Liitännäistekniikka auttaa lisäämään siihen uusia ominaisuuksia, kuten fysiikan laskentaa ja tekee siitä erittäin modulaarisen. Tämän avulla sovelluksen kehityksessä on helppoa ottaa käyttöön uusia ominaisuuksia vain lisäämällä projektin asetuksiin liitännäisen nimen. OGRE:a on myös kehuttu siitä, että se on joustava, täysin dokumentoitu ja parhaiten suunniteltu ilmaisista 3D-moottoreista. (OGRE – Open Source 3D Graphics Engine 2009.)

OGRE toimii useissa eri käyttöjärjestelmissä, jotka tukevat sekä OpenGL:ää että Direct3D:ta. Se voi toistaa samaa graafista sisältöä eri käyttöjärjestelmillä ja laitteilla, ilman että sisällön tekijän täytyy huomioida niiden erityisominaisuuksia. Tämä vähentää ongelmallisuutta sovellusten julkaisussa eri käyttöjärjestelmille. (OGRE – Wikipedia 2009.)

OGRE:n kirjastot sisältävät muistin virheenjäljitys- ja ulkoisten lähteiden lataamisominaisuuksia. 3D-grafiikkaan liittyviä työkaluja on käytettävissä eniten käytetyille 3D-mallinnusohjelmille kuten 3D Studio Maxille, Mayalle ja Blenderille. (OGRE – Wikipedia 2009.)

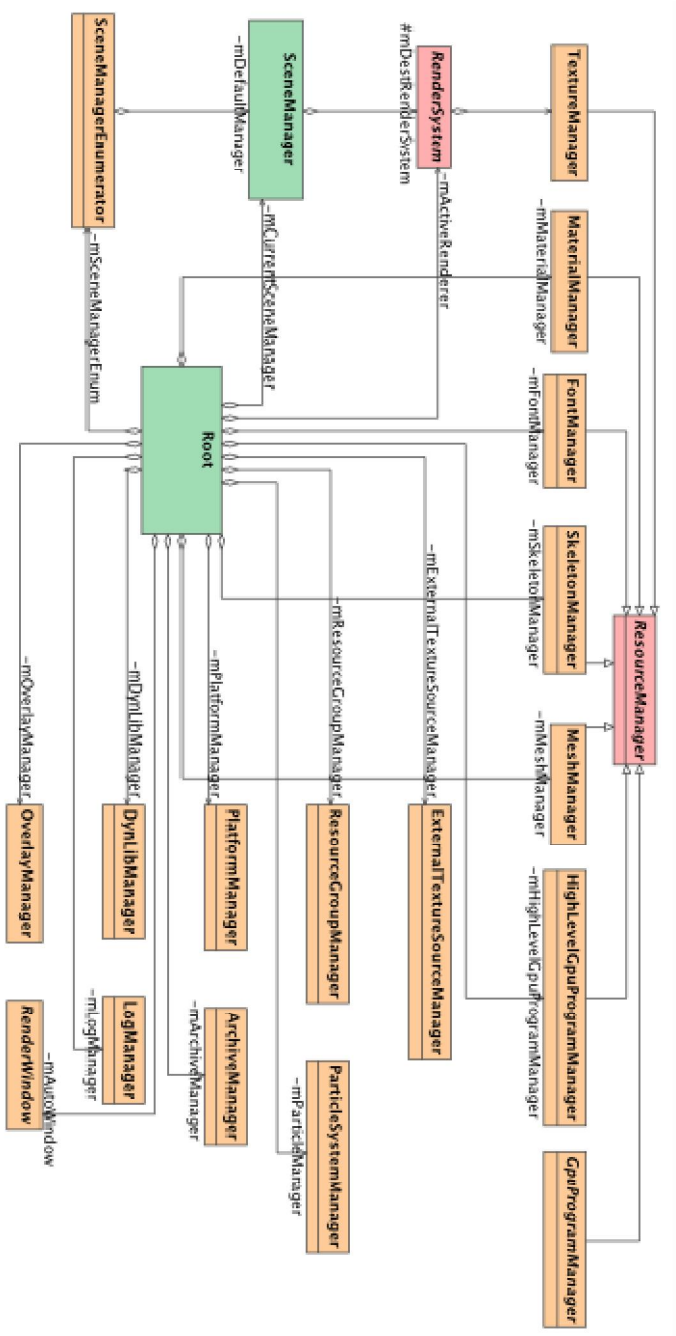
3.2 OGRE 3D -grafiikkamoottorin arkkitehtuuri

Kuvassa 2 esitetään OGRE 3D -grafiikkamoottorin arkkitehtuuri. Luokkakaaviossa esitetään OGRE:n Root-luokka ja Manager-luokat, joiden avulla päästään käsittelemään jokaiselle luokalle ominaisia piirteitä.

Root-luokka on yhdistettynä SceneManager-luokkaan, joka mahdollistaa erityyppisten skene-grafiikoiden latauksen 3D-grafiikkamoottorin käyttöön.

RenderSystem-luokka on abstrakti luokka, joka erottaa Root-luokan OpenGL- ja Direct3D-rajapinnoista.

ResourceManager-luokka on myös abstrakti luokka, jonka useimmat manageriluokat perivät. Näiden avulla päästään käsiksi useisiin eri lähteisiin, kuten tekstuureihin ja fontteihin. Kaikki muut luokat kaaviossa pois lukien RenderSystem, SceneManager ja RenderWindow on suunniteltu ainokaisiksi. Tämä takaa sen, ettei näistä luokista voi olla olemassa kuin yksi instanssi kerrallaan. Näin ollen OGRE-moottoreita voi olla sovelluksessa vain yksi, mutta se voi hallita useita ikkunoita ja toistaa grafiikkaa useisiin eri alueisiin.



Kuva 2 OGRE 3D-grafiikkamoottorin arkkitehtuuri
(Documentation Architecture 2009.)

3.3 OGRE 3D-grafiikkamoottorin ominaisuuksia

OGRE:ssa on suuri määrä graafisia ominaisuuksia, jotka ovat helposti käytettävissä missä tahansa OGRE-sovelluksessa. Graafisia ominaisuuksia ovat mm:

- Työkalut, joiden avulla saadaan ulkoisten mallinnusohjelmien mallit toimimaan OGRE-moottorin kanssa.
- Luurankomaisen animaation käsittely, jonka avulla hahmot saadaan liikkeelle.
- Hyvin muokattava ja joustava skene-hallinta, joka ei ole sidottu mihinkään tiettyyn tyyppiin. Käyttää valmiiksi määriteltyjä skenemanagereita, joiden avulla erityyppiset skenet saadaan täysin käyttöön.
- BSP-pohjainen liitännäinen, joka mahdollistaa nopean skene-mallintamisen esimerkiksi Quake3-pelin tasoista.
- Terrain-liitännäinen, joka mahdollistaa maastojen mallintamisen.
- Hierarkkinen skene-grafiikka, mikä mahdollistaa eri elementtien keskinäisen suhteen. Tämän avulla elementit voivat seurata toisiaan tai toimia muuten ennalta määrätyllä tavalla suhteessa toisiin.
- Elementeille määriteltävät reitit, joita pitkin elementit haluttaessa voivat liikkua tai monia eri animaatioita, joita ne suorittavat eri tiloissa.
- Läpinäkyvät objektit, jotka käsitellään automaattisesti ilman, että suunnittelijan täytyy erityisesti selvittää, kuinka graafinen toisto suoritetaan.
- Joustava sumuefektin käyttö ja kontrollointi.
- Joustava liitännäisarkkitehtuuri, mikä mahdollistaa moottorin laajentamisen ilman uudelleen rakentamista.
- Muistin virheenjäljitys, joka auttaa tunnistamaan muistivuotoja. (CurrentOgreFeatures 2009.)

4 KEHITYSYMPÄRISTÖ

Tässä luvussa käsitellään kehitysympäristöä, joka suoriutuu OGRE- ja Qt-kirjastojen yhteistyöstä Linux-ympäristössä. Kehitysympäristö asennetaan lokakuussa 2009 julkaistuun Debian-pohjaiseen Ubuntu 9.10 -käyttöjärjestelmään.

Kehitysympäristön tarkoituksena on saada toimimaan OGRE:n 3D -grafiikkaominaisuudet yhdessä Qt-kirjastojen kanssa. Tämä edellyttää sitä, että kehitysympäristöön on asennettava Qt- ja OGRE-kirjastot sekä niiden tarvitsemat riippuvuudet ja työkalut.

Kun kehitysympäristö on asennettu, sovelluskehittäjä pystyy toteuttamaan graafisen käyttöliittymäsovelluksen, jossa voidaan esittää 3D-grafiikkaa OGRE:n pitkälle kehittyneiden ominaisuuksien avulla.

4.1 Kehitysympäristön asentaminen

Tässä luvussa käsitellään kehitysympäristön asentamista Linux-ympäristöön. Asennusvaiheet käydään läpi siinä järjestyksessä kuin asennus on tarkoitus tehdä. Toisenlainen asennusjärjestys saattaa johtaa toimimattomaan kehitysympäristöön.

4.1.1 Qt-kirjastojen asentaminen

Ensimmäisessä vaiheessa asennetaan Qt-kirjastot, joiden avulla voidaan toteuttaa monimutkaisia graafisia sovelluksia. Kuitenkin ennen kuin itse kirjasto voidaan asentaa, täytyy asentaa Qt:n tarvitsemat riippuvuudet. Nämä riippuvuudet on listattu seuraavasti:

- libgstreamer0.10-dev
- libfontconfig1-dev
- libx11-dev

- libxcursor-dev
- libxext-dev
- libxft-dev
- libxi-dev
- libxrandr-dev

Näiden pakettien asentaminen Ubuntuun tapahtuu kirjoittamalla seuraava komento konsoliin:

```
sudo aptitude install <paketin nimi>
```

Paketit asennetaan aptitude-ohjelmalla käyttäen root-oikeuksia. Kun kehityskirjastot, joihin Qt:lla on riippuvuus, ovat asennettuna, voidaan asentaa itse Qt-kirjastot. Tässä esimerkissä asennetaan Qt 4.6 Beta -paketti. Sen asentaminen tapahtuu konsolissa seuraavilla komennoilla:

```
cd /tmp
wget http://get.qt.nokia.com/qt/source/qt-everywhere-opensource-src-4.6.0-beta1.tar.gz
gunzip qt-everywhere-opensource-src-4.6.0-beta1.tar.gz
tar xvf qt-everywhere-opensource-src-4.6.0-beta1.tar
cd /tmp/qt-everywhere-opensource-src-4.6.0-beta1/
./configure
make
sudo make install
```

Wget-komento lataa tiedoston internetistä siitä polusta, joka sille on ilmoitettu. Lataamisen jälkeen tiedosto puretaan gunzip-ohjelmalla ja vielä tar-paketointiohjelmalla. Purkamisen jälkeen ajetaan configure-skripti puretussa hakemistossa. Kun scripti on onnistuneesti suoritettu, voidaan Qt:n lähdekoodi kääntää make-komennolla. Lopuksi käännetyt kirjastot ja binäärit asennetaan järjestelmän käyttämiin kansioihin.

