JÄRJESTELMÄRIIPPUMATTOMAN SOVELLUKSEN TOTEUTTAMINEN C++-KIELELLÄ QT-KEHYSYMPÄRISTÖSSÄ

Samu Juvonen

Pro gradu -tutkielma

Tietojenkäsittelytieteen laitos

Tietojenkäsittelytiede

Toukokuu 2012

ITÄ-SUOMEN YLIOPISTO, Luonnontieteiden ja metsätieteiden tiedekunta, Kuopio

Tietojenkäsittelytieteen laitos
Tietojenkäsittelytiede

Juvonen, Samu: Järjestelmäriippumattoman sovelluksen toteuttaminen C++-kielellä Qt-kehysympäristössä
Pro gradu -tutkielma, 44 s.
Pro gradu -tutkielman ohjaaja: FT Marko Hassinen
Toukokuu 2012

Avainsanat: Qt, C++, järjestelmäriippumattomuus, siirrettävyys, ohjelmointi

Tutkielmassa perehdytään C++-sovelluskehityksen ja sovellusten siirrettävyyden helpottamiseen Qt-kehysympäristöä käyttäen. Qt on avoimen lähdekoodin projekti, joka on alun perin norjalaisen Trolltech-yrityksen kehittämä ja sittemmin siirtynyt Suomen ylpeyden, Nokian, omistukseen. Vaikka Nokian alkuperäiset suunnitelmat Qt:n maailmanvalloituksen aloittamisesta mobiilipuolella kariutuivatkin yrityksen lopetettua MeeGo-mobiilialustansa kehittämisen ja loikattua Windows-käyttöjärjestelmiin matkapuhelimissaan, Qt:lla on historiallisesti vahva asema erityisesti avoimen lähdekoodin projekteissa, mutta siihen on kohdistunut kiinnostusta myös kaupallisesta näkökulmasta.

Käytännönosuutena palvelee tutkielman kirjoittajan henkilökohtainen harrasteprojekti, joka on kehitetty Linux-ympäristössä, mutta Qt:n ansiosta sen voi vähäisin muutoksin saattaa toimimaan myös mm. Windows-käyttöjärjestelmällä.

UNIVERSITY OF EASTERN, Faculty of Science and Forestry, Kuopio
School of Computing
Computer Science

Juvonen, Samu: Creating a platform-independent application in C++ with Qt
Master's Thesis, 44 p.
Supervisor of the Master's Thesis: PhD Marko Hassinen
May 2012

Keywords: Qt, C++, platform-independence, programming

This thesis describes the problems of creating portable software in C++ language and how Qt can be used to make the task easier. Qt is an open source development platform for platform-independent applications that makes software development with C++ faster and easier. It was initially developed by a Norwegian company called Trolltech, who were later acquired by one of the world's greatest mobile phone manufacturers, Nokia.

Even though Nokia's initial plans of making Qt it's jewel in mobile systems development on MeeGo were scrapped in favour of Windows Phone operating system, Qt still has a strong stance especially in the field of open source development. Qt also enjoys some interest in the business side of application development.

The practical section of this thesis consists of the author's hobby project called Monolake. It's an open source application mainly targeting Linux but also made to run on Windows for the sake of this thesis.

Esipuhe

Tämän tutkielman aiheeksi valikoitui Qt-ohjelmointi siksi, että olen ollut jo pitkään kiinnostunut järjestelmäriippumattomien sovellusten kehittämisestä. Aloitin ohjelmoinnin jo lukiossa, jolloin löysin PHP-kielen, joka on hyvin käytetty internet-palveluiden puolella. Yliopistossa opiskellessani laajensin repertuaariani työpöytäsovellusten puolelle, jolloin kiinnostuin Qt-kehysympäristöstä.

Merkittävimmät syyt tähän olivat Qt:n avoimuus sekä laaja suosio ja tuki Linuxilla, joka on ollut pääasiallinen käyttöjärjestelmäni jo hieman yli viisi vuotta. Lisäksi samojen sovellusten helpohko siirtäminen Windows- ja Mac-alustoille kiehtoi. Qt on myös ominaisuuksiltaan hyvin kattava, joten sovelluksia ei tarvitse kasata pala palalta useista eri kirjastoista, joiden yhteensovittaminen voisi vaatia lisätyötä, ja jotka eivät välttämättä ole yhtä helposti siirrettävissä käyttöjärjestelmästä toiseen.

Lisäksi olin jo aiemmin valinnut erikoistyöni aiheeksi erään kehittämäni Qt-sovelluksen, mutta kiireistä johtuen se ei loppujen lopuksi edennyt aikataulussaan ja päätin lykätä kyseisen projektin loppuunsaattamista. Sovelluksella on kaikesta huolimatta roolinsa tässäkin tutkielmassa.

Kuopiossa 09.05.12

_____

Samu Juvonen

Käsitteet ja lyhenteet

C++	Ohjelmointikieli
XML	Extensible Markup Language
Widget	Qt:ssä käytetty termi graafisen käyttöliittymän komponentille
KDE	Avoimen lähdekoodin työpöytäympäristö Unix-järjestelmille
Gnome	Avoimen lähdekoodin työpöytäympäristö Unix-järjestelmille
Sovelluskehys	Runko, jonka päälle tietokoneohjelmia voi kehittää.

Sisällysluettelo

1 JOHDANTO

Qt on avoimen lähdekoodin kehysympäristö C++-kielelle. Sen avulla voidaan kehittää sovelluksia, jotka ovat helposti siirrettävissä eri käyttöjärjestelmille vähäisin muutoksin lähdekoodiin. Qt:lla on takanaan jo 20 vuoden historia, jonka aikana sovelluskehys on kasvanut tärkeäksi tekijäksi erityisesti Linux-ympäristössä, mutta on myös herättänyt jonkin verran kiinnostusta kaupallisessakin mielessä. Qt-kehysympäristöä käyttäen on toteutettu mm. Google Earth, Opera-internetselain ja Wolfram Researchin Mathematica-matematiikkaohjelmisto.

Tässä tutkielmassa käydään aluksi luvussa 2 läpi ohjelmistojen siirrettävyyttä teoreettiselta kannalta. Luvussa 3 perehdytään Qt-kehysympäristön ominaisuuksiin. Luvussa 4 perehdytään Qt-sovelluksen toteuttamiseen ja erään esimerkkisovelluksen ominaisuuksiin. Luvussa 5 käydään läpi varteenotettavia vaihtoehtoja Qt-viitekehykselle.

2 SIIRRETTÄVYYDESTÄ

Tässä luvussa käsitellään ohjelmistojen käyttöjärjestelmästä toiseen ja jopa prosessoriarkkitehtuurista toiseen siirrettävyyden ongelmia. Ongelmiin perehdytään erityisesti C++-kielen näkökulmasta, sillä se on keskeisin kieli tässä tutkielmassa.

2.1 Määritelmä

Siirrettävyys tarkoittaa ohjelmiston saattamista toimivaksi uudessa ympäristössä olemassaolevan toteutuksen pohjalta. [Moo12] Siirrettävyyttä voidaan mitata esimerkiksi sillä, kuinka suuri osa ohjelmistosta voidaan siirtää uuteen järjestelmään siten, että prosessin kustannukset pysyvät halvempina kuin sovelluksen suunnittelu uutta ympäristöä varten olisi maksanut. Ohjelmisto olisi täydellisesti siirrettävissä, jos siirtämisen prosessi ei tuottaisi uusia kuluja. Tätä ei voida koskaan saavuttaa käytännön tasolla. [Moo12]

2.2 Siirrettävyyden ongelmat

Useimmiten siirrettävyyden muodoiksi mielletään binäärin siirrettävyys ja lähdekoodin siirrettävyys. Binääri tarkoittaa lähdekoodista käännettyä konekielistä dataa. Binäärin siirtäminen uuteen ympäristöön onnistuu vain hyvin samankaltaisten järjestelmien välillä. Lähdekoodin siirtäminen taas vaatii pääsyn käsiksi lähdekoodiin, mutta mahdollisuudet ohjelmiston siirtämiseen ovat paremmat. [Moo12]

Siirrettävää ohjelmistoa suunnitellessa tärkein tekijä on kysymys siitä, kuinka paljon vaivaa on kannattavaa nähdä sovelluksen suunnittelussa, jotta lopputuote olisi siirrettävissä tulevaisuudessa. Erityisesti seuraavat kysymykset vaativat vastauksen: [Moo12]

1. Mihin ympäristöihin sovellus halutaan tulevaisuudessa siirtää?
2. Kuinka suuri siirrettävyyden taso halutaan saavuttaa?
3. Kuinka paljon lisäkuluja näiden tavoitteiden täyttäminen saa aiheuttaa?
4. Kuinka suuri heikennys sovelluksen laadussa on hyväksyttävää, jotta haluttu siirrettävyyden taso saavutetaan?

Siirtämisprosessia suunnitellessa tulee määritellä, kuinka ja missä määrin senhetkisen toteutuksen siirrettävyyttä voidaan hyödyntää uudessa toteutuksessa, eli mitkä komponentit voidaan siirtää ja mitkä tulee kehittää uusiksi.

Siirrettävyyttä heikentäville tekijöille on useita syitä. Esimerkiksi ohjelmointikielen ominaisuudet vaikuttavat olennaisesti sillä kirjoitetun ohjelman siirrettävyyteen. Erillisessä komentotulkissa tai virtuaalikoneessa ajettavat ohjelmat toimivat periaatteessa suoraan toisessa käyttöympäristössä, kunhan vain sen vaatima komentotulkki (tai virtuaalikone) on saatavilla kyseiseen ympäristöön.

Yleensä käytetyt ohjelmointikielet kuitenkin edellyttävät lähdekoodin kääntämisen suoraan konekielisiksi ohjelmiksi. Tällöin ne tulevat tiukasti sidotuiksi siihen ympäristöön, jossa ne on käännettykin. Ohjelmakoodin voi toki kääntää toisessakin ympäristössä, mutta tällöin täytyy varmistua siitä, että kaikki käytetyt kolmannen osapuolen ohjelmakirjastot ja rajapinnat ovat myös saatavilla uudessa ympäristössä.

Kaikki ei ole kuitenkaan kiinni pelkästään käyttöjärjestelmästä, sillä jopa sama käyttöjärjestelmä voi asettaa erilaisia vaatimuksia, jos esimerkiksi vaihdetaan prosessorin arkkitehtuuria. Nykypäivänä tavalliset työ- ja kotikoneet ovat lähestulkoon aina Intelin x86-arkkitehtuurin prosessorilla varustettuja, joten ongelma ei vaikuta työpöytäsovelluksia ajatellen mahdottomalta. Omat haasteensa välillä asettaa kuitenkin se, että vaikka merkittävä osa näistä suorittimista on edelleen 32-bittisiä, 64-bittiset suorittimet kasvattavat koko ajan osuuttaan. ”Bittisyys” vaikuttaa erityisesti muistinkäsittelyyn, sillä 32-bittisen muistiavaruuden rajat saavutetaan joissakin tilanteissa. [Moo12, Wik12c]

2.3 C++-standardin epämääräisyydet

C++ on standardoitu kieli ja saatavilla lähestulkoon mihin ympäristöön tahansa. Se kuuluu käännettävien kielten joukkoon, mikä tarkoittaa sitä, että tekstimuotoinen lähdekoodi muunnetaan binäärimuotoiseksi ohjelmaksi käyttämällä ohjelmakoodin kääntäjää. Jokainen kääntäjä on itse asiassa oma implementaationsa C++-kielestä. [Obi12] Vaikka C++ on standardoitu kielenä, niin samasta lähdekoodista voi syntyä hyvinkin erilaisia ohjelmia, kun kääntäjä vaihdetaan. Erilaiset ”tulkinnat” voivat johtua moninaisista syistä.

C++-standardiin on jätetty joitakin tarkoituksellisia aukkoja, jotta kieli toimisi mahdollisimman tehokkaasti erilaisissa ympäristöissä. Joissakin tapauksissa on määritelty vain sallittu arvovalue, joissain tapauksissa on määritelty vain löyhät reunaehdot, mutta tarkempi toteutus voi olla mitä tahansa. Lisäksi kaikissa tapauksissa toiminnalle ei ole asetettu lainkaan rajoituksia. [Moo12]

2.3.1 Perustietotyyppien kokoa ei ole määritelty yksikäsitteisesti

Eräs konkreettinen esimerkki on perustietotyyppien yksikäsitteisen määrittelyn puute. Esimerkiksi eri kokonaislukutyypeille on periaatteessa määritelty vain arvoalueen alaraja ja tyyppien keskinäinen järjestys. Todellinen alkion koko voi riippua niin käytetystä kääntäjästä kuin käyttöjärjestelmästäkin. Arvoalueen ohella kyse on myös siitä, kuinka paljon muistia muuttujat vievät. ”Tavallinen kokonaisluku”, integer, on määritelty olemaan ”ajoympäristön luonnollinen koko” ja char-tyypin tulee pystyä esittämään kaikki ympäristön perusmerkistön merkit. [Bec05]

Koska ohjelmoijat kuitenkin tarvitsevat vakiokokoisia tietotyyppejä, saattavat kääntäjät tarjota standardin ulkopuolelta omia tietotyyppejään. Näiden käyttäminen kuitenkin sitoo ohjelmakoodin tiettyyn kääntäjään, jolloin ohjelma ei ole edes siirrettävissä kääntäjästä toiseen, vaikka käyttöjärjestelmä pysyisi samana. [Moo12]

Tietotyyppien vaihteleva koko tulee selkeästi esille, kun vaihdetaan 32-bittisestä käyttöjärjestelmästä toiseen. Esimerkiksi 32-bittisessä Linux-käyttöjärjestelmässä

long-tyyppinen kokonaisluku on kooltaan neljä tavua, mutta 64-bittiseen Linuxissa sen koko onkin kahdeksan tavua.

```
#include <iostream>
#include <limits>
int main() {
        const long a = std::numeric_limits<long>::max();
        std::cout << a << " " << sizeof(a) << std::endl;
        return 0;
}
```

Kaava 1: Long-tietotyypin enimmäisarvon tulostus

2.3.2 Määrittelemätön käyttäytyminen

Joissakin tapauksissa käyttäytymiselle ei ole määritelty kovinkaan tarkkoja ehtoja. Esimerkiksi funktiokutsulle annettujen parametrien käsittelyjärjestys on määrittelemätön. Tällöin näennäisen yksinkertaisetkin asiat voivat tuottaa hyvinkin erilaisia mutta perusteltavissa olevia tuloksia. [Moo12]

Esimerkiksi funktiokutsulle syötettyjen parametrien käsittelyjärjestys on määrittelemätön:

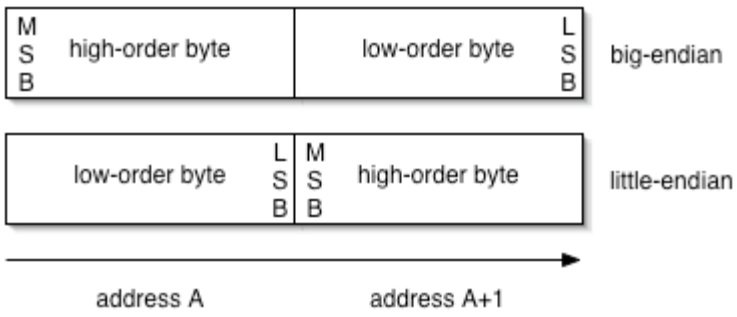
```
void f(int a, int b, int c) {
        cout << "a = " << a << ", b = " << b << ", c = " << c << endl;
}
int main(int argc, char** argv) {
        int i = 0;
        f(i++, i++, i++);
        return 0;
}
```

Kaava 2: Esimerkki funktiokutsun parametrien käsittelyjärjestyksestä [Moo12]

Kun ylläoleva esimerkki ajetaan, f-funktio voi standardin puitteissa tulostaa niin kolmikon 0, 1, 2 kuin myös 0, 0, 0.

2.4 Tavujärjestys

Eräs kiinnostava ongelma on tavujärjestys eri järjestelmissä. Ongelma tulee esille erityisesti tallentaessa tietoa levylle tai kun sitä välitetään verkon yli eri järjestelmien välillä. Monitavuisten alkioiden tavut voidaan käsitellä kahdessa eri järjestyksessä, joko eniten merkitsevä tavu ensimmäisenä (*big-endian*) tai viimeisenä (*little-endian*). Koska tavujärjestys riippuu prosessoriarkkitehtuurista eikä niinkään käyttöjärjestelmästä, voi ongelma konkretisoitua hyvinkin herkästi. [Moo12, Wik12d]



Kuva 1: Little- ja big-endianin ero [Moo12]

2.5 Järjestelmäkohtaiset rajapinnat

C++-standardi ei käsittele esimerkiksi verkkoyhteyksiä tai graafisten käyttöliittymiä laisinkaan. Tällöin täytyy turvautua järjestelmän omiin rajapintoihin. Lisäksi käyttöympäristöillä on omat ominaispiirteensä, joiden tukeminen parantaa integraatiota kyseisellä alustalla, mikä on tärkeä osa hyvää käytettävyyttä.

Lisäksi eri käyttöympäristöillä on omat suunnittelulliset ohjeistuksensa, *human interface guidelines*, joilla pyritään yhtenäistämään eri toimittajien tekemien sovellusten käyttökokemukset käyttöympäristön natiivien sovellusten mukaisiksi. Ohjeistukset voivat kattaa niin painikkeiden ja muiden interaktiivisten komponenttien sijainnin kuin visuaalisen ilmeen ja tavan reagoida määrättyihin toimintoihin. Myös niinkin perusteelliset seikat kuten kuvakkeiden visuaaliset suunnitteluperiaatteet, esimerkiksi perspektiivi ja kontrastierot, voivat kuulua määrityksiin. [Wik12a]

Käyttöliittymäsuunnittelu ei välttämättä kuulu siirrettävyyden kategoriaan eikä ole varsinaisesti tekninen ongelma, mutta siihen liittyvät asiat on tärkeä huomioida, kun sovellusta ollaan siirtämässä uuteen ympäristöön. [Wik12a]

3 QT-KEHYSYMPÄRISTÖ

Qt on avoimen lähdekoodin kehysympäristö työpöytäsovellusten kehittämiseen C++-kielellä. Sille on tehty rajapintoja myös lukuisille muille kielille, joista tärkeimpänä luultavasti Python, jolle on saatavilla useampikin kilpaileva toteutus Qt:sta.

Qt:n erikoisuus on sen monikirjoinen tuki eri käyttöjärjestelmille ja prosessoriarkkitehtuureille. Virallisesti tuetut käyttöjärjestelmät ovat uusimmat Windows-käyttöjärjestelmät, Mac OS X sekä Ubuntu Linux. [QtP11c] C++-kielellä kirjoitettu, Qt-kirjastoja käyttävä sovellus voidaan hyvin helposti siirtää mihin tahansa toiseen käyttöjärjestelmään vähäisin lähdekoodiin tehtävin muutoksin. Eräs avaintekijä on se, että Qt on varsin laaja ominaisuuksiltaan, joten tarve lisäkirjastoille on vähäinen. [BlS08]

Qt tarjoaa tuen niin verkkotoiminnoille kuin tietokannoille ja jopa äänen ja videoiden toistamiseksi. Lisäksi Qt:iin on integroitu Applen kehittämä avoimen lähdekoodin WebKit-selainmoottori ja alkujaan KDE-työpöytäympäristöön kehitetty multimediarajapinta Phonon. Qt:n avulla voidaan luoda myös graafisia käyttöliittymiä, jotka mukautuvat käyttöympäristöönsä ja luovat vaikutelman natiivista sovelluksesta.

Tämän luvun sisältö perustuu paljolti kirjaan C++ GUI Programming with Qt 4 [BlS08].

3.1 Historiaa

Qt:n kehitti norjalainen Trolltech-niminen yritys, jonka perustivat Haavard Nord ja Eirik Chambe-Eng. Alkunsa Qt sai Haavardin visiosta olio-ohjelmoinnin paradigmaa noudattavasta näyttöjärjestelmästä (display system). Ensimmäiset luokat kirjoitettiin vuonna 1991 ja ensimmäinen versio, Qt 0.90, julkaistiin vuonna 1995. Qt-nimen jälkimmäinen kirjain tulee englannin kielen sanasta toolkit. Q-kirjain päätyi nimeen

siksi, että se näytti Haavardin mielestä kauniilta hänen tekstieditorissaan. ”Qt” lausutaan kuten englannin kielen sana *cute* eli söpö. [BlS08]

Qt tuki heti ensimmäisestä julkaisustaan saakka Windows- ja Unix-järjestelmiä. Lisensointimalleja oli myös kaksi: avoimen lähdekoodin lisenssi avoimeen sovelluskehitykseen ja kaupallinen lisenssi yksityiseen kehitykseen. Versio 1.0 julkaistiin vuoden 1996 syksyllä ja saman vuoden lopulla sitä käytti jo kahdeksan kaupallisen lisenssin hankkinutta asiakasyritystä. Samana vuonna perustettiin myös KDE-projekti, jonka tarkoituksena oli tuottaa Qt-pohjainen avoimen lähdekoodin työpöytäympäristö Unix-järjestelmille. KDE:stä onkin tullut erittäin suosittu ja esimerkiksi noin 50 % Linux-työasemista on varustettu KDE-työpöydillä. [BlS08]

Version 2.0 julkaisun yhteydessä lanseerattiin myös uusi avoimen lähdekoodin lisenssin määritelmän täyttävä lisenssi, *Q Public License* (QPL). Windowsille Qt:a ei kuitenkaan avoimella lisenssillä julkaistu, ja päätöstä perusteltiin sillä, ettei Windows itse ollut avoin alusta. Käyttöjärjestelmätuki laajeni version 3.0 myötä, jolloin myös Mac OS X oli liitetty tuettujen järjestelmien listalle. [BlS08]