Tämän jälkeen Qt-kirjastojen asennus on valmis. Jotta työkalut toimisivat helpommin, voidaan kirjaston hakemistopolku vielä asettaa ympäristömuuttujaan. Se tapahtuu lisäämällä /etc/profile-tiedoston loppuun seuraavat kaksi riviä:

```
PATH=/usr/local/Trolltech/Qt-4.6.0/bin:$PATH
```

```
export PATH
```

4.1.2 OGRE-kirjastojen asentaminen

Aluksi täytyy asentaa useita paketteja ennen kuin OGRE-kirjaston pystyy kääntämään. Asennettavat paketit on listattu seuraavasti:

- subversion
- libxxf86vm-dev
- nvidia-cg-toolkit
- libfreetype6-dev
- libpcre3-dev
- libopenexr-dev
- freeglut-dev
- mesa-common-dev
- libtiff4-dev
- libglademm-2.4-dev
- libcppunit-dev
- cmake
- libxt-dev
- libxaw7-dev
- libfreeimage-dev
- libzzip-dev

Paketit asennetaan samalla tavalla kuin aikaisemmin kirjoittamalla seuraava komento konsoliin.

```
sudo apt-get install <paketin nimi>
```

OGRE 3D -moottorin asentaminen on hyvin yksinkertaista. Se tapahtuu kirjoittamalla konsoliin seuraavat komennot:

```
svn co https://svn.ogre3d.org/svnroot/ogre/trunk ogre
cd ogre
mkdir build && cd build
```

```
cmake ..
make
sudo make install
sudo ldconfig
```

Komentojen alussa ladataan svn-ohjelmaa käyttäen viimeisin versio OGRE:sta versionhallinnasta. Tämän jälkeen ajetaan cmake-ohjelma, joka generoi tarvittavat tiedostot make-sovellusta varten. Kun cmake-sovellus on onnistuneesti suoritettu ilman virheitä, voidaan suorittaa make-komento, joka kääntää OGRE:n lähdekoodit. Lopuksi käännetyt kirjastot ja binäärit asennetaan järjestelmään. Ldconfig-komento linkittää OGRE:n kirjastot niin, että ne löytyvät sovellusten suoritusten aikana.

Asennuksen jälkeen, jos mitään ongelmia ei esiintynyt, on kehitysympäristö valmis käytettäväksi. Täytyy kuitenkin huomioida, että viimeisimpien kehitysversioiden käyttö saattaa aiheuttaa epävakautta tai sovellus ei välttämättä toimi ollenkaan.

5 SOVELLUSTEN KEHITTÄMINEN

Tässä luvussa käydään läpi esimerkkisovelluksen kehittämistä. Selvitetään mitä tiedostoja projektissa täytyy olla, mitä ne sisältävät ja mikä niiden tarkoitus on projektin kannalta. Sen lisäksi selvitetään kuinka OGRE-kirjastot saadaan toimimaan Qt-sovelluksessa. Sovellusten kehittäminen voidaan aloittaa, kun kehitysympäristö on ensin onnistuneesti asennettuna.

5.1 Projektin tiedostot

Projektissa tarvitaan tietyt tiedostot, jotta projekti on mahdollista kääntää ja linkittää. Tämän lisäksi projektissa täytyy olla tiedostoja erilaisia asetuksia varten, joita käytetään ohjelman suorituksen aikana. Näitä asetustiedostoja, joita projektissa tarvitaan ovat plugins.cfg-, resources.cfg-, ogre.cfg- ja projektin oma .pro-tiedosto. Näistä tiedostoista on seuraavaksi esitetty tarkempi kuvaus.

Plugins.cfg-tiedosto määrittelee mitä liitännäisiä sovellus käyttää suoritusvaiheessa. Sen sisältö ei vaikuta itse käännös- tai linkitysvaiheeseen. Esimerkkitiedoston sisältö on esitetty seuraavaksi.

```
# Defines plugins to load
# Define plugin folder -- Plugin folder is ignored
# in Project Builder build -- plugins are found in Resources/
# in the Main (Application) bundle, then in the Framework bundle
PluginFolder=/usr/local/lib/OGRE

Plugin=RenderSystem_GL
Plugin=Plugin_ParticleFX
Plugin=Plugin_BSPSceneManager
Plugin=Plugin_OctreeSceneManager
```

Tässä esimerkissä ilmoitetaan ensimmäisenä OGRE:lle mistä kansiosta se löytää tarvittavat liitännäiset. Tämän jälkeen ilmoitetaan mitä liitännäisiä sovelluksessa aiotaan käyttää. Jos tästä tiedostosta puuttuu jokin liitännäisistä, jota käytetään

sovelluksessa, sovelluksen suoritus keskeytyy. OGRE ei automaattisesti etsi tarvittavia liitännäisiä vaan ne on esiteltävä tässä tiedostossa.

Resources.cfg-tiedosto sisältää polut käytettäviin ulkoisiin lähteisiin kuten 3D-malleihin. Sen sisältö voi olla esimerkiksi seuraavanlainen:

```
# Resource locations to be added to the default path
[General]
FileSystem=Data/samples
```

Ogre.cfg-tiedosto sisältää tarvittavat graafiset määrittelyt OGRE:a varten. Joissakin kohdissa, kuten resoluution tarkkuudessa, voidaan asettaa useampiakin arvoja. Tällöin OGRE valitsee käyttöönsä tilanteeseen sopivimman. Tiedostossa määritetään myös näytön päivitystaajuus sekä käytetäänkö koko näyttöä vai vain ikkunaa. Tiedoston sisältö voi esimerkiksi olla seuraavanlainen:

```
Render System=OpenGL Rendering Subsystem
[OpenGL Rendering Subsystem]
Display Frequency=60 MHz
FSAA=0
Full Screen=Yes
RTT Preferred Mode=PBuffer
Video Mode=800x600
```

Projektin oma .pro-tiedosto sisältää tarvittavat määrittelyt sovelluksen kääntöä ja linkittämistä varten. Siinä ilmoitetaan mitkä lähdekooditiedostot ja kirjastot liittyvät projektiin ja mistä ne löytyvät. Tiedostoon voidaan tehdä erilaisia määrityksiä eri käyttöjärjestelmiä varten.

HEADERS-muuttuja sisältää kaikki käytettävät otsikkotiedostot, SOURCES-muuttuja sisältää kaikki lähdekooditiedostot ja FORMS-muuttuja sisältää kaikki käyttöliittymätiedostot. QT-muuttuja sisältää käytettävät Qt-moduulit ja TARGET-muuttuja kertoo käännetyn binäärin nimen.

INCLUDEPATH-muuttuja sisältää polut, joista projektissa käytetyt otsikkotiedostot löytyvät. Lisäksi LIBS-muuttuja sisältää kirjastot, joita käytetään linkittämisessä. Projektitiedoston sisältö voi esimerkiksi olla seuraavanlainen.

```
QT += svg
TEMPLATE = app
TARGET = QOgreWidget_Demo
DEPENDPATH += .
INCLUDEPATH += ./include

# Input
HEADERS += include/demologic.h \
   include/aboutwidget.h \
   include/menuwidget.h \
   include/demowidget.h \
   include/QOgreWidget.h \
   include/ogrewidget.h
SOURCES += src/main.cpp \
   src/demologic.cpp \
   src/aboutwidget.cpp \
   src/menuwidget.cpp \
   src/demowidget.cpp \
   src/QOgreWidget.cpp \
   src/ogrewidget.cpp

LIBS += -lOgreMain
INCLUDEPATH += /usr/local/include/OGRE

FORMS += ui/menuwidget.ui \
   ui/aboutwidget.ui \
   ui/demowidget.ui
```

5.2 Projektin rakenne

Tässä luvussa käydään läpi projektin suunnittelu- ja toteutusvaihetta. Esimerkkiprojektin yleiskuva on esitetty liitteessä 1. Siitä käy ilmi mistä luokista luokat periytyvät sekä minkä luokan olioita on kunkin olion alaisuudessa.

Liitteessä 2 on esitetty kaavio, josta käy ilmi minkälainen rakenne on OgreWidget-luokalla, jonka avulla OGRE:n ominaisuuksia käytetään. Tässä luokassa on kuitenkin yksi erityispiirre tällä toteutuksella. Luokasta voi olla vain yksi toteutus

kerrallaan, koska OGRE:n juuriluokka toimii ainokaisena. Tästä seuraa se, että joka kerta, kun OgreWidget-olio tuhotaan, täytyy myös OGRE:n Root-olio tuhota. Ellei Root-oliota tuhota, seuraavalla kerralla, kun OgreWidget-oliota luodaan, sovelluksen suoritus todennäköisesti keskeytyy. Esimerkkisovelluksen lähdekoodi esitetään liitteessä 3.

Projektin lähdekoodissa on muutama kohta, jotka mahdollistavat OGRE 3D-grafiikkamoottorin käytön. DemoWidget.ui-tiedostossa on tyhjä ogreRenderWidget-widgetti, joka DemoWidget:in rakentajassa annetaan OgreWidget:in vanhemmaksi seuraavalla tavalla:

```
m_ogreWidget = new OgreWidget( ui->ogreRenderWidget );
```

DemoWidget:ssa on OgreWidgetiin osoitin, jonka nimi on m_ogreWidget. Osoitin on määritelty demowidget.h-tiedostossa seuraavasti:

```
OgreWidget* m_ogreWidget;
```

Tämän avulla grafiikkamoottori piirtää grafiikan DemoWidget:ssä olevaan tyhjään ogreRenderWidget-widgettiin.

Kun OgreWidget ensimmäisen kerran päivitetään kutsumalla paintEvent-funktiota, funktio kutsuu createRenderWindow-funktiota, joka on peritty QOgreWidget:stä. PaintEvent-funktio on seuraavanlainen:

```
void OgreWidget::paintEvent(QPaintEvent *) {
        if(!m_renderWindow) {
                createRenderWindow();
                setupResources();
                setupScene();
        }
        update();
}
```

Aluksi createRenderWindow-funktio alustaa OGRE-moottorin ja lukee käytettävät asetustiedostot kutsumalla configure-funktiota. Configure-funktio selvittää lopuksi

vielä mitkä 3D-ajurit ovat järjestelmässä käytössä. Ajureista käyttöön otetaan ensimmäinen.
Configure-funktio on seuraavanlainen:

```
void QOgreWidget::configure(void) {
if (m_ogreRoot)
return;

m_ogreRoot = new Ogre::Root("Data/plugins.cfg", "Data/ogre.cfg", "Data/ogre.log");
if (!m_ogreRoot->restoreConfig()) {
// setup a renderer
const Ogre::RenderSystemList renderers = m_ogreRoot->getAvailableRenderers();
assert(!renderers.empty()); // we need at least one renderer to do anything useful

//Ogre::RenderSystem *renderSystem = chooseRenderer(renderers);
Ogre::RenderSystem *renderSystem = *renderers.begin();
assert(renderSystem); // user might pass back a null renderer, which would be bad!

m_ogreRoot->setRenderSystem(renderSystem);
QString dimensions = QString("%1x%2").arg(this->width()).arg(this->height());
renderSystem->setConfigOption("Video Mode", dimensions.toStdString());

// initialize without creating window
m_ogreRoot->getRenderSystem()->setConfigOption("Full Screen", "No");
m_ogreRoot->saveConfig();
}
m_ogreRoot->initialise(false);
}
```

Tämän jälkeen createRenderWindow-funktio luo OGRE:lle ikkunan, johon se voi toistaa grafiikkaa. Tässä tapauksessa ikkunana toimii DemoWidget:ssä oleva ogreRenderWidget, jonka OGRE kaappaa itsellensä käyttöön. Ikkuna luodaan Linux:iin seuraavalla tavalla:

```
QX11Info info = x11Info();
Ogre::String winHandle;
winHandle  = Ogre::StringConverter::toString((unsigned long)(info.display()));
winHandle += ":";
winHandle += Ogre::StringConverter::toString((unsigned int)(info.screen()));
winHandle += ":";
```

```
winHandle += Ogre::StringConverter::toString((unsigned long)(this->parentWidget()->winId()));
params["parentWindowHandle"] = winHandle;
m_renderWindow = m_ogreRoot->createRenderWindow("View" +
Ogre::StringConverter::toString((unsigned long) this), this->parentWidget()->width(), this-
>parentWidget()->height(), false, &params);

// take over ogre window
WId ogreWinId = 0x0;
m_renderWindow->getCustomAttribute("WINDOW", &ogreWinId);
assert(ogreWinId);
create(ogreWinId);
```

Aluksi tutkitaan ogreRenderWidget:in tunnus ja koko. Tämän jälkeen OGRE asettaa ikkunan omaan käyttöönsä samankokoisena.