KDE-projektin myötä huolet Qt:n avoimuudesta kasvoivat, sillä kaikkien tahojen mielestä QPL ei ollut lisenssinä riittävän avoin eikä taannut Qt:n pysymistä avoimena lähdekoodina tulevaisuudessakin. Nämä huolet kuitenkin osoittautuivat turhiksi, kun Qt:n Unix-versiot julkaistiin GNU General Public Licensen -lisenssin (GPL) alla vuonna 2000. GPL on yksi käytetyimmistä avoimen lähdekoodin lisensseistä. Lisäksi KDE-projektin kanssa solmittiin erityinen *KDE Free Qt* -sopimus, jonka mukaan KDE-yhteisö on vapaa julkaisemaan Qt:n GPL:ääkin sallivammalla BSD-tyylisellä lisenssillä, mikäli Trolltech lopettaisi Qt:n kehitystyön. [Wik12e]

Kesällä 2005 näki päivänvalon nykyinen Qt 4. Tämän julkaisun myötä Qt tuli saataville avoimella GPL-lisenssillä kaikille tuetuille ympäristöille. Qt 4 sisältää yli 500 luokkaa ja 9000 funktiota ja se on kasvanut erittäin kattavaksi sovelluskehykseksi. [BlS08]

Vuoden 2008 alussa Nokia ilmoitti ostaneensa Trolltechin ja Qt:n. Sen tavoitteena oli tehdä Qt:sta mobiilikäyttöjärjestelmiensä de facto -kehitysympäristö. Qt julkaistiin

sekä Symbianille että uudelle MeeGo-mobiilikäyttöjärjestelmälle. Nämä suunnitelmat kuitenkin kariutuivat sen jälkeen, kun Nokia julkisti uudet suunnitelmansa Symbianin ja MeeGon alasajosta vuoden 2011 keväällä. Tämän jälkeen Qt:n kehitystyö siirrettiin Qt Project -yhteisölle, jossa Nokia tosin on mukana merkittävimpänä yhteistyökumppanina. [BlS08, Pau08, Wik12c]

3.2 Qt-moduulit

Qt on laajuutensa takia jaettu moduuleihin, joita sovelluskehittäjä voi tarpeen mukaan liittää Qt-projektiinsa.

3.2.1 QtCore - Qt:n perusta

QtCore-moduuli sisältää ydintoiminnot, jotka Qt-sovellus tarvitsee toimiakseen. Se ei kuitenkaan sisällä graafisen käyttöliittymän omaavan sovelluksen vaatimia lisätoimintoja tai -ominaisuuksia. QtCore-moduulin perusluokat ovat *QCoreApplication* ja *QObject*.

QCoreApplication on Qt-sovelluksen sydän. Jokaisella sovelluksella on yksi instanssi QCoreApplication-luokasta tai sen graafisen ympäristön vastineesta *QApplication*-luokasta. Se käynnistää Qt:n alijärjestelmän, jota kutsutaan *tapahtumasilmukaksi* (*event loop*). Tapahtumat ovat matalan tason kutsuja, joilla Qt-oliot voivat reagoida sovelluksessa tapahtuviin muutoksiin.

QObject on perusluokka, josta kaikki muut Qt-olioluokat periytyvät. Se on Qt:n oliomallin ydinluokka. Oliomallissa on kaksi keskeistä perusideaa: muistinhallinnan helpottaminen ja automatisointi vanhempi-lapsi-paradigman avulla sekä Qt-olioiden keskinäinen vuorovaikuttaminen signals and slots -ohjelmointimallin avulla. [BlS08]

3.2.2 QtGui - graafisten käyttöliittymien ydin

QtGui-moduuli laajentaa QtCorea tuomalla mukaan graafisten käyttöliittymien vaatimia ominaisuuksia. QApplication laajentaa QCoreApplication-luokan toiminnallisuutta ja *QWidget* vastaavasti QObject-luokan toimintoja.

QApplication mukauttaa sovelluksen käyttäjän työpöytäympäristön asetuksiin. Tällaisia ovat esimerkiksi väripaletti, joka määrä esimerkiksi tekstin värin tai näkymien taustavärin, sovelluksen kirjasintyylin eli fontin tai vaikkapa kaksoisklikkaukseksi tulkittavien klikkausten välisen intervallin. Lisäksi se

mahdollistaa session hallinnan eli sovellusten tilan tallentamisen ja palauttamisen esimerkiksi uloskirjautumisen yhteydessä, mikäli käyttöjärjestelmä tukee sellaista.

QWidget periytyy QObject-luokasta ja toimii perustana kaikille Qt-sovelluksen käyttöliittymäkomponenteille. Se voi vastaanottaa hiiren ja näppäimistön syötettä kuten kursorin liikkeen ja näppäinten painallukset. Jokainen QWidget-olio eli widgetti myös vastaa sisältönsä piirtämisestä ruudulle.

Kaikki käyttöliittymän komponentit, kuten esimerkiksi painikkeet, työkalurivit tai jopa ikkunan vierityspalkit ja ikkunat itse ovat QWidget-olioita eli *widgettejä*. Ikkunat ovat käytännössä vain widgettejä, joille ei ole määritetty vanhempaa. Jokainen QWidget-olio voi olla vanhempi mille tahansa muulle QWidgetille, joilloin sen lapsielementit sijoitetaan sen vaaramalle visuaaliselle alueelle.

Widgettien sijoittelua vanhempansa sisällä hallitaan asettelumalleilla (*layouts*), jotka hallitsevat widgettien geometrisia ominaisuuksia eli kokoa ja sijaintia. Asettelumallit ovat yleinen ominaisuus muissakin graafisten käyttöliittymien tekoon tarkoitetuissa kehysympäristöissä. [BlS08]

3.2.3 Muita hyödyllisiä moduuleita

Pelkkä mahdollisuus näyttää ikkunoita ja painikkeita ei vielä tee sovelluksesta kovinkaan hyödyllistä. Tätä varten Qt:ssa on useita moduuleita, jotka tarjoavat tiettyjä toimintoja.

- QtNetwork-moduuli tarjoaa verkkotoiminnot. Se mahdollistaa kommunikoinnin verkon yli esimerkiksi *TCP/IP*- tai *UDP*-protokollien avulla.

- QtSql-moduulin avulla sovellus voi tallentaa tietojaan tietokantaan. Qt tarjoaa tuen useille eri tietokantamoottoreille kuten *MySQL* ja *SQLite*.

- QtXml-moduulin toimintojen avulla voidaan lukea ja kirjoittaa *XML*-dokumentteja *DOM*- tai *SAX*-rajapintoja käyttäen.

- Phonon on alun perin KDE-projektin kehittämä korkean tason multimediarajapinta, jonka avulla voidaan helposti toistaa ääntä ja liikkuvaa kuvaa. Phonon ei itsessään sisällä multimediatoimintoja, vaan se on eri taustajärjestelmille yhteinen abstrakti rajapinta, jonka avulla voidaan toteuttaa perusominaisuudet sisältävä musiikkia tai videoita toistava sovellus.

- QtWebkit-moduulin avulla voidaan sovellukseen integroida WebKit-internetselainmoottori, jolla voidaan näyttää verkkosivuja. WebKit on Applen kehittämä *HTML*-moottori, joka perustuu KDE-projektin kehittämään *KHTML*-moottoriin.

- QtOpenGl-moduuli mahdollistaa standardin OpenGL-rajapinnan käyttämisen 3D-grafiikan tuottamiseen Qt-sovelluksissa.

[BlS08, Wik12f]

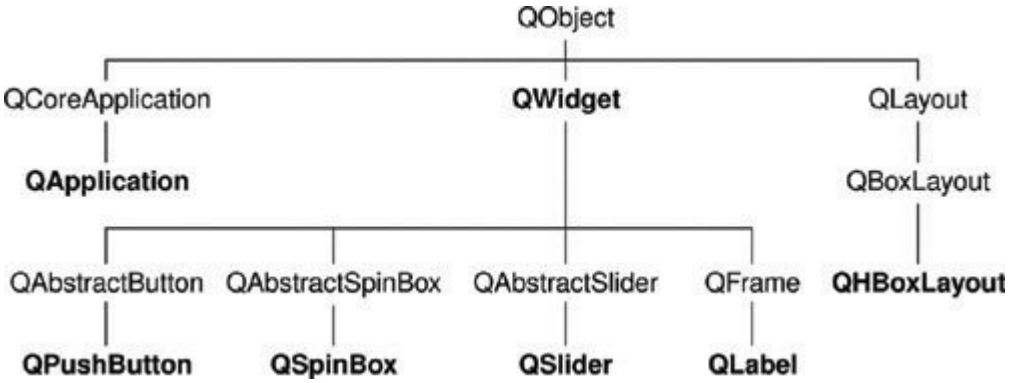
3.3 Qt:n arkkitehtuuri

Qt siis perustuu oliomalliin, jossa jokainen toimija on QObject-luokan instanssi. Oliot ovat hierarkkisesssa järjestyksessä, jossa oliolla voi olla yksi vanhempi ja useita lapsia. Tämän mallin avulla voidaan helpottaa mm. muistinhallintaa. Kun yksi olio tuhotaan, se tuhoaa samalla kaikki lapsiolionsa. Näin ohjelmoijan tarvitsee huolehtia muistivuotojen ehkäisemisestä vähemmän itse. [BlS08]

3.3.1 Oliohierarkia

Qt:ssa on kaksi perusluokkaa: QObject ja siitä periytyvä QWidget. Näistä ensimmäinen on perusluokka kaikille interaktiivisille Qt-olioille. QWidget lisää QObjectiin graafiset perustoiminnot ja on pohja kaikille graafisille käyttöliittymäkomponenteille.

QObject-luokassa toteutetaan hierarkkinen rakenne, jossa oliot järjestetään puuhun lapsi-vanhempi-periaatteella. Jokaisella oliolla voi olla yksi vanhempi ja monta lasta. Tämä mahdollistaa automaattisen muistinhallinnan, jolloin varattujen muistialueiden vapauttaminen ja luotujen olioiden tuhoaminen ei ole yksin ohjelmoijan vastuulla. Kun olio tuhotaan, se tuhoaa automaattisesti myös kaikki lapsiolionsa. C++-ohjelmoinnissa perinteisesti ohjelmoijan täytyy itse muistaa vapauttaa varaamansa muistialueet.