PaintEvent-funktio kutsuu myös setupResources- ja setupScene-funktioita. SetupResources-funktio lataa ulkoiset lähteet tiedostoista kuten 3D-hahmojen mallit ja ulkonäön. SetupScene-funktio puolestaan luo 3D-avaruuteen kameran ja 3D-hahmot niin, että kamera kuvaa 3D-avaruuden ja sen näkymä toistetaan ogreRenderWidget:ssa.

Lopulta SetupScene-funktiossa käynnistetään OgreWidget:in oma ajastin halutulla intervallilla. Ajastimen timerEvent-funktiossa kutsutaan QogreWidget:in update-funktiota.

Update-funktio on seuraavanlainen:

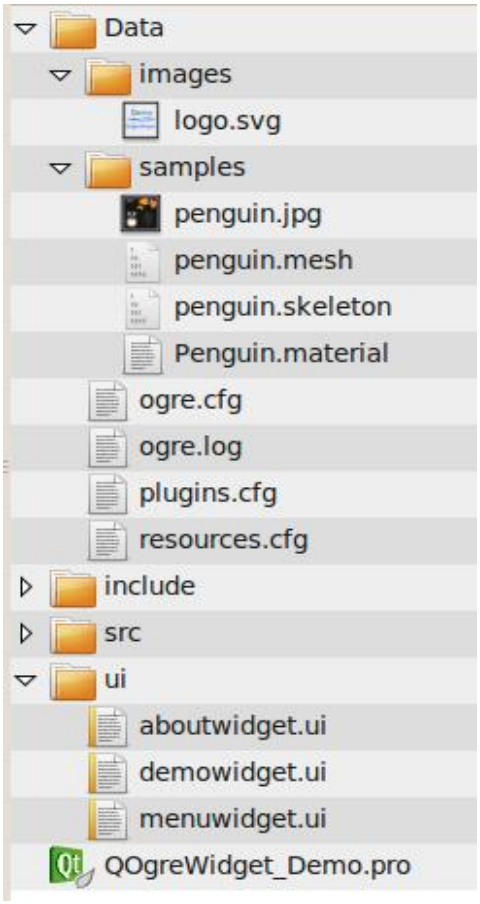
```
void QOgreWidget::update(void) {
        if (m_renderWindow) {
                m_ogreRoot->_fireFrameStarted();
                m_renderWindow->update();
                m_ogreRoot->_fireFrameEnded();
        }
}
```

Aluksi OGRE-moottorille kerrotaan, että päivitys alkaa ja lopuksi, että päivitys loppuu. Josta seuraa, että grafiikan piirtäminen on ensisijaisessa asemassa. Muuten

funktio käskee OGRE:a päivittämään ikkunansa graafisen sisällön. Sovelluksen päivitystaajuus riippuu täysin tämän funktion kutsujen taajuudesta.

OgreRenderWidgetin erilaisuudesta huolimatta, siinä toimii täysin QWidgettien ominaisuudet kuten signaalit ja tapahtumat. Tästä johtuen myös hiiren ja näppäimistön käyttäminen ogreRenderWidget:n kohdalla tuottaa tapahtumia. Tämä mahdollistaa käyttäjän ja widgetin välisen vuorovaikutuksen.

Esimerkkiprojektissa tarvittavat tiedostot on sijoitettu tiettyihin kansioihin. Kuvassa 4 havainnollistetaan projektin rakenne tiedostotasolla. Asetustiedostot sijaitsevat Data-kansiossa. Samassa kansiossa on myös muita kansioita. Kansio nimeltä samples sisältää 3D-mallin tiedostot. Näitä tiedostoja ovat Penguin.material, penguin.mesh, penguin.skeleton ja penguin.jpg. Kansio nimeltä images sisältää logo.svg-tiedoston, joka toistetaan DemoWidgetissä. DemoWidgetin ja muiden widgettien ulkoasut ovat esitettynä liitteessä 4.



Kuva 4 Projektin tiedostorakenne

5.3 Projektin testaus

Tässä luvussa käydään läpi kuinka projektin toimivuus voidaan testata ja miltä sovellus näyttää suorituksen aikana. Ennen kuin projektin suoritusta voidaan testata, se täytyy myös kääntää ja linkittää. Nämä tapahtuvat kirjoittamalla seuraavat komennot konsoliin projektin juurihakemistossa:

```
qmake QogreWidget_Demo.pro
make
./QOgreWidget_Demo
```

Qmake-komento tekee tiedostot make-sovellusta varten projektitiedoston perusteella. Kun se on ajettu onnistuneesti, voidaan suorittaa make-komento, joka kääntää lähdekoodit binääriksi. Lopuksi käännetty sovellus käynnistetään suorittamalla viimeinen komento.

Sovelluksen suoritusvaiheesta on kuvia liitteessä 4. Jos käännös- tai linkitysvaiheessa tulee esiin virheitä, kannattaa tarkistaa QogreWidget_Demo.pro-tiedostossa olevien polkujen oikeellisuus. Jos taas sovelluksen suoritus keskeytyy Start-painikkeen painamisen jälkeen, kun DemoWidget-rakennetaan, on ongelma todennäköisimmin OGRE:n asetustiedostoissa, jotka sijaitsevat esimerkkiprojektissa Data-kansiossa.

6 KEHITYSYMPÄRISTÖ MAEMOSSA

Tässä luvussa käsitellään kehitysympäristöä, jonka avulla voidaan toteuttaa esimerkkiprojekti Maemolle. On kuitenkin huomattava, että kehitysympäristössä nVidian Cg Toolkit:in ominaisuudet eivät toimi tällä hetkellä Armelilla, koska nVidia ei ole tehnyt Cg-toolkit:sta asennusbinääriä Arm-arkkitehtuurille. Tästä paketista ei myöskään ole saatavissa lähdekoodia, koska se on suljettu, joten binäärien kääntäminen ei ole mahdollista.

Tulevaisuudessa, jos nVidia tekee Arm-arkkitehtuurille asennuspaketit, Cg-Toolkit:in ominaisuuksien käyttö mobiililaitteessa on täysin mahdollista. Tämä ongelma ei kuitenkaan vaikuta esimerkkiprojektin toimintaan millään tavalla, koska Cg-Toolkitin ominaisuuksia ei ole käytössä.

6.1 Kehitysympäristön asentaminen

Tässä luvussa käydään läpi mitä ohjelmistokomponentteja kehitysympäristöön kuuluu ja kuinka ne asennetaan Linux-käyttöjärjestelmään. Kehitysympäristö rakennetaan Scratchbox-hiekkalaatikkoympäristöön, joka asennetaan Linux-järjestelmään. Scratchbox:in sisälle puolestaan asennetaan Maemo 5 SDK ja Qt- ja OGRE-kirjastot sekä tarvittavat riippuvuudet. Scratchbox:in avulla voidaan simuloida Maemo-pohjaisia laitteita ja kääntää binäärejä Arm-arkkitehtuuria varten.

6.1.1 Maemo 5 SDK:n asentaminen

Ensimmäisessä vaiheessa asennetaan Scratchbox-hiekkalaatikkoympäristö Linux-koneelle. Sen tarkoituksena on toimia eristettynä ympäristönä, jossa voidaan kehittää ohjelmistoa eri prosessoriarkkitehtuureille. Syy Scratchbox:n käyttöön on se, että Maemo-pohjaiset laitteet ovat tähän asti olleet Arm-prosessoreilla varustettuja.

Scratchbox asennetaan kirjoittamalla seuraavat komennot konsoliin:

```
cd /tmp
wget http://repository.maemo.org/stable/5.0/maemo-scratchbox-install_5.0.sh
sudo chmod a+x ./maemo-scratchbox-install_5.0.sh
sudo ./maemo-scratchbox-install_5.0.sh -u <username>
```

Wget-komennolla ladataan skriptitiedosto internetistä ilmoitetusta sijainnista. Tämän jälkeen muutetaan tiedoston oikeuksia, jotta se voidaan suorittaa. Viimeinen komento suorittaa skriptitiedoston annetulla käyttäjänimellä.

Seuraavaksi asennetaan Maemo SDK Scratchbox:iin. Jos Maemo SDK:n asennusvaiheessa tulee vastaan ongelmia oikeuksien kanssa, täytyy kirjautua uudestaan sisään käyttöjärjestelmään, jotta uudet ryhmäasetukset tulevat voimaan. Maemo SDK asennetaan kirjoittamalla seuraavat komennot konsoliin:

```
wget http://repository.maemo.org/stable/fremantle/maemo-sdk-install_5.0.sh
sudo chmod a+x ./maemo-sdk-install_5.0.sh
./maemo-sdk-install_5.0.sh
```

Nokian suljetut binäärit täytyy asentaa tämän jälkeen vielä erikseen niitä arkkitehtuureja varten, joita aiotaan käyttää. Niiden lisäämiseksi täytyy käydä http://tablets-dev.nokia.com/eula/index.php -sivulla, jossa näytetty linkki täytyy kopioida Scratchbox:in sources.list -tiedostoon. Asennus tapahtuu kirjoittamalla seuraavat komennot konsoliin:

```
apt-get update
fakeroot apt-get install nokia-binaries nokia-apps
```

Apt-get-sovellus suoritetaan update-parametrilla, mikä päivittää pakettilistan. Tämän jälkeen asennetaan halutut paketit Scratchbox:n sisäisillä root-oikeuksilla.

Kun Nokian binäärit ovat asennettu, viimeisessä vaiheessa asennetaan Xephyr-sovellus. Xephyr:n asennus tapahtuu Ubuntuun kirjoittamalla seuraava komento konsoliin:

```
sudo aptitude install xserver-xephyr
```

Xephyriä käytetään näyttämään sovelluksen graafinen ulkoasu. Se on ikkuna, johon sovelluksen ulkoasu toistetaan. Sille voidaan antaa erilaisia käynnistysparametreja, jotta se toimisi halutulla tavalla. Esimerkiksi voidaan käyttää resoluutio- ja tarkkuusparametreja. Xephyr:n käytöstä on kerrottu lisää luvussa 7.

6.1.2 Maemo Qt-kirjastojen asentaminen

Tässä luvussa käsitellään Qt-kirjastojen asennusta Scratchbox:iin. Nämä kirjastot täytyy asentaa myös kaikille arkkitehtuureille, joita aiotaan käyttää, samalla tavalla kuin edellisessä vaiheessa. Kirjastot asennetaan samalla tavalla kuin Ubuntussa. Komennot ovat selitetty luvussa 4.1.1. Kirjastojen asennus tapahtuu kirjoittamalla konsoliin seuraavat komennot:

```
wget http://get.qt.nokia.com/qt/source/qt-x11-maemo-src-4.6.0-alpha1.tar.gz
gunzip qt-x11-maemo-src-4.6.0-alpha1.tar.gz
tar xvf qt-x11-maemo-src-4.6.0-alpha1.tar
cd qt-x11-maemo-src-4.6.0-alpha1
./configure
make
fakeroot make install
```

Asennuksen jälkeen Maemo-kehitysympäristö on valmis Qt-kehitystä varten. Tässä vaiheessa voidaan siis toteuttaa ja simuloida Qt-sovelluksia ilman kehitysympäristön muutoksia.

6.1.3 OGRE-kirjastojen asentaminen

OGRE-kirjastojen asennus Scratchbox:iin vaatii hieman enemmän työtä. Ensiksi täytyy asentaa muutamia paketteja ennen kuin OGRE-kirjaston pystyy kääntämään. Asennettavat paketit ovat listattuna seuraavaksi:

- automake
- cmake
- libtool

Pakettit asennetaan yksittäin kirjoittamalla seuraavanlainen komento konsoliin.

```
fakeroot apt-get install <paketin nimi>
```

Cmake-paketti Armelille täytyy kuitenkin asentaa käsin, koska sen viimeisin versio ei tämän työn toteuttamisen hetkellä toimi Armelilla. Se asennetaan kirjoittamalla seuraavat komennot konsoliin:

```
wget http://repository.maemo.org/pool/maemo5.0/free/c/cmake/cmake-data_2.6.3-
2maemo4+0m5_all.deb
wget http://repository.maemo.org/pool/maemo5.0/free/c/cmake/cmake_2.6.3-
2maemo4+0m5_armel.deb
fakeroot dpkg -i cmake-data_2.6.3-2maemo4+0m5_all.deb
fakeroot dpkg -i cmake_2.6.3-2maemo4+0m5_armel.deb
```

Komentojen alussa ladataan kaksi debian-pakettia internetistä. Tämän jälkeen molemmat paketit asennetaan järjestelmään dpkg-sovelluksen avulla käyttäen Scratchbox:n sisäisiä root-oikeuksia.

Tämän jälkeen täytyy vielä kääntää muutama paketti lähdekoodeista, koska niistä ei ole saatavilla valmiita asennuspaketteja. Seuraavaksi käydään läpi ZZIPlib-, FreeImage- ja OpenGL-kirjastojen asennus.

ZZIPlib:n asennus lähdekoodista tapahtuu kirjoittamalla seuraavat komennot konsoliin:

```
wget http://sourceforge.net/projects/zziplib/files/zziplib13/0.13.58/zziplib-0.13.58.tar.bz2/download
tar -xjf zziplib-0.13.58.tar.bz2
cd zziplib-0.13.58
./configure && make && make check && fakeroot make install
```

FreeImage asennetaan seuraavalla tavalla:

```
wget http://downloads.sourceforge.net/freeimage/FreeImage3130.zip
unzip FreeImage3130.zip
cd FreeImage
make && fakeroot make install
```

OpenGL-kehityskirjasto tarvitsee asentaa x86-arkkitehtuurille. Armelille on valmiiksi asennettuna OpenGL ES, joka on tarkoitettu mobiililaitteita varten. Kirjaston asennetaan seuraavalla komennolla:

```
fakeroot apt-get install libgl-dev
```

OGRE-kirjastot asennetaan samalla tavalla kuin Ubuntussa. Komennot ovat selitettynä luvussa 4.1.2. Kirjastot asennetaan kirjoittamalla seuraavat komennot konsoliin:

```
svn co https://svn.ogre3d.org/svnroot/ogre/trunk ogre
cd ogre
mkdir build && cd build
cmake ..
```

Ennen kääntämistä, jos OGRE:a asennetaan Armelille, täytyy muokata CmakeLists.txt -tiedostoa ogre-kansiossa. Muutama rivi täytyy kommentoida, ettei kääntäjä yritä käyttää vääriä määrityksiä. Seuraavaksi on esimerkki kommentoi-duista riveistä.

```
# Set compiler specific build flags
#if (CMAKE_COMPILER_IS_GNUCXX)
#  add_definitions(-msse)
#endif ()
```

Lisäksi täytyy muokata malloc.c.h -tiedostoa niin, että pthread_mutexattr_setkind_np-funktion tilalle vaihdetaan pthread_mutexattr_settype-funktio. Tämän jälkeen itse kääntäminen voidaan aloittaa. Asennus jatkuu kirjoittamalla seuraavat komennot konsoliin:

```
make
fakeroot make install
```

Jos käännösvaiheessa, kun asennetaan OGRE:a Armelille, tulee virheitä liittyen käyttöjärjestelmän tunnistukseen, voidaan OgrePlatformInformation.cpp -tiedostoa muokata niin, että _detectCpuIdentifier -funktio palauttaa aina ”ARMv7” -tekstin.

7 SOVELLUKSIEN KEHITTÄMINEN MAEMOLLE

Tässä luvussa käydään läpi kuinka esimerkkisovellus saadaan toimimaan Maemolla. Sovellusten kehittäminen voidaan aloittaa, kun kehitysympäristö on onnistuneesti asennettu. Tarkoituksena on selvittää kuinka Qt-sovelluksessa saadaan toimimaan OGRE:n ominaisuudet.

Sovelluksen kehittäminen tapahtuu täysin samalla tavalla kuin luvussa 5. Kehittämisen samanlaisuus johtuu siitä, että molemmissa tapauksissa sovellusta kehitetään Linux-käyttöjärjestelmälle. Projektissa on samat tiedostot ja sama tiedostorakenne. Joitakin asetustiedostoja täytyy muokata, niiltä osin joissa esitellään hakemistopolkuja. Muilta osin projekti toimii täysin samanlaisena x86- ja Arm-arkkitehtuureilla.

7.1 Projektin testaus

Projektin testaaminen vaatii muutamia välivaiheita. Ensimmäisenä rakennetaan projekti samalla tavalla kuin luvussa 5. Projekti käännetään kirjoittamalla seuraavat komennot konsoliin:

```
qmake QogreWidget_Demo.pro
make
```

Kääntämisen jälkeen täytyy luoda ikkuna, jossa sovellus ajetaan. Tämä onnistuu käynnistämällä Xephyr Scratchboxin ulkopuolella. Xephyr käynnistetään kirjoittamalla seuraava komento konsoliin:

```
Xephyr :2 -host-cursor -screen 800x480x16 -dpi 96 -ac -kb &
```

Tämän jälkeen näytölle ilmestyy tyhjä ikkuna. Kun ikkuna on luotu, tarkoituksena on saada sovellus näkymään siinä ja välittämään käyttäjän antamia syötteitä. Sovellus käynnistetään Scratchbox:issa kirjoittamalla seuraavat komennot konsoliin:

```
export DISPLAY=:2
af-sb-init.sh start
run-standalone.sh <executable> &
```

Export-komentolla kerrotaan mitä X-palvelimen istuntoa käytetään emuloimaan laitteen näyttöä. Seuraavaksi käynnistetään Maemo 5-emulaattori ja lopuksi suoritetaan luotu sovellus emulaattorissa. Tämän jälkeen sovellus näkyy ikkunassa. Sovelluksen suorituksesta emulaattorissa on kuvia liitteessä 5.

Sovelluksen kääntäminen Arm-arkkitehtuurille tapahtuu kirjoittamalla seuraavat komennot Armelilla:

```
qmake QogreWidget_Demo.pro
make
```

Tämän jälkeen binääri on valmis käytettäväksi Maemo 5-pohjaisella Arm-arkkitehtuurin laitteella. Laitteeseen täytyy kuitenkin asentaa OGRE:n ja Qt:n -kirjastot ennen kuin sovellus saadaan toimimaan kunnolla.

8 TULOKSET JA YHTEENVETO

Työssä selvitettiin kuinka on mahdollista integroida Qt-kirjastot ja OGRE 3D-grafiikkamoottori toimimaan samassa sovelluksessa. Selvitettiin lisäksi onko integrointi mahdollista toteuttaa myös Maemo 5-käyttöjärjestelmälle. Integroinnin toiminta todennettiin ohessa tehdyn esimerkkiprojektin avulla.

Aiheesta kiinnostavan tekee se, että Qt-kirjastojen avulla voidaan toteuttaa graafisia käyttöliittymäsovelluksia lähes kaikille käytetyimmille käyttöjärjestelmille, älypuhelimille ja Internet Tableteille. OGRE 3D-grafiikkamoottori puolestaan toimii jo nyt useimmilla työasema käyttöjärjestelmillä kuten Windowsilla, Linuxilla ja Max OS X:lla, mutta siinä on myös kannettavien laitteiden käyttämä OpenGL ES-tuki. Tämä tekee mahdolliseksi sen, että molempien erityisominaisuudet saadaan toimimaan myös Maemo-pohjaisilla laitteilla.

Työn ohessa toteutettu esimerkkiprojekti toimi samaan tapaan Ubuntu 9.10-Linuxissa ja Maemo 5-emulaattorissa. OGRE:n ja Qt:n tehokkuutta ei kuitenkaan voitu selvittää itse kannettavassa Maemo 5-laitteessa, koska niitä ei ollut vielä saatavilla.

Maemo 5-emulaattorissa 3D-kiihdytys on niin heikko, ettei sillä voi käytännössä suorittaa kuin erittäin kevyitä 3D-sovelluksia, joten suorituskykytestaus jäi pois tarpeettomana. Emulaattorissa sovelluksen grafiikka oli osin eri sävyistä ja osittain toistamatonta, mikä näkyi 3D-mallien osien puuttumisena. Ubuntussa esimerkkiprojekti toimi kuitenkin täysin moitteetta erittäin hyvällä vasteajalla. Se, että OGRE 3D-grafiikkamoottori toimi Qt-sovelluksen sisällä ei näyttänyt vaikuttavan lainkaan suorituskykyyn.

Tulevaisuudessa, kun kannettavien laitteiden 3D-suoritusteho kasvaa entisestään, voidaan tällä tekniikalla toteuttaa suurempia ja monimutkaisempia 3D-sovelluksia. Tällä hetkellä se on kuitenkin jo mahdollista tietokoneiden osalta. Tämä mahdollistaa sen, että voidaan toteuttaa huomattavasti helpommin sovelluksia, jotka toimivat eri käyttöjärjestelmillä ja joissa on käytössä 3D-grafiikkamoottori.