Kuva 2: Esimerkki Qt-luokkien hierarkkisesta rakenteesta [BlS08]

Oliohierarkian taustalla on myös Qt:n metaoliojärjestelmä, jonka avulla QObject-olioista on mahdollista saada ajonaikaisesti ns. metatietoja kuten olioon kuuluvat signaalit ja slotit ja sen kantaluokat aina QObject-luokkaan asti ovat. Standardi C++-kieli ei tue tällaista järjestelmää, joten sitä varten on luotu moc-työkalu, joka parsii QObjectista periytyviin luokkiin lisätyn Q_OBJECT-makron perusteella ja luo tarvittavat toiminnot käyttäen standardia C++-kieltä. Täten tämä laajennus ei vaadi kääntäjältä erikoistukea. [BlS08]

3.3.2 Tapahtumasilmukka

Qt noudattaa matalalla tasolla tapahtumapohjaisen ohjelmoinnin periaatetta. Se luo ohjelman käynnistyksessä tapahtumasilmukaksi kutsutun alijärjestelmän, jonka avulla voidaan yhdessä säikeessä suorittaa näennäisen rinnakkaisesti eri tehtäviä eli tapahtumia. Tapahtumat ovat yksinkertaisia operaatioita kuten kursorin siirtäminen ruudulla, näppäimen painallus tai vapautus, tai graafisen elementin lähettämä käsky piirtää sen sisältö uusiksi. Tapahtumasilmukka on yleisesti käytetty tapa graafisten käyttöliittymien toteuttamiseen. [Wik12b]

Qt:ssa tapahtumat on pääasiassa tarkoitettu olioiden omaan käyttöön niiden sisäisen tilan muuttamiseksi ja muutoksiin reagoimiseksi. Ohjelmoinnillisesta näkökulmasta tapahtumien metodit onkin määritelty protected-tasolla, eli niihin on pääsy vain saman luokan instansseilla. Muut oliot voivat kuitenkin kuunnella toisen olion tapahtumia erillisen tapahtumasuotimen (event filter) avulla. Suodin mahdollistaa myös tapahtuman estämisen kokonaan. [BlS08]

3.3.3 Signals and slots -ohjelmointimalli

Toinen tapahtumien kanssa hyvin samankaltainen paradigma olioiden väliseen kommunikointiin on tarkkailijaperiaatteeseen (observer pattern) perustuva signals and slots -malli. Siinä missä tapahtumien kuuntelu sitoo oliot toisiinsa, signals and slots -mallissa viestejä voidaan välittää olioiden välillä ilman, että kummankaan tarvitsee tietää mitään toisistaan.

Tarkkailijaperiaate perustuu kahteen toisistaan erilliseen toimijaan: tarkkailijaan (observer) ja tarkkailtavaan (observable). [Wik12j] Tarkkailtavan olion tilan muuttuessa tarkkailijat voivat reagoida muutoksiin haluamallaan tavalla, mutta tarkkailtavan olion ei tarvitse tietää, ketkä tai kuinka monta muuta oliota sitä tarkkailevat.

Esimerkiksi Java- ja C#-ohjelmointikielet tarjoavat omat toteutuksensa tarkkailijaperiaatteesta [WIk12j], mutta Qt:ssa se on yksi tärkeimmistä ohjelmoijan työkaluista ja integraalinen osa Qt-sovellusten kehitystä. [BlS08]

Sen sijaan, että tarkkailijat vain kuuntelisivat tarkkailtavaa oliota ja vastaanottaisivat kaikki ilmoitukset mistä tahansa tilan muutoksesta, voidaan tarkkailijat asettaa kuuntelevaan vain tiettyjä ilmoituksia (signaalit) ja jokainen ilmoitus asettaa kutsumaan tiettyä tarkkailijan slot-metodia. Lisäksi signaalit voivat kuljettaa mukanaan arvoja, jolloin tarkkailijan ei edes tarvitse tietää mitään tarkkailemansa olion rajapinnasta tai ominaisuuksista. [BlS08]

Ohjelmoinnillisesta näkökulmasta signaali on suojattu metodi, jonka ohjelmoija esittelee otsikkotiedostossa mutta jonka koodia hän ei itse kirjoita. Signaalimetodin koodi generoidaan Qt:n moc-esiprosessoijan (meta object compiler) toimesta. Slot-metodit ovat tavallisia metodeja, jotka voivat olla julkisia, suojattuja tai yksityisiä, ja niitä voidaan kutsua kuten tavallisiakin metodeita. [BlS08]

Tapahtumien lisäksi oliot voivat kommunikoida keskenään signals and slots -mallin avulla. Siinä missä olion tapahtumakutsut ovat suojattuja, signals & slots tarjoaa julkisen rajapinnan olioiden väliseen viestintään. Signaalin lähettäjä ja vastaanottaja voivat jopa sijaita eri säikeissä. [BlS08]

Signaali on metodi, jonka ohjelmoija vain määrittelee otsikkotiedostossa mutta toteutuksesta huolehtii Qt:n esiprosessoija ohjelmakoodin käännöksen yhteydessä. Signaalit ovat näkyvyysalueeltaan suojattuja. [BlS08]

Slot on tavallinen metodi, joka vain määritellään otsikkotiedostossa slot-tyyppiseksi. Niitä voi silti kutsua kuten muitakin olion metodeja. Slot-metodien näkyvyysaluetta ei ole rajoitettu, vaan ne voidaan määritellä julkisiksi, suojatuiksi tai yksityisiksi.

Signaaleja tai slot-metodeja sisältävän luokan tulee periytyä QObject-perusluokasta. Signaalien kytkemiselle ei ole varsinaisesti rajoituksia. Signaalin voi kytkeä sekä slot-metodeihin että toisiin signaaleihin. Yhden signaalin voi kytkeä yht'aikaisesti useaan eri vastaanottajaan, jopa samaan slot-metodiin. Tällöin kyseistä slot-metodia kutsutaan yhtä monta kertaa, kuin signaali on siihen kytketty.

Kytkettävän signaalin ja vastaanottajan ns. allekirjoitusten eli metodin ottamien parametrien täytyy kuitenkin täsmätä. Vastaanottaja saa ottaa *enintään* yhtä monta parametria ja niiden täytyy olla samassa järjestyksessä. [BlS08]

Esimerkkejä kelvollisista kytkennöistä:

```
connect(object_a, SIGNAL(signal_a(int, bool)),
        object_b, SLOT(foo(int, bool)));

connect(object_a, SIGNAL(signal_a(int, bool)),
        object_b, SLOT(bar(int)));

connect(object_a, SIGNAL(signal_a(int, bool)),
        object_c, SIGNAL(test_signal(int)));
```

3.4 Siirrettävyys ja eri ympäristöihin mukautuminen

Qt-kirjastot on suunniteltu järjestelmäriippumattomiksi. Tämä tarkoittaa sitä, ettei ohjelmoijan itse tarvitse toteuttaa mekanismeja ja yhteensopivuuskerroksia, jotta ohjelmakoodi toimisi eri ympäristöissä. Tämä helpottaa ja nopeuttaa ohjelmiston kehittämistä ja julkaisua eri alustoille.

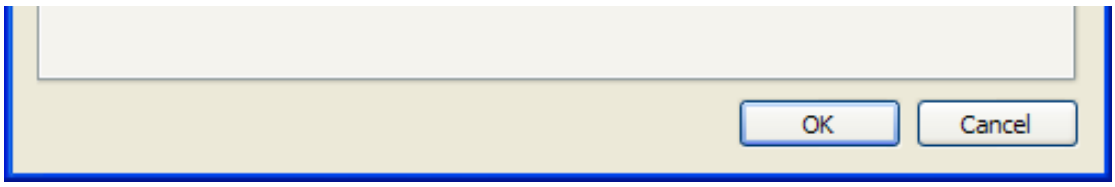
Qt tarjoaa kuitenkin rajapintoja myös joillekin alustariippuvaisille toiminnoille, jotka mahdollistavat paremman integraation kyseisessä ympäristössä. Nämä eivät kuitenkaan näyttele merkittävää roolia Qt-kehityksessä, vaan perustason ohjelmiston voi toteuttaa geneerisesti ilman käyttöympäristökohtaisia muutoksia. [BlS08]

3.4.1 Natiivi käyttöliittymä

Qt:ssa on tuki useille eri käyttöliittymille. Tuettuja käyttöliittymiä ovat mm. Windows XP, Windows Vista/7, Mac OS X sekä Unix-ympäristöt Gnome ja KDE SC. Käyttöliittymätuki tarkoittaa sitä, että ohjelma noudattaa kyseisen ympäristön käyttöliittymän yhtenäistämisohjeita (*Human Interface Guidelines, HIG*).

Yhtenäinen käyttöliittymä tarkoittaa paitsi visuaalisten elementtien ulkoasua, myös esimerkiksi niiden järjestystä tai sijoittelua tietyssä kontekstissa kuten dialogi-ikkunoissa. [BlS08]

Kuva 3: Painikkeiden järjestys ja ulkoasu Gnome-ympäristössä



Kuva 4: Painikkeiden järjestys ja ulkoasu Windows-ympäristössä

Ulkoasujen ajonaikainen mukauttaminen on toteutettu keskittämällä kaikki graafisten elementtien piirtotoiminnot yhteen luokkaan: QStyle. Vaikka jokainen käyttöliittymän elementti eli widget vastaa itse itsensä renderöimisestä, tosiasiassa ne vain kutsuvat QStyle-luokan piirtometodeja itse määräämillään arvoilla.

Jokainen eri käyttöliittymä on toteuttu omassa QStylen aliluokassa. Nämä aliluokat taas voivat olla riippuvaisia käyttöympäristön omista rajapinnoista, joten kaikki tyylit eivät välttämättä ole siirrettävissä toisiin ympäristöihin. Esimerkiksi Mac OS X:n ulkoasuluokka QMacStyle ei ole siirrettävissä muille alustoille.

Niiltä osin kuin ulkoasuvaihtoehtoja on saatavilla, niiden välillä voidaan vaihdella jopa ajonaikaisesti. Lisäksi ohjelman voi määrittää käyttämään tiettyä tyyliä antamalla sille käynnistysparametrin *-style <tyylin_nimi>*. [BlS08]

3.4.2 Multimediarajapinta

Qt:n multimediamoduuli on nimeltään Phonon. Sen kehitys alkoi alun perin KDE-projektina, jonka tarkoitus oli tuottaa pitkällä aikavälillä vakaa ja yhtenäinen rajapinta eri käyttöjärjestelmille. Myöhemmin se otettiin kiinteäksi osaksi Qt-projektia ja on siten käytettävissä ilman KDE-kirjastoja.

Phonon on korkean tason rajapinta, joka mahdollistaa mediatiedostojen helpon toistamisen. Se ei tarjoa kaikkia eri ympäristöjen erikoisuuksia vaan tarkoitus on tarjota helppo ja yhtenäinen rajapinta median toistamiseen. Phonon toimii välikappaleena Qt-sovelluksen ja käyttöjärjestelmän oman multimediarajapinnan välillä. Täten sen tuki eri multimediaformaateille riippuu käytettävästä taustakomponentista (backend). [BlS08]

Kiintolevyllä sijaitsevien ääni- ja videotiedostojen lisäksi Phonon tarjoaa rajapinnan myös CD- ja DVD-levyjen toistamiseksi.