LÄHTEET

All Qt Modules [www-sivu][Viitattu: 01.10.2009] Saatavissa: http://doc.qt.nokia.com/4.6-snapshot/modules.html

CurrentOgreFeatures [www-sivu][Viitattu: 28.09.2009] Saatavissa: http://www.ogre3d.org/wiki/index.php/CurrentOgreFeatures

Documentation Architecture [www-sivu][Viitattu: 28.09.2009] Saatavissa: http://www.ogre3d.org/wiki/index.php/Documentation_Architecture

Junker, Gregory 2006. Pro OGRE 3D Programming. New York: Apress.

OGRE - Wikipedia [www-sivu][Viitattu: 28.09.2009] Saatavissa: http://en.wikipedia.org/wiki/OGRE

OGRE – Open Source 3D Graphics Engine [www-sivu][Viitattu: 28.09.2009] Saatavissa: http://www.ogre3d.org/about

Qt – A cross platform application and UI framework [www-sivu][Viitattu: 28.09.2009] Saatavissa: http://qt.nokia.com/?currentflipperobject=04fb871fbc28d8e8e46c51fe89ef8d67

Qt - Wikipedia [www-sivu][Viitattu: 28.09.2009] Saatavissa: http://fi.wikipedia.org/wiki/Qt

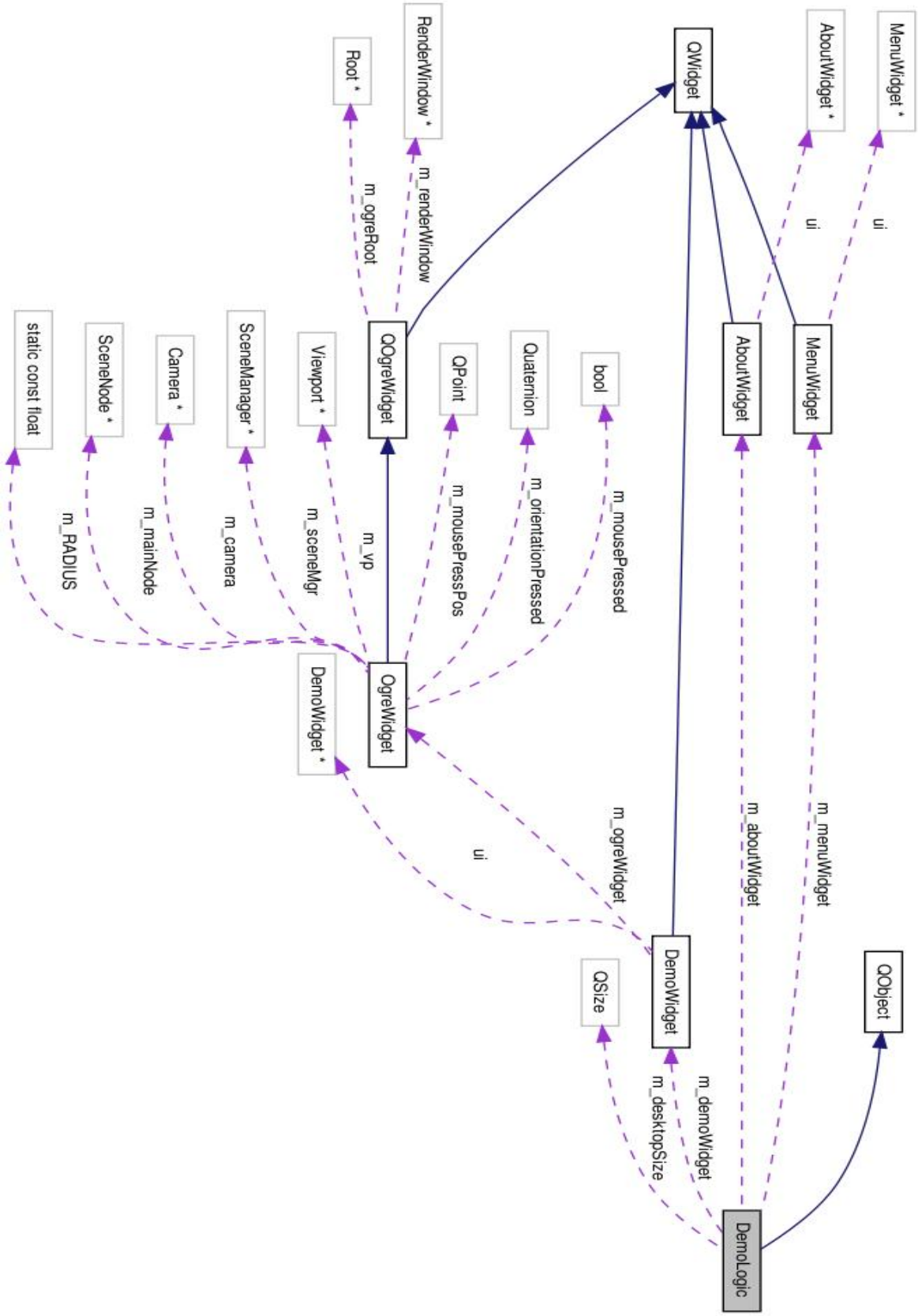
Qt 4.6 Technology Preview [www-sivu][Viitattu: 28.09.2009] Saatavissa: http://qt.nokia.com/developer/qt-4.6-technology-preview

Qt auttaa jo Symbianiin tuskastuneita – Tietokone.fi [www-sivu][Viitattu: 28.09.2009] Saatavissa: http://www.tietokone.fi/uutta/uutinen.asp?news_id=39023&tyyppi=1

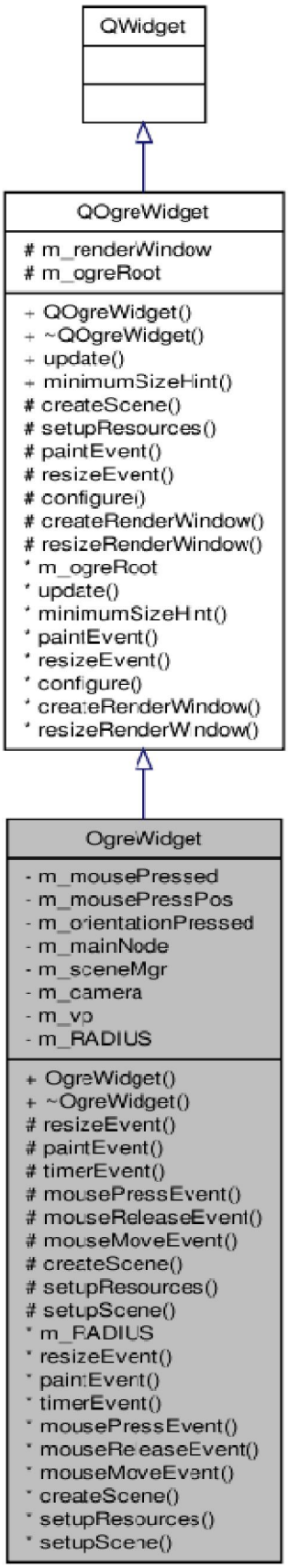
Qt Licensing [www-sivu][Viitattu: 06.10.2009] Saatavissa: http://qt.nokia.com/products/licensing

LIITTEET

Liite 1: Kaavio esimerkkisovelluksen luokista



Liite 2: OGRE:en liityvien luokkien kaavio



Liite 3: Esimerkkisovelluksen lähdekoodi (1/25)

main.cpp

```
#include <QApplication>
#include <QDesktopWidget>
#include "demologic.h"

/*!
 * Main
 * Creates DemoLogic object.
 */
int main(int argc, char **argv)
{
	QApplication app(argc, argv);

	/*!
	 * Gets screen size for OgreWidget.
	 * DemoWidget needs to resize before OgreWidget is initialized.
	 */
	QSize desktopSize = app.desktop()->screenGeometry().size();
	DemoLogic logic( desktopSize );
		return app.exec();
}
```

demologic.h

```
#ifndef DEMOLOGIC_H
#define DEMOLOGIC_H

#include <QApplication>
#include <QObject>
#include "demowidget.h"
#include "menuwidget.h"
#include "aboutwidget.h"

//!  A DemoLogic class.
/*!
  Class for application logic. Inherits QObject.
*/
class DemoLogic : public QObject
{
   Q_OBJECT

public:

   //! A constructor.
   /*!
```

Liite 3: Esimerkkisovelluksen lähdekoodi (2/25)

```
     Creates DemoLogic.
     \param desktopSize a QSize
   */
   DemoLogic(QSize desktopSize);

   //! A destructor.
   /*!
     Removes DemoLogic.
   */
   ~DemoLogic();

private:

   //! A private variable.
   /*!
     Pointer of DemoWidget.
   */
   DemoWidget* m_demoWidget;

   //! A private variable.
   /*!
     Pointer of MenuWidget.
   */
   MenuWidget* m_menuWidget;

   //! A private variable.
   /*!
     Pointer of AboutWidget.
   */
   AboutWidget* m_aboutWidget;

   //! A private variable.
   /*!
     size of desktop
   */
   QSize m_desktopSize;

private slots:

   //! A private slot function taking no arguments.
   /*!
     Creates MainMenu and sets it visible.
   */
   void showMenu();

   //! A private slot function taking no arguments.
   /*!
     Creates DemoWidget and sets it visible.
   */
```

Liite 3: Esimerkkisovelluksen lähdekoodi (3/25)

```
    void slotStart();

    //! A private slot function taking no arguments.
    /*!
      Creates AboutWidget and sets it visible.
    */
    void slotAbout();
};

#endif // DEMOLOGIC_H
```

demologic.cpp

```
#include "demologic.h"

DemoLogic::DemoLogic(QSize desktopSize)
{
    this->m_desktopSize = desktopSize;
    this->m_aboutWidget = 0;
    this->m_demoWidget = 0;
    this->m_menuWidget = 0;

    this->showMenu();
}

DemoLogic::~DemoLogic()
{
}

void DemoLogic::showMenu()
{
    this->m_menuWidget = new MenuWidget();
    this->m_menuWidget->show();

    connect(this->m_menuWidget, SIGNAL(signalStart()), this, SLOT(slotStart()));
    connect(this->m_menuWidget, SIGNAL(signalAbout()), this, SLOT(slotAbout()));
    connect(this->m_menuWidget, SIGNAL(signalQuit()), qApp, SLOT(quit()));
}

void DemoLogic::slotStart()
{
    this->m_demoWidget = new DemoWidget(this->m_desktopSize);
    this->m_demoWidget->showFullScreen();

    connect(this->m_demoWidget, SIGNAL(signalMenu()), this, SLOT(showMenu()));
}

void DemoLogic::slotAbout()
```

Liite 3: Esimerkkisovelluksen lähdekoodi (4/25)

```
{
   this->m_aboutWidget = new AboutWidget();
   this->m_aboutWidget->show();
}
```

menuwidget.h

```
#ifndef MENUWIDGET_H
#define MENUWIDGET_H

#include <QWidget>

namespace Ui {
   class MenuWidget;
}

//!  A MenuWidget class.
/*!
  Widget for main menu. Inherits QWidget.
*/
class MenuWidget : public QWidget
{
   Q_OBJECT

public:

   //! A constructor.
   /*!
     Creates MenuWidget.
     \param parent a QWidget*
   */
   MenuWidget(QWidget *parent = 0);

   //! A destructor.
   /*!
     Removes MenuWidget.
   */
   ~MenuWidget();

protected:

   //! A protected member function taking one argument and returning nothing.
   /*!
     \param e a QEvent*.
   */
   void changeEvent(QEvent *e);

private:
```