3.4.3 Verkkotoiminnot

Internet-yhdeyden käyttäminen Qt:lla onnistuu QtNetwork-moduulin toiminnoilla. Se tarjoaa tuen niin matalan tason TCP/IP- ja UDP-protokollille kuin myös korkeamman tason *HTTP*- ja *FTP*-protokollille sekä myös SSL-suojaukselle. Myöhemmin Qt:iin on lisätty uusi Network Access *API*, jonka perustana on QNetworkAccessManager-luokka. Se tarjoaa asynkronisen rajapinnan verkkotoiminnoille ja osaa myös lähettää useita pyyntöjä rinnakkain.

3.4.4 Monisäikeistys

Monisäikeistys Qt:ssa on pyritty tekemään helpommaksi kuin se saattaisi olla käyttöjärjestelmän omilla rajapinnoilla. Monisäikeistykseen on tarjolla kaksi tapaa: QObject-olioiden siirtäminen toiseen säikeeseen QThread-luokan avulla ja QtConcurrent-malli, joka tekee resurssien mukaan ajonaikaisesti skaalautuvien ohjelmien kirjoittamisesta hyvinkin yksinkertaista.

QThread-luokan instanssi ei itsessään ole säie vaan se hallitsee yhtä säiettä. QObject-luokassa on metodi moveToThread(QThread *), jolla periaatteessa mikä tahansa QObject-luokan instanssi voidaan siirtää toiseen säikeeseen. Rajoituksena on se, ettei graafisen käyttöliittymän komponentteja voida siirtää säikeestä toiseen.

QtConcurrent on erillinen nimiavaruus, joka mahdollistaa erilaisten algoritmien suorittamisen listalle elementtejä käyttäen rinnakkaislaskennan paradigmoja. Se huolehtii itse säieturvallisuuden eri aspekteista, kuten käsiteltävän datan lohkottamisesta niin, etteivät eri säikeet käsittele samoja alkioita samaan aikaan. Lisäksi laskennan etenemistä on mahdollista seurata asynkronisesti ulkopuolelta käsin.

Tällä hetkellä QtConcurrent tarjoa on kolme eri algoritmia: filter, map ja MapReduce.

- Filter eli suodin: Poistaa listasta alkiot, jotka eivät täsmää suodinfunktion ehtoihin.

- Map: Suorittaa jokaiselle listan alkiolle saman funktion. Tuloksena on lista, jossa on yhtä monta alkiota kuin syötteessäkin.

- MapReduce: Koostuu kahdesta välivaiheesta: map ja reduce. Jokaiselle alkiolle suoritetaan map-funktio, jonka jälkeen alkiot syötetään yksi kerrallaan reduce-funktiolle. Map-osuus on rinnakkainen, mutta reduce-funktiota kutsuu vain yksi säie kerrallaan.

Käyttääkseen näitä rinnakkaislaskennan malleja ohjelmoijan tarvitsee ainoastaan toteuttaa yhtä alkiota käsittelevä funktio (filter tai map ja reduce) ja kutsua QtConcurrent-nimiavaruuden metodia, jolle kyseinen käsittelyfunktio annetaan parametrina. [BlS08]

4 QT-SOVELLUKSEN TOTEUTTAMINEN

Tässä luvussa perehdytään sovelluksen toteuttamiseen Qt-kirjastoja ja C++-kieltä käyttäen. Ensin käydään läpi viralliset Qt-ympäristön kehitystyökalut. Käytännön esimerkin virkaa toimittaa eräs keskeneräinen harrasteprojektini, jonka työnimi on Monolake. Kyseisestä sovelluksesta oli tarkoitus tulla erikoistyöni tämän pro gradu -tutkielman oheen, mutta erinäisistä kiireistä johtuen olen päättänyt lykätä sen aktiivisen kehitystyön jatkamisen valmistumisenjälkeiseen elämään.

4.1 Työkalut

Paitsi että Qt on kattava ohjelmakirjasto, sen ympärille on myös rakennettu kokonainen kehitysympäristö, Qt *SDK*. Oleellisin apuohjelma lienee *qmake*, jota ohjelmoija voi käyttää Qt-projektien perustamiseen ja ylläpitämiseen. Qmake on komentoriviltä ajettava sovellus ilman graafista käyttöliittymää.

Qmake generoi ensin Qt-projektin konfigurointitiedoston, *projektin_nimi.pro*, joka sisältää mm. projektin lähdekooditiedostot, määritykset tarvittavien Qt-moduulien sisällyttämiseksi ja ohjeet kolmannen osapuolten ohjelmakirjastojen lisäämiseksi projektiin. Lähdekoodista tuotetaan sovellus käyttämällä erityisesti Unix-ympäristöissä laajalti käytettyä make-työkalua. Qmake generoi pro-tiedoston tietojen avulla Makefile-nimisen tiedoston, jonka pohjalta make osaa käyttää C++-kääntäjää tuottamaan lähdekoodista sovelluksen.

Qmaken lisäksi Qt-kehitysympäristöön kuuluu esimerkiksi Microsoftin Visual Studiota tai Oraclen NetBeans-kehitysympäristöä vastaava Qt Creator. Qt Creator automatisoi ylläpidollisia toimenpiteitä ja mm. ajaa qmaken aina tarvittaessa automaattisesti. Sovelluskehittäjän ei kuitenkaan ole pakko valita tavallisen tekstieditorin ja komentorivisovellusten sekä Qt Creatorin välillä miettiessään valintaa kehitysympäristöksi, sillä Microsoftin Visual Studio -kehitysympäristöön tehty laajennus mahdollistaa Qt-projektien lisäämisen Visual Studioon.

Kolmas työkalu on Qt Designer, jota käyttäen voidaan suunnitella graafisia käyttöliittymiä ns. *WYSIWYG*-ympäristössä (what you see is what you get). Käyttöliittymälomakkeet tallennetaan levylle XML-muodossa. Ne lisätään projektin .pro-tiedostoon ja sieltä qmake poimii ne generoimaansa Makefileen. Näin lomakkeista generoidaan projektin kääntämisen yhteydessä C++-otsikkotiedostoiksi, jotka voidaan sisällyttää ohjelmakoodiin #include-direktiivillä kuten muutkin C++-otsikkotiedostot.

Käyttöliittymien suunnittelun lisäksi Designerissa voidaan kytkeä widgettien signaaleita ja slotteja toisiinsa, mikä vähentää entisestään tarvetta kirjoittaa itseään toistavaa ohjelmakoodia. [BlS08, QtP11a, QtP11b]

4.2 Esittely

Monolake on avoimen lähdekoodin projekti, jonka tarkoituksena on tuottaa perusominaisuudet kattava musiikkisoitin pääasiallisesti Linux-ympäristöön. Toissijaisina ympäristöinä on tarkoitus tukea Windowsia ja mahdollisesti myöhemmin myös Mac OS X:ää.

Tällä hetkellä Monolake toimii Linuxilla ja Windowsilla, joskin jälkimmäisellä alustalla ilmeni muutamia pienempiä toimintavirheitä eli bugeja. Toimintaa tai ohjelmakoodin kääntämistä Mac OS X:llä ei ole ollut mahdollista testata. Monolake koostuu valmiiden Qt:n luokkien lisäksi 76 uudesta luokasta.

Monolaken ominaisuuksiin kuuluvat musiikin toistamisen ja musiikkikokoelman ylläpitämisen ohella muita perustoimintoja

- Musiikkikokoelman tallentaminen SQLite- tai MySQL-tietokantaan.
- Kansikuvagrafiikoiden haku internetistä Microsoftin Bing-hakukoneen avulla.
- Internet-radioiden toisto.

- Tavalliset soittolistat ja hakuehtojen perusteella automaattisesti päivitettävät soittolistat.

- Soittotietojen päivittäminen Last.fm-musiikkipalveluun.

- Multimedianäppäinten tuki Linux-ympäristössä *DBus*-protokollaa käyttäen.

Monolake käyttää yhteensä kuutta Qt-moduulia: QtCore, QtGui, QtNetwork, QtSql, QtXml ja Phonon. Lisäksi se hyödyntää kahta kolmannen osapuolen ohjelmakirjastoa: TagLib ja Liblastfm, jotka ovat kumpikin avoimen lähdekoodin projekteja.

TagLib-kirjastoa käytetään musiikkitiedostojen metatietojen lukemiseen. Se on tuettu kaikilla yleisimmillä käyttöjärjestelmillä. Vaikkei TagLib itsessään sisällä riippuvuuksia Qt-kirjastoihin, sen kehitys on alun perin alkanut osana Amarok-nimistä Qt/KDE-projektia, joten se on turvallinen valinta Qt-sovelluksissa käytettäväksi.

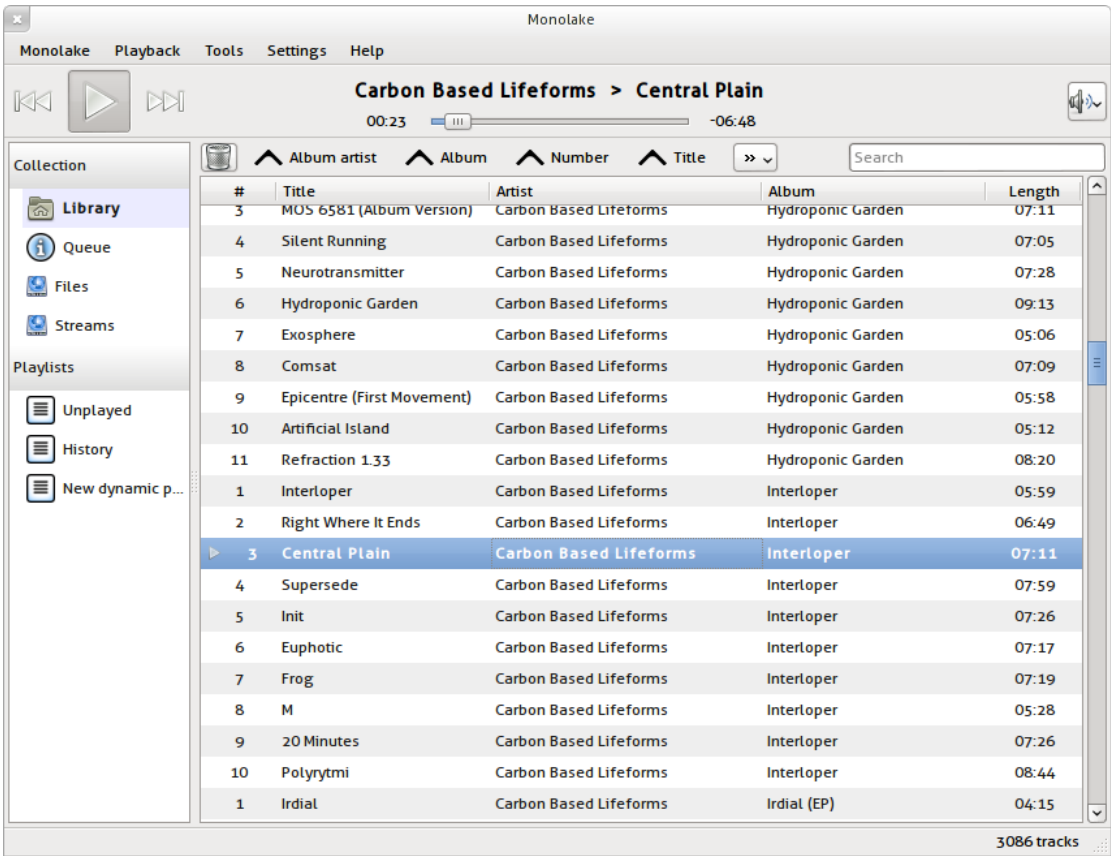
Liblastfm on Last.fm-palvelun kehittämä kirjasto heidän verkkopalvelunsa käyttämiseksi. Se mahdollistaa käyttäjän kuuntelemisen musiikkikappaleiden tiedot reaaliajassa hänen julkiselle profiilisivulleen. Lisäksi sovellukset voivat hakea tietoa musiikkikappaleista Last.fm:n tietokannoista. Tällä hetkellä Monolake tukee ainoastaan soittotietojen päivittämistä, mutta parempi Last.fm-integraatio on suunnitelmalistalla.