Liite 3: Esimerkkisovelluksen lähdekoodi (5/25)

```
    //! A private variable.
    /*!
      Pointer of UI.
    */
    Ui::MenuWidget *ui;

private slots:

    //! A private slot function taking no arguments.
    /*!
      Run after Start-button is clicked.
    */
    void slotStart();

    //! A private slot function taking no arguments.
    /*!
      Run after About-button is clicked.
    */
    void slotAbout();

    //! A private slot function taking no arguments.
    /*!
      Run after Quit-button is clicked.
    */
    void slotQuit();

signals:

    //! A signal taking no arguments.
    /*!
      emitted after Start-button is clicked.
    */
    void signalStart();

    //! A signal taking no arguments.
    /*!
      emitted after About-button is clicked.
    */
    void signalAbout();

    //! A signal taking no arguments.
    /*!
      emitted after Quit-button is clicked.
    */
    void signalQuit();
};

#endif // MENUWIDGET_H
```

Liite 3: Esimerkkisovelluksen lähdekoodi (6/25)

menuwidget.cpp

```
#include "menuwidget.h"
#include "ui_menuwidget.h"

MenuWidget::MenuWidget(QWidget *parent) :
   QWidget(parent),
   ui(new Ui::MenuWidget)
{
   ui->setupUi(this);

   connect(ui->startButton, SIGNAL(clicked()), this, SLOT(slotStart()));
   connect(ui->aboutButton, SIGNAL(clicked()), this, SLOT(slotAbout()));
   connect(ui->quitButton, SIGNAL(clicked()), this, SLOT(slotQuit()));
}

MenuWidget::~MenuWidget()
{
   delete ui;
}

void MenuWidget::changeEvent(QEvent *e)
{
   QWidget::changeEvent(e);
   switch (e->type()) {
   case QEvent::LanguageChange:
      ui->retranslateUi(this);
      break;
   default:
      break;
   }
}

void MenuWidget::slotStart()
{
   emit signalStart();
   this->deleteLater();
}

void MenuWidget::slotAbout()
{
   emit signalAbout();
}

void MenuWidget::slotQuit()
{
   emit signalQuit();
   this->deleteLater();
}
```

Liite 3: Esimerkkisovelluksen lähdekoodi (7/25)

aboutwidget.h

```
#ifndef ABOUTWIDGET_H
#define ABOUTWIDGET_H

#include <QWidget>

namespace Ui {
   class AboutWidget;
}

//!  An AboutWidget class.
/*!
  Widget for about information. Inherits QWidget.
*/
class AboutWidget : public QWidget
{
   Q_OBJECT

public:

   //! A constructor.
   /*!
     Creates AboutWidget.
     \param parent a QWidget*
   */
   AboutWidget(QWidget *parent = 0);

   //! A destructor.
   /*!
     Removes AboutWidget.
   */
   ~AboutWidget();

protected:

   //! A protected member function taking one argument and returning nothing.
   /*!
     \param e a QEvent*.
   */
   void changeEvent(QEvent *e);

private:

   //! A private variable.
   /*!
     Pointer of UI.
   */
   Ui::AboutWidget *ui;
};
```

Liite 3: Esimerkkisovelluksen lähdekoodi (8/25)

```
#endif // ABOUTWIDGET_H
```

aboutwidget.cpp

```
#include "aboutwidget.h"
#include "ui_aboutwidget.h"

AboutWidget::AboutWidget(QWidget *parent) :
   QWidget(parent),
   ui(new Ui::AboutWidget)
{
   ui->setupUi(this);

   connect(ui->closeButton, SIGNAL(clicked()), this, SLOT(deleteLater()));
}

AboutWidget::~AboutWidget()
{
   delete ui;
}

void AboutWidget::changeEvent(QEvent *e)
{
   QWidget::changeEvent(e);
   switch (e->type()) {
   case QEvent::LanguageChange:
      ui->retranslateUi(this);
      break;
   default:
      break;
   }
}
```

demowidget.h

```
#ifndef DEMOWIDGET_H
#define DEMOWIDGET_H

#include <QWidget>
#include <QSvgWidget>
#include "ogrewidget.h"

namespace Ui {
   class DemoWidget;
}

//!  A DemoWidget class.
```

Liite 3: Esimerkkisovelluksen lähdekoodi (9/25)

```
/*!
  Widget for ogre demo. Inherits QWidget.
*/
class DemoWidget : public QWidget
{
    Q_OBJECT

public:

    //! A constructor.
    /*!
      Creates DemoWidget.
      \param desktopSize a QSize
      \param parent a QWidget*
    */
    DemoWidget(QSize desktopSize, QWidget *parent = 0);

    //! A destructor.
    /*!
      Removes DemoWidget.
    */
    ~DemoWidget();

protected:

    //! A protected member function taking one argument and returning nothing.
    /*!
      \param e a QEvent*.
    */
    void changeEvent(QEvent *e);

private:

    //! A private variable.
    /*!
      Pointer of UI.
    */
    Ui::DemoWidget *ui;

    //! A private variable.
    /*!
      Pointer of OgreWidget.
    */
    OgreWidget* m_ogreWidget;

private slots:

    //! A private slot function taking no arguments.
    /*!
```

Liite 3: Esimerkkisovelluksen lähdekoodi (10/25)

```
     Run after Quit-button is clicked.
   */
   void slotQuit();
signals:

   //! A signal taking no arguments.
   /*!
     emitted after Quit-button is clicked.
   */
   void signalMenu();
};

#endif // DEMOWIDGET_H
```

demowidget.cpp

```
#include "demowidget.h"
#include "ui_demowidget.h"

DemoWidget::DemoWidget(QSize desktopSize, QWidget *parent) :
   QWidget(parent),
   ui(new Ui::DemoWidget)
{
   ui->setupUi(this);
   this->resize(desktopSize);
   m_ogreWidget = new OgreWidget( ui->ogreRenderWidget );
   QString svgFile = "Data/images/logo.svg";
   ui->svgWidget->load( svgFile );

   connect(ui->quitButton, SIGNAL(clicked()), this, SLOT(slotQuit()));
   connect( this->m_ogreWidget, SIGNAL(signalFps(int)), ui->lcdNumber,
SLOT(display(int)));
}

DemoWidget::~DemoWidget()
{
   delete ui;
}

void DemoWidget::changeEvent(QEvent *e)
{
   QWidget::changeEvent(e);
   switch (e->type()) {
   case QEvent::LanguageChange:
      ui->retranslateUi(this);
      break;
   default:
      break;
```

Liite 3: Esimerkkisovelluksen lähdekoodi (11/25)

```
    }
}

void DemoWidget::slotQuit()
{
    emit signalMenu();
    this->deleteLater();
}
```

ogrewidget.h

```
#ifndef OGREWIDGET_H
#define OGREWIDGET_H

#include "QOgreWidget.h"
#include <QTime>

//!  An OgreWidget class.
/*!
  Widget for rendering Ogre content. Inherits QOgreWidget.
*/
class OgreWidget : public QOgreWidget
{
    Q_OBJECT

public:

    //! A constructor.
    /*!
      Creates OgreWidget.
      \param parent a QWidget*
    */
    OgreWidget(QWidget* parent);

    //! A destructor.
    /*!
      Removes OgreWidget.
    */
    ~OgreWidget(void);
protected:

    //! A protected member function taking one argument and returning nothing.
    /*!
      \param e a QResizeEvent*.
    */
    void resizeEvent(QResizeEvent *e);

    //! A protected member function taking one argument and returning nothing.
```

Liite 3: Esimerkkisovelluksen lähdekoodi (12/25)

```
    /*!
      \param e a QPaintEvent*.
    */
    void paintEvent(QPaintEvent *e);
    //! A protected member function taking one argument and returning nothing.
    /*!
      \param e a QTimerEvent*.
    */
    void timerEvent(QTimerEvent *e);

    //! A protected member function taking one argument and returning nothing.
    /*!
      \param e a QMouseEvent*.
    */
    void mousePressEvent(QMouseEvent *e);

    //! A protected member function taking one argument and returning nothing.
    /*!
      \param e a QMouseEvent*.
    */
    void mouseReleaseEvent(QMouseEvent *e);

    //! A protected member function taking one argument and returning nothing.
    /*!
      \param e a QMouseEvent*.
    */
    void mouseMoveEvent(QMouseEvent *e);

    //! A protected member function taking no arguments and returning nothing.
    /*!
      Creates Scene.
    */
    void createScene(void);

    //! A protected member function taking no arguments and returning nothing.
    /*!
      Setups resources.
    */
    void setupResources(void);

    //! A protected member function taking no arguments and returning nothing.
    /*!
      Setups Scene.
    */
    void setupScene(void);

private:

    //! A private variable.
```

Liite 3: Esimerkkisovelluksen lähdekoodi (13/25)

```
    /*!
      Boolean for mouse press.
    */
    bool m_mousePressed;
    //! A private variable.
    /*!
      QPoint for mouse position.
    */
    QPoint m_mousePressPos;

    //! A private variable.
    /*!
      Ogre::Quaternion for mouse press.
    */
    Ogre::Quaternion m_orientationPressed;

    //! A private variable.
    /*!
      Ogre::SceneNode
    */
    Ogre::SceneNode *m_mainNode;

    //! A private variable.
    /*!
      Ogre::SceneManager
    */
    Ogre::SceneManager *m_sceneMgr;

    //! A private variable.
    /*!
      Ogre::Camera
    */
    Ogre::Camera *m_camera;

    //! A private variable.
    /*!
      Ogre::Viewport
    */
    Ogre::Viewport *m_vp;

    //! A private static const variable.
    /*!
      float for radius
    */
    static const float m_RADIUS;

signals:

    //! A signal taking one arguments.
```

Liite 3: Esimerkkisovelluksen lähdekoodi (14/25)

```
    /*!
      \param fps an integer
      emitted after timer timeout.
    */
    void signalFps( int fps );
};

#endif // OGREWIDGET_H
```

ogrewidget.cpp

```
#include "ogrewidget.h"
#include <QMouseEvent>

///
/// Default constructor
///
OgreWidget::OgreWidget(QWidget *parent) : QOgreWidget(parent) {
                m_sceneMgr = NULL;
                m_vp = NULL;
        m_mousePressed = false;
}
///
/// Destructor
///
OgreWidget::~OgreWidget(void) {
    this->m_ogreRoot->shutdown();
    delete this->m_ogreRoot;
}
///
///
/// \name Protected functions
/// Protected helper functions
///
//@{
///
/// Handle a resize event (pass it along to the render window)
/// \param e The event data
///
void OgreWidget::resizeEvent(QResizeEvent *e) {
                QOgreWidget::resizeEvent(e);

                if (m_renderWindow) {
                                // Alter the camera aspect ratio to match the viewport
                                m_camera->setAspectRatio(Ogre::Real(width()) /
Ogre::Real(height()));
                                m_vp->update();
                }
```