4.3 Ohjelman rakenne

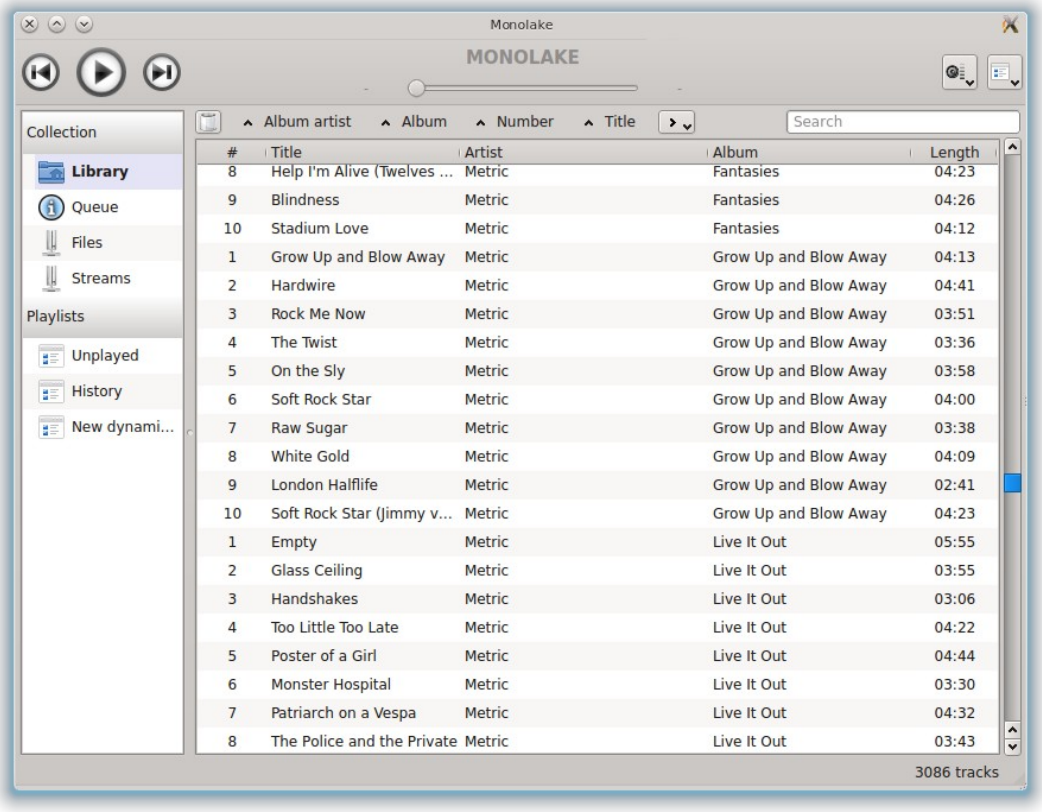
Tässä luvussa esitellään Monolaken ydintoiminnot ja muut perustominaisuudet.

4.3.1 Perustoiminnot

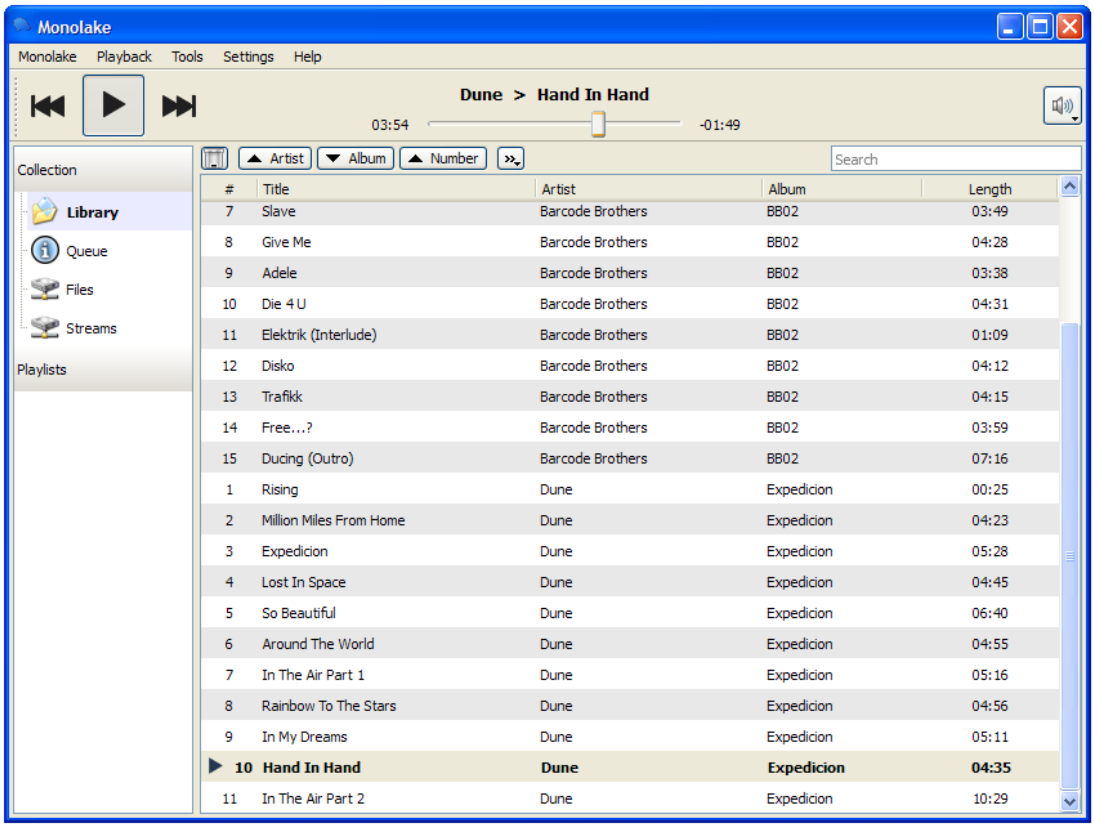
Monolaken kaikki oleellisimmat toiminnot on keskitetty sovelluksen pääikkunaan. Sen kautta käyttäjä voi hallita musiikkikokoelmaansa ja luoda uusia soittolistoja sekä hallita musiikin toistamista. Kuvissa on Monolaken pääikkuna esitettynä kolmessa eri käyttöympäristössä.



Kuva 5: Monolake Gnome-työpöytäympäristössä



Kuva 6: Monolake KDE-työpöytäympäristössä



Kuva 7: Monolake Windows XP:llä

Monolaken perustoiminnot koostuvat neljästä ydinkomponentista: MainWindow, PlayerEngine, PlaylistManager ja CollectionManager. MainWindow on siis äsken mainittu pääikkuna. PlayerEngine-luokka on äänen toistamisesta huolehtiva moottori. PlaylistManager ja CollectionManager hallitsevat soittolistoja ja musiikkikokoelmaa.

PlaylistManager on niinikään singleton, ja se ylläpitää listaa soittolistoista ja niiden ryhmittelyistä. Soittolista koostuu Model-View-Controller- eli MVC-arkkitehtuurin mukaisesti kolmesta eri komponentista: tietorakenteesta (model), näkymä-widgetistä (view) ja taustalla olevasta ohjainluokasta (controller).

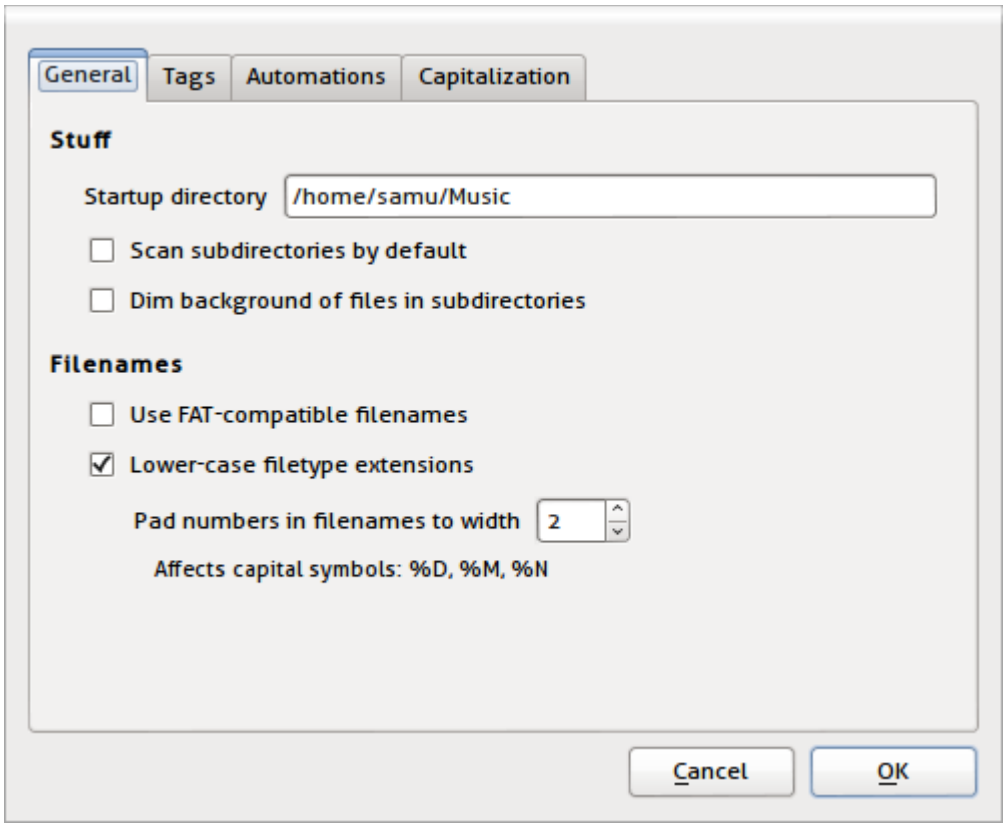
CollectionManager hallitsee keskitettyä musiikkikokoelmaa, johon kuuluvat lokaalin musiikkikirjaston lisäksi internetin yli toistettavat mediat ja tulevaisuudessa myös tietokoneeseen kytkettävien lisälaitteiden kuten kannettavien mediatoistinten ja

ulkoisten kiintovylevyjen tiedostot. Soittolistojen malliluokat ovat ns. välittäjämalleja (proxy model), jotka eivät itse sisällä dataa vaan ainoastaan ohjaavat tiettyyn indeksiin kohdistuvan pyynnön lähdemalliin (source model). Tämän lähestymistavan etuna on se, että muutos musiikkikappaleeseen missä tahansa soittolistassa välittyy automaattisesti kaikkiin muihinkin soittolistoihin. Tämä perustuu siihen, että datan muuttuessa malli lähettää aina *dataChanged*-signaalin, jota mallia käyttävät näkymäluokat kuuntelevat ja reagoivat siihen päivittämällä näkymänsä muuttuneiden tietojen osalta.

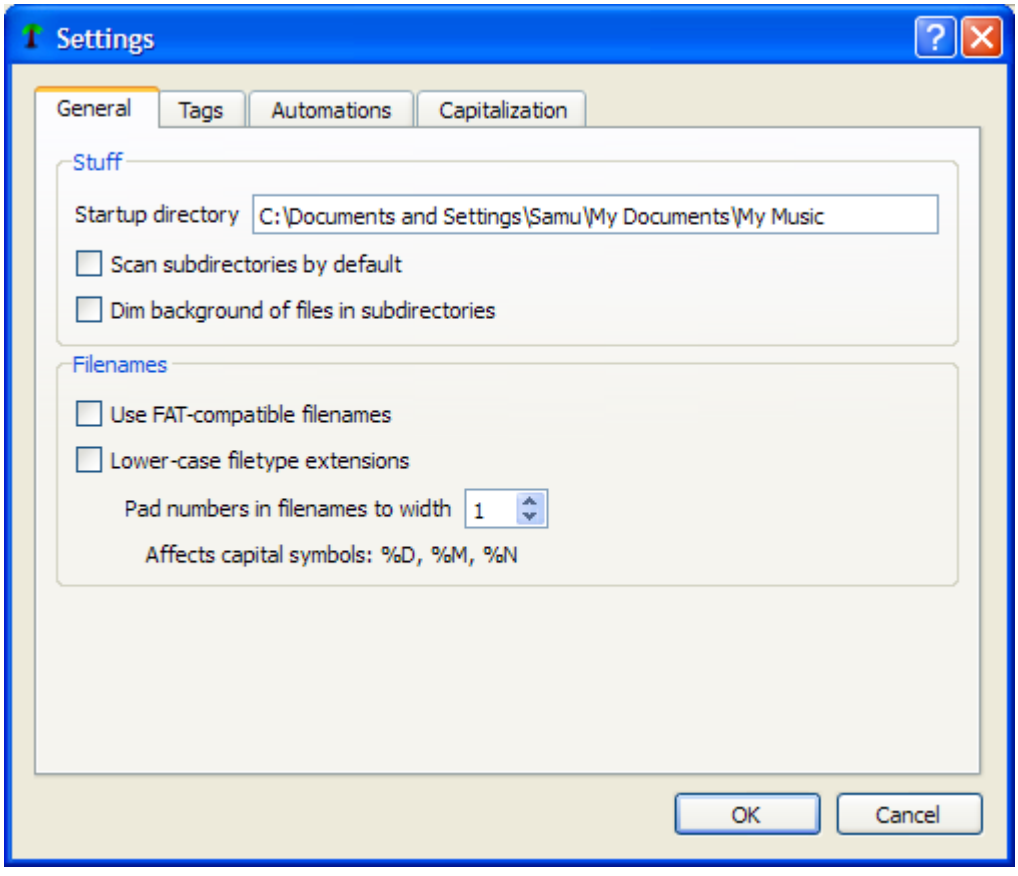
CollectionManager ylläpitää kokoelmaa tietokannassa, jonka moottorina voi toimia Qt:ssa natiivisti tuettu SQLite tai ulkoinen MySQL-tietokanta. Soittolistat säilytetään samassa tietokannassa ja ne vain linkittävät tietyn collection-taulun kappaleen yhteen tai useampaan riviin soittolistalla. CollectionManager myös lisää uudet löytämänsä kappaleet tietokantaan. Koska kappaleiden metatietojen lukeminen tiedostoista on raskas toimenpide, on tiedostojen etsiminen ja lukeminen eristetty omaan MediaScanner-komponenttiinsa, joka sijaitsee eri säikeessä kuin muu sovellus.

Soittolistoja on neljää eri tyyppiä: kirjasto, soittojono (queue) ja tavalliset sekä dynaamiset soittolistat. Tavalliset soittolistat ovat käyttäjän hallittavissa olevia listoja, joiden sisältöä käyttäjä voi vapaasti muokata. Dynaamiset soittolistat hakevat sisältönsä käyttäjän määrittämien hakuehtojen perusteella ja päivittyvät reaaliajassa.

Jono on erikoistyyppi soittolistoista. Käyttäjä voi lisätä siihen musiikkikappaleita, jotka saavat korkeimman prioriteetin. Tämän jälkeen seuraava toistettava kappale haetaan aina jonosta siinä järjestyksessä, kuin ne on siihen lisättykin. Jonon tyhjennyttyä kappaleet haetaan jälleen aktiivisena olevalta soittolistalta.



Kuva 8: Dialogi-ikkunan mukautuminen Gnome-ympäristöön



Kuva 9: Dialogi-ikkunan mukautuminen Windows-ympäristöön

4.3.2 Lisätoiminnot

Lisätoimintoina Monolakeen on toteutettuna tällä hetkellä kokoelmaan kuuluvien albumien kansikuvien hakeminen internetistä sekä soittotietojen päivittäminen Last.fm-palveluun.

Kansikuvien haku on toteutettu käyttäen Microsoftin Bing-hakukoneen hakurajapintaa. Haun tekeminen rajapintaa käyttäen tapahtuu tavallisella GET-pyynnöllä, jossa hakuehdot välitetään palvelimelle www-osoitteessa GET-muuttujina. Palvelin vastaa pyyntöön XML-muotoisella viestillä, josta Monolaken CoverFinder-olio lukee ladattavien kuvien osoitteet ja syöttää ne CoverDownloader-luokan instansille. CoverDownloader lataa kuvat käyttäjän tietokoneelle ja tallentaa ne levylle.

Last.fm-tuki on niin ikään internet-yhteyttä hyödyntävä toiminto. Sen avulla käyttäjän soittotiedot päivitetään automaattisesti käyttäjän Last.fm-profiiliin, josta ne näkyvät julkisella verkkosivulla muille ihmisille. Viestintä tapahtuu hyvin samankaltaisesti kuin kansikuvahaun yhteydessä. Tällä kertaa palvelimelle lähetettävät viestit kulkevat kuitenkin POST-muotoisina. Palvelin vastaa jälleen XML-viestillä. Lisähaasteena on kuitenkin autentikointi, sillä jokaisen käyttäjän tulee tietysti päästä muokkaamaan vain omaa profiiliaan. Autentikointi on toteutettu OAuth-protokollalla, jota käyttävät myös suuremmatt palvelut kuten Facebook ja Google.

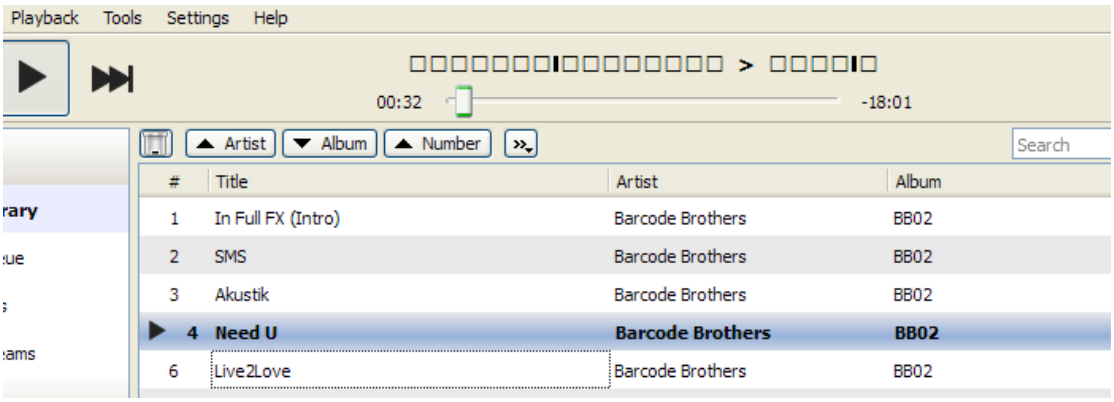
4.3.3 Huomattuja ongelmia

Linux-käyttäjänä minulle henkilökohtaisesti tärkein alusta on Linux, joten Windows-versiota ei ole testattu nimeksikään gradutyöskentelyn ulkopuolella. Siksi olikin huojentavaa todeta, että Monolake toimi ensimmäisellä yrittämällä varsin kohtuullisesti.

Ensimmäinen Windowsiin liittyvä ongelma ilmeni jo käännösvaiheessa, kun ensimmäiseksi yritin kääntää Monolakesta versiota, joka oli pilkottu useampaan

moduuliin. Linuxilla käännös sujui ongelmitta, mutta Windowsilla jostain syystä polkumäärittelyt eivät toimineet yhtä luotettavasti, vaan moduuleita yritettiin hakea väärästä hakemistosta. Koska Monolaken nykyinen versio on vielä suhteellisen pieni, niin yhdistin kaikki moduulit yhdeksi suureksi kokonaisuudeksi, jolloin moduuleita ei tarvinnut linkittää laisinkaan.

Toisen ongelman havaitsin liittyen itse tekemääni widgettiin, joka näyttää toistettavan musiikkikappaleen metatietoja. Joidenkin kappaleiden kohdalla kyseisen widgetin tekstinäyttö sekosi ja näytti virheellisiä merkkejä. Ongelma on ihmeellinen erityiseksi siksi, ettei kyseisissä nimissä ollut lainkaan erikoismerkkejä, jotka olisivat voineet olla looginen selitys toimintahäiriölle.



Kuva 10: Tekstinäytön virheellinen toiminta Windowsilla

4.4 Ohjelmakoodin kääntäminen eri ympäristöissä

Qt-projektin lähdekoodien kääntäminen toimivaksi sovellukseksi on suhteellisen yksinkertaista. Qmakella luotu pro-tiedosto sisältää kaikki tarvittavat tiedot, joiden pohjalta qmake generoi monimutkaisemman Makefilen, jonka perusteella make-työkalu osaa kutsua tarvittavia ohjelmia oikeilla parametreilla lähdekoodin kääntämiseksi ja kirjastojen linkittämiseksi toimivaksi kokonaisuudeksi.