Liite 3: Esimerkkisovelluksen lähdekoodi (15/25)

```
}
///
/// Handle a paint event (just render again, if needed create render window)
/// \param e The event data
///
void OgreWidget::paintEvent(QPaintEvent *) {
                if(!m_renderWindow) {
                                createRenderWindow();
                                setupResources();
                                setupScene();
                }
        update();
}
///
/// Handle a timer event
/// \param e The event data
///
void OgreWidget::timerEvent(QTimerEvent *) {
   update();

   emit signalFps( this->m_renderWindow->getLastFPS() );
}
///
/// The user pressed a mouse button, start tracking
/// \param e The event data
///
void OgreWidget::mousePressEvent(QMouseEvent *e) {
   m_mousePressPos = e->pos();
   if (m_mainNode)
      m_orientationPressed = m_mainNode->getOrientation();
   m_mousePressed = true;
}
///
/// The user released a mouse button, stop tracking
/// \param e The event data
///
void OgreWidget::mouseReleaseEvent(QMouseEvent *) {
   m_mousePressed = false;
}
///
/// The user moved the mouse, if tracking process it
/// \param e The event data
///
void OgreWidget::mouseMoveEvent(QMouseEvent *e) {
   if (m_mousePressed) {
      QPoint curPos = e->pos();

      double w = width();
      double h = height();
```

Liite 3: Esimerkkisovelluksen lähdekoodi (16/25)

```
        double curX = (curPos.x() * 2. - w) / w;
        double curY = (h - curPos.y() * 2.) / h;
        double x0 = (m_mousePressPos.x() * 2. - w) / w;
        double y0 = (h - m_mousePressPos.y() * 2.) / h;

        Ogre::Vector3 v1(x0, y0, 0);
        Ogre::Vector3 v2(curX, curY, 0);

        double radiusSqr = m_RADIUS * m_RADIUS;
        double cutoff = radiusSqr * 0.5;
        double Rho = v1[0] * v1[0] + v1[1] * v1[1];
        v1[2] = (Rho < cutoff) ? sqrt(radiusSqr - Rho) : (cutoff / sqrt(Rho));

        Rho = v2[0] * v2[0] + v2[1] * v2[1];
        v2[2] = (Rho < cutoff) ? sqrt(radiusSqr - Rho) : (cutoff / sqrt(Rho));

        // v_cross is the normal of rotating plane
        Ogre::Vector3 cross = v2.crossProduct(v1);
        cross.normalise();

        // compute the angle
        v1.normalise();
        v2.normalise();
        double cosAngle = v1.dotProduct(v2);
        if (cosAngle < -1.0)
           cosAngle = -1.0;
        else if(cosAngle > 1.0)
           cosAngle = 1.0;
        double angle = acos(cosAngle);

        m_mainNode->rotate(cross, Ogre::Radian(angle));

        m_mousePressPos = curPos;
        m_orientationPressed = m_mainNode->getOrientation();

        //updateOgre();
    }
}
///
/// Create the Ogre scene
///
void OgreWidget::createScene(void) {
                m_sceneMgr->setAmbientLight(Ogre::ColourValue(0.6, 0.6, 0.6));

                // Setup the actual scene
                Ogre::Light* l = m_sceneMgr->createLight("MainLight");
                l->setPosition(0, 100, 500);

        Ogre::Entity* robotEntity = m_sceneMgr->createEntity("Entity", "penguin.mesh");
```

Liite 3: Esimerkkisovelluksen lähdekoodi (17/25)

```
	m_mainNode = m_sceneMgr->getRootSceneNode()->createChildSceneNode();
	m_mainNode->attachObject(robotEntity);

		m_camera->setAutoTracking(true, m_mainNode);
}
///
/// Configure the resources in Ogre
/// \todo This should be moved somewhere else
///
void OgreWidget::setupResources(void) {
		// Load resource paths from config file
		Ogre::ConfigFile config;
	config.load("Data/resources.cfg");

		// Go through all sections & settings in the file
		Ogre::ConfigFile::SectionIterator it = config.getSectionIterator();

		Ogre::String secName, typeName, archName;
		while (it.hasMoreElements()) {
			secName = it.peekNextKey();
			Ogre::ConfigFile::SettingsMultiMap *settings = it.getNext();
			Ogre::ConfigFile::SettingsMultiMap::iterator i;

			for (i = settings->begin(); i != settings->end(); ++i) {
				typeName = i->first;
				archName = i->second;

		Ogre::ResourceGroupManager::getSingleton().addResourceLocation(archName,
typeName, secName);
			}
   }
}
//@}
///
/// \name Private functions
/// Private helper functions
///
//@{
///
/// Setup the scene
///
void OgreWidget::setupScene(void) {
		m_sceneMgr = m_ogreRoot->createSceneManager(Ogre::ST_GENERIC);

		// Create the camera
		m_camera = m_sceneMgr->createCamera("PlayerCam");
		m_camera->setPosition(Ogre::Vector3(0, 0, 200));

		// Look back along -Z
		m_camera->lookAt(Ogre::Vector3(0, 0, -300));
```

Liite 3: Esimerkkisovelluksen lähdekoodi (18/25)

```
            m_camera->setNearClipDistance(5);

            // Create one viewport, entire window
            m_vp = m_renderWindow->addViewport(m_camera);
            m_vp->setBackgroundColour(Ogre::ColourValue(0,0,0));
            m_vp->setClearEveryFrame(true);

            Ogre::ResourceGroupManager::getSingleton().initialiseAllResourceGroups();
            createScene();

            Ogre::MaterialManager::getSingleton().setDefaultTextureFiltering(Ogre::TFO_BI
LINEAR);
            Ogre::MaterialManager::getSingleton().setDefaultAnisotropy(1);

            // Alter the camera aspect ratio to match the viewport
            m_camera->setAspectRatio(Ogre::Real(m_vp->getActualWidth()) /
Ogre::Real(m_vp->getActualHeight()));

    startTimer(50);
}

//@}
///
/// \name Private constants
/// Private constants this class uses
///
//@{
///
/// The radius used for rotating
///
const float OgreWidget::m_RADIUS = 0.8;
//@}
//
//
```

QOgreWidget.h

```
#ifndef QOgreWidget_H
#define QOgreWidget_H

#include <QWidget>
#include <OGRE/Ogre.h>

#if defined(Q_WS_MAC)
typedef struct __AGLContextRec  *AGLContext;
#endif

//!  A QOgreWidget class.
```

Liite 3: Esimerkkisovelluksen lähdekoodi (19/25)

```
/*!
  This Class is inherited by ogrewidget which handles ogre engine.
  Inherits QWidget.
*/
class QOgreWidget : public QWidget
{
    Q_OBJECT

public:

    //! A constructor.
    /*!
      Creates QOgreWidget.
      \param parent a QWidget*
    */
    QOgreWidget(QWidget* parent);

    //! A destructor.
    /*!
      Removes QOgreWidget.
    */
    ~QOgreWidget(void);

    //! A public virtual function taking no arguments and returning nothing.
    /*!
      Updates Ogre renderwindow
    */
    virtual void update(void);

    //! A public member function taking no arguments and returning a QSize.
    /*!
      \return minimumsize a Qsize.
    */
    QSize minimumSizeHint(void) const;

protected:

    //! A protected member function taking one argument and returning nothing.
    /*!
      \param e a QPaintEvent*.
    */
    void paintEvent(QPaintEvent *e);

    //! A protected member function taking one argument and returning nothing.
    /*!
      \param e a QResizeEvent*.
    */
    void resizeEvent(QResizeEvent *e);
```

Liite 3: Esimerkkisovelluksen lähdekoodi (20/25)

```
    //! A protected pure virtual function taking no arguments and returning nothing.
    /*!
      Creates Scene.
    */
    virtual void createScene(void) = 0;

    //! A protected pure virtual function taking no arguments and returning nothing.
    /*!
      Setups resources.
    */
    virtual void setupResources(void) = 0;

    //! A protected virtual function taking no arguments and returning nothing.
    /*!
      Configures Ogre engine.
    */
    virtual void configure(void);

    //! A protected member function taking no arguments and returning nothing.
    /*!
      Creates Ogre renderwindow.
    */
    void createRenderWindow(void);

    //! A protected member function taking no arguments and returning nothing.
    /*!
      Resizes Ogre renderwindow.
    */
    void resizeRenderWindow(void);

    //! A protected variable.
    /*!
      Pointer of Ogre renderwindow.
    */
    Ogre::RenderWindow *m_renderWindow;

    //! A protected static variable.
    /*!
      Pointer of OgreRoot. Singleton. Must be deleted before creation of next one.
    */
    static Ogre::Root *m_ogreRoot;

#if defined(Q_WS_MAC)
    AGLContext m_aglContext;
#endif
};

#endif
```

Liite 3: Esimerkkisovelluksen lähdekoodi (21/25)

QOgreWidget.cpp

```
#include "QOgreWidget.h"
#if defined(Q_WS_MAC)
#include <Carbon/Carbon.h>
#include <Ogre/OgreOSXContext.h>
#include <AGL/agl.h>
#elif !defined(Q_WS_WIN)
#include <QX11Info>
#endif

///
/// Default constructor
///
QOgreWidget::QOgreWidget(QWidget *parent) : QWidget(parent) {
		setAttribute(Qt::WA_PaintOnScreen);
		setAttribute(Qt::WA_NoBackground);

		m_renderWindow = NULL;
		m_ogreRoot = NULL;
}
///
/// Destructor
///
QOgreWidget::~QOgreWidget(void) {
}
///
/// \name Misc
/// Miscellaneous functions of this class
///
//@{
///
/// Render a frame
///
void QOgreWidget::update(void) {
		if (m_renderWindow) {
				m_ogreRoot->_fireFrameStarted();
				m_renderWindow->update();
				m_ogreRoot->_fireFrameEnded();
		}
}
///
/// Give the minimum size for this widget
/// \return Returns a size
/// \warning Needs to be big enough so that ogre doesn't crash (50 50 looks big enough)
///
QSize QOgreWidget::minimumSizeHint(void) const {
		return QSize(50, 50);
}
```

Liite 3: Esimerkkisovelluksen lähdekoodi (22/25)

```
//@}
///
/// \name Protected functions
/// Protected helper functions
///
//@{
///
/// Handle a resize event (pass it along to the render window)
/// \param e The event data
///
void QOgreWidget::resizeEvent(QResizeEvent *) {
		if (m_renderWindow)
				resizeRenderWindow();
}
///
/// Handle a paint event (just render again, if needed create render window)
/// \param e The event data
///
void QOgreWidget::paintEvent(QPaintEvent *) {
		if (!m_renderWindow)
				createRenderWindow();

		update();
}
//@}
///
/// \name Private functions
/// Private helper functions
///
//@{
///
/// Create the Ogre render window
///
void QOgreWidget::createRenderWindow(void) {
		Ogre::NameValuePairList params;

		if (m_renderWindow)
				return;
		if (!m_ogreRoot)
				configure();

#if defined(Q_WS_MAC) || defined(Q_WS_WIN)
		params["externalWindowHandle"] = Ogre::StringConverter::toString((size_t) wi-
nId());
#else
   QX11Info info = x11Info();
		Ogre::String winHandle;
		winHandle  = Ogre::StringConverter::toString((unsigned long)(info.display()));
		winHandle += ":";
```

Liite 3: Esimerkkisovelluksen lähdekoodi (23/25)

```
                winHandle += Ogre::StringConverter::toString((unsigned int)(info.screen()));
                winHandle += ":";
                winHandle += Ogre::StringConverter::toString((unsigned long)(this-
>parentWidget()->winId()));
                params["parentWindowHandle"] = winHandle;
#endif

                m_renderWindow = m_ogreRoot->createRenderWindow("View" +
Ogre::StringConverter::toString((unsigned long) this),
        this->parentWidget()->width(), this->parentWidget()->height(), false, &params);

#if defined(Q_WS_MAC)
                // store context for hack
                Ogre::OSXContext *context;
                m_renderWindow->getCustomAttribute("GLCONTEXT", &context);
                context->setCurrent();
                m_aglContext = aglGetCurrentContext();
                resizeRenderWindow();
#endif

                // take over ogre window
#if !defined(Q_WS_MAC)
                WId ogreWinId = 0x0;
                m_renderWindow->getCustomAttribute("WINDOW", &ogreWinId);
                assert(ogreWinId);
                create(ogreWinId);
#endif
}
///
/// Configure Ogre
/// \todo This really needs to be moved somewhere else (singleton?)
///
void QOgreWidget::configure(void) {
                if (m_ogreRoot)
                                return;

      m_ogreRoot = new Ogre::Root("Data/plugins.cfg", "Data/ogre.cfg", "Data/ogre.log");

                if (!m_ogreRoot->restoreConfig()) {
                                // setup a renderer
                                const Ogre::RenderSystemList renderers = m_ogreRoot-
>getAvailableRenderers();
                                assert(!renderers.empty()); // we need at least one renderer to do any-
thing useful

                                //Ogre::RenderSystem *renderSystem = chooseRenderer(renderers);
                                Ogre::RenderSystem *renderSystem = *renderers.begin();
                                assert(renderSystem); // user might pass back a null renderer, which
would be bad!
```

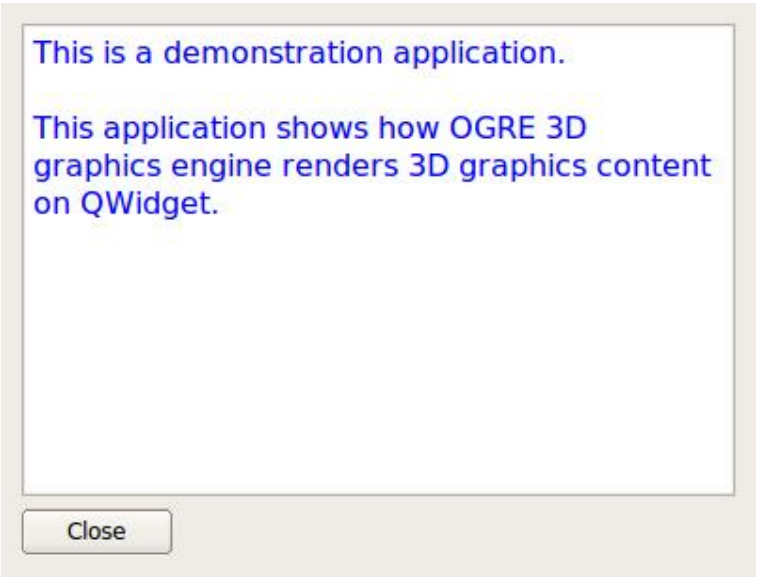
Liite 3: Esimerkkisovelluksen lähdekoodi (24/25)

```
                        m_ogreRoot->setRenderSystem(renderSystem);
                        QString dimensions = QString("%1x%2").arg(this->width()).arg(this-
>height());
                        renderSystem->setConfigOption("Video Mode", dimen-
sions.toStdString());

                        // initialize without creating window
                        m_ogreRoot->getRenderSystem()->setConfigOption("Full Screen",
"No");
                        m_ogreRoot->saveConfig();
                }
                m_ogreRoot->initialise(false);
}
///
/// Resize the render window (when the widget was resized)
///
void QOgreWidget::resizeRenderWindow(void) {
                if (!m_renderWindow)
                        return;

#if !defined(Q_WS_MAC)
                m_renderWindow->resize(width(), height());
                m_renderWindow->windowMovedOrResized();
#else
                GLint bufferRect[4];
                HIViewRef mView = HIViewRef(winId());

                m_renderWindow->windowMovedOrResized();

                // reposition our drawing region
                HIRect viewBounds, winBounds;
                HIViewGetBounds(mView, &viewBounds);
                HIViewRef root = HIViewGetRoot(HIViewGetWindow(mView));
                HIViewRef content_root;
                HIViewFindByID(root, kHIViewWindowContentID, &content_root);

                HIViewGetBounds(content_root, &winBounds);
                HIViewConvertRect(&viewBounds, mView, content_root);

                bufferRect[0] = x();
                bufferRect[1] = GLint((winBounds.size.height) - (viewBounds.origin.y + view-
Bounds.size.height));
                bufferRect[2] = width();
                bufferRect[3] = height();

                aglSetInteger(m_aglContext, AGL_BUFFER_RECT, bufferRect);
                aglEnable (m_aglContext, AGL_BUFFER_RECT);
#endif
}
```

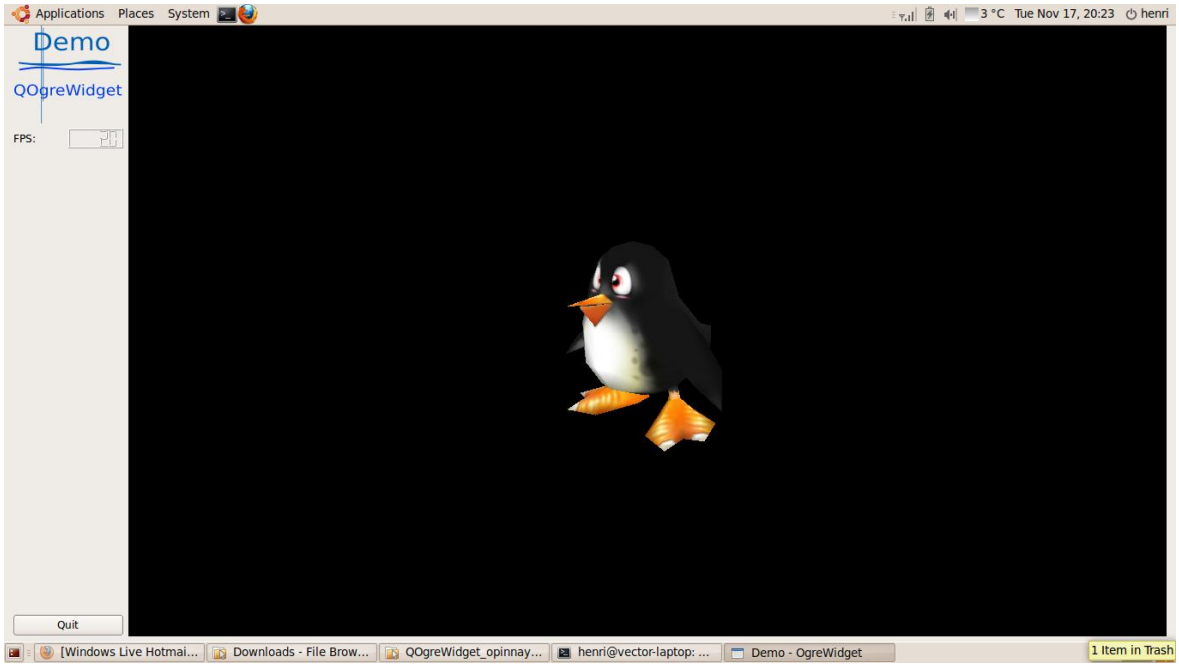
Liite 3: Esimerkkisovelluksen lähdekoodi (25/25)

```
//@}
///
/// \name Protected variables
/// Protected variables this class uses
///
//@{
///
/// The Ogre root
///
Ogre::Root * QOgreWidget::m_ogreRoot = NULL;
//@}
//
//
```

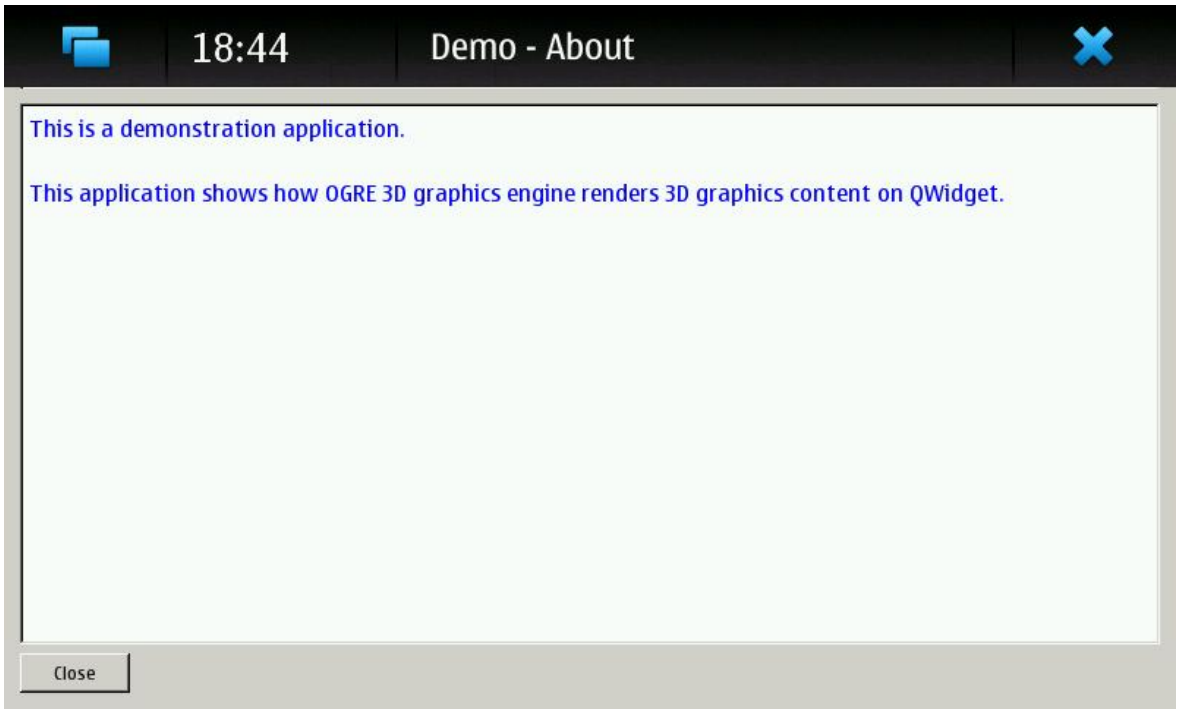
Liite 4: Esimerkkisovelluksen ulkoasu Ubuntussa (1/2)



Liite 4: Esimerkkisovelluksen ulkoasu Ubuntussa (2/2)



Liite 5: Esimerkkisovelluksen ulkoasu Maemo-emulaattorissa (1/2)



Liite 5: Esimerkkisovelluksen ulkoasu Maemo-emulaattorissa (2/2)