Pro-tiedostoon pitää lisätä käsin tarvittavat Qt-moduulit QT-muuttujaan.

```
QT += network phonon sql xml
```

Lisäksi täytyy määritellä käytetyt lisäkirjastot, joita tässä tapauksessa ovat Last.fm-kirjasto *liblastfm* sekä TagLibin kirjasto *libtag*. Tässä kohdin tulee hieman eroja eri käyttöjärjestelmien välillä, sillä Linuxissa voi käyttää järjestelmään globaalisti asennettuja kirjastoja, mutta Windowsilla on parempi paketoida tarvittavat lisäkirjastot sovelluksen mukaan erillisinä DLL-tiedostoina.

Tämä ei ole isokaan ongelma, sillä pro-tiedostoon voi määritellä järjestelmäkohtaisia lohkoja. Linux-ympäristön erikoismäärittelyt laitetaan unix-lohkoon ja Windowsin määrittelyt win32-lohkoon. Linuxissa globaalit kirjastot voidaan sisällyttää vivulla *-l*, kun Windowsissa viitataan suoraan DLL-tiedostoon. DLL-tiedostot on sijoitettu projektin juuresta katsoen hakemistoon *resources/windows/<kirjaston_nimi>/*.

Unix-lohkossa on myös linkitys Monolaken mlk_plugins-moduuliin, joka lisää Linux-versioon tuen multimedianäppäimille Linuxilla. Vivulla *-L* määritellään hakemisto, josta linkittäjä etsii lisäkirjastoja. Mlk_plugins-kirjasto muodostaa oman aliprojektinsa, jolla on erillinen pro-tiedosto.

```
QT += network phonon sql xml
unix {
        LIBS += -ltag -llastfm
        LIBS += -L../libs -lmlk_plugins
}
win32 {
        LIBS += ../resources/windows/taglib/libtag.dll
        LIBS += ../resources/windows/lastfm/liblastfm.dll
}
```

Kaava 3: Käyttöjärjestelmäkohtaiset määritykset .pro-tiedostossa

Lähdekoodit käännetään tämän jälkeen sovellukseksi ajamalla ensin komento *qmake*, jonka jälkeen suoritetaan Linuxilla komento *make.* Windowsissa makea vastaa komento *mingw32-make*, kun käytetään QtSDK:n mukana tulevaa Mingw-käännösympäristöä. Tämän jälkeen tuloksena on valmis sovellus.

Windows-version käynnistämiseksi tarvitsee vielä kopioida tarvittavat DLL-tiedostot samaan hakemistoon sovelluksen kanssa. Nämä tiedostot ovat: mingwm10.dll, phonon4.dll, QtCore4.dll, QtGui4.dll, QtNetwork.dll, QtSql4.dll, QtXml4.dll, libtag.dll ja liblastfm.dll.

5 VAIHTOEHDOT QT:LLE

Vaikka Qt on hyvin laaja kehysympäristö, on sille olemassa myös joitakin vaihtoehtoja, jotka tarjoavat enemmän tai vähemmän samat ominaisuudet. On olemassa useita pienempiä kirjastoja, jotka tarjoavat käyttöjärjestelmä-riippumattomuuden, mutta joiden ominaisuudet riittävät lähinnä käyttöliittymän luomiseen, kun laajemmat ominaisuudet puuttuvat.

5.1 GTK+

Suurin kilpailija ainakin Linux-kehittäjän näkökulmasta on ollut *GTK*+ (Gimp Toolkit). Se sai alkunsa grafiikkaohjelma Gimpin kehityksen perustana, mutta on sittemmin muuttunut täysin omaksi projektikseen ja sen ympärille on Qt:n tapaan kehitetty kokonainen työpöytäympäristö Unix-käyttöjärjestelmille. GTK+ ei itse sisällä varsinaisesti muuta kuin käyttöliittymäkomponentit, mutta sen yhteyteen on kehitetty Gnome-työpöytäympäristöä silmällä pitäen Qt:a vastaavat multimedia- ja verkkotoiminnot erillisinä kirjastoina.

Qt:n tavoin GTK+ noudattaa olio-ohjelmoinnin periaatteita, joskin se on kirjoitettu C-kielellä, jossa ei ole varsinaisesti olio-ohjelmoinnin paradigmaa. GTK+:sta on kuitenkin virallinen C++-versio, joka tunnetaan nimellä *Gtkmm*. GTK-kirjastot ovat käytettävissä C:n ja C++:n ohella lukuisilla muillakin ohjelmointikielillä kuten Python ja Ruby. [Wik12g]

GTK+ ei ole kuitenkaan yhtä järjestelmäriippumaton. Linuxin ohella uusin GTK+:n 3-versio on virallisesti saatavilla vain Mac OS X:lle, joskin Windows-tuki saattaa saapua myöhemmin. Vanhempi 2-versio oli jossain määrin tuettu myös Windowsilla, tosin päivityksiä julkaistiin harvemmin. GTK+ tarjoaa natiivin ulkoasun myös Macilla, mutta siinä ei ole tukea Qt-ympäristöön mukautumiseen. Qt:ssa sen sijaan on tuki GTK-ulkoasuun mukautumiselle. [GTK12, Wik12g]

GTK on kuitenkin de facto -standardi Unix-maailmassa siinä mielessä, että siihen perustuva Gnome-työpöytäympäristö on virallinen työpöytäympäristö suosituimmissa Linux-jakeluissa. Lisäksi Linuxin kanssa kilpailevan Oraclen Solaris-käyttöjärjestelmän oletusympäristönä on vanhempaan GTK+ 2:een perustuva Gnome 2.30. [Wik12g, Wik12h]

5.2 Java

Java on siinä mielessä mielenkiintoinen kohde, ettei se ole ohjelmakirjasto vaan itsenäinen ohjelmointikieli, joka sisältää graafiset käyttöliittymäkomponentit ja kattaa myös monet muut tarpeelliset ominaisuudet. Java on järjestelmäriippumaton ohjelmointikieli, joka on saatavilla mm. Windowsille, Mac OS X:lle ja Linuxille. Java on ominaisuuksiltaan laajempi kuin C++ ja sen standardikirjasto, Java Class Library, tarjoaa tuen niin monisäikeistykselle, verkkotoiminnoille kuin graafisille käyttöliittymillekin. Javassa on oma graafisten käyttöliittymien kirjastonsa, Swing. Se osaa myös Qt:n tavoin mukautua GTK+-sovellusten ulkoasuun Unix-ympäristössä. [Ora12, Wik12i]

Koska Java on oma ohjelmointikielensä, sille ei tietenkään ole vaihtoehtoisia toteutuksia toisin kuin Qt:n kanssa, jolloin ohjelmoija voi periaatteessa valita haluamansa kielen tuettujen joukosta ja käyttää Qt-kirjastoja sen kanssa. Qt:sta on olemassa myös Java-versio, QtJambi, jonka kehitystyö on kuitenkin lähes pysähtynyt Trolltechin pudotettua sen pois virallisten Qt-projektien listalta. [BlS08, QtJ11]

Lisäksi Javan luonne on erilainen kuin esimerkiksi C++:n tai Pythonin. Java-sovellukset on tarkoitettu ajettavaksi Javan virtuaalikoneessa, mikä taas edellyttää kyseisen virtuaalikoneen asentamista jokaiseen tietokoneeseen, jossa Java-sovelluksia halutaan ajaa. C++:lla kirjoitettu Qt-sovellus vaatii toimiakseen vain muutaman Qt:n kirjaston, jotka voidaan jakaa sovelluksen mukana DLL-tiedostoina, eivätkä siten vaadi erillistä asennusta. Qt:n modulaarisen arkkitehtuurin ansiosta sovelluksen mukaan voidaan paketoida vain tarvitut Qt-moduulit. [Ora12]

6 POHDINTA

Qt on hyvä valinta sovelluskehitykseen, kun kohteena on Linux tai muu avoimen lähdekoodin käyttöjärjestelmä, minkä lisäksi se on varsin toimiva työkalu käyttöjärjestelmäriippumattomaan sovelluskehitykseen. Qt-sovellukset integroituvat kohtuullisen hyvin käyttöympäristöönsä. Se on erittäin monipuolinen sekä ominaisuuksiltaan kattava ja mielestäni vailla vertaistansa.

Onkin suoranainen harmi, että Nokia päätti ajaa MeeGo- ja Qt-kehityksensä alas, joskin kuulemani mukaan heillä saattaa olla vielä joitain muita suunnitelmia Qt:n osalta. Nokia itse julisti Windows-suunnitelmiensa julkistamisen yhteydessä silti pyrkivänsä tuomaan Qt-sovellukset miljardille uudelle käyttäjälle, mutta toistaiseksi mistään tällaisesta ei ole merkkejä havaittavissa.

Vaikka Nokia on jo ulkoistanut kaupallisen Qt-lisensoinnin muualle ja kehitystyökin on virallisesti siirretty Qt-yhteisön harteille, on se edelleen vahvasti mukana Qt:n kehityksessä. Qt:n tulevaisuus ei silti ole suinkaan riippuvainen Nokiasta, ja olihan sillä vahva tuki jo ennen kuin Nokia edes osti Trolltechin. Viime aikoina erityisesti Intelin puolelta on panostettu voimakkaasti Qt:n kehitykseen. Qt:stä on lisäksi tulossa tämän vuoden kesällä tai viimeistään syksyllä uusittu 5.0-versio, jossa erityisesti mobiiliominaisuuksiin on panostettu aiempaa enemmän.

Valitettavasti useilla käyttöjärjestelmillä toimiminen ei taida olla nykypäivänä asia, jolla olisi isoille ohjelmistotaloille suurempaa merkitystä. Markkinoilla ei ole Windowsin valtaannousun jälkeen oikeastaan edes ollut kilpailua eri käyttöjärjestelmien välillä, ja vaikka Applen Mac-tietokoneet ovat alkaneet kasvattaa suosiotaan, sovelluskehittäjien kiinnostus ei ole laajamittaisesti vielä herännyt. Linux-järjestelmät eivät ole työpöytäkäytössä keränneet mittavaa suosiota.

VIITTEET

[Bec05] Becker, P.: Working Draft, Standard for Programming Language C++. ACM, sivu 62, 2005. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/.

[BlS08] Blanchette, J., Summerfield, M.: C++ GUI Programming with Qt 4, Second edition. Prentice Hall, 2008. http://blog.hartwork.org/?p=156

[GTK12] GTK+ Packages. http://www.gtk.org/download/win32.php (25.4.2012)

[Moo12] Mooney, J.D.: Issues in the Specification and Measurement of Software Portability. Department of Statistics and Computer Science, West Virginia University. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.6878&rep=rep1&type=pdf (20.3.2012)

[Pau08] Ars Technica: Nokia to Buy Trolltech, will become a patron of KDE. Ars Technica, 2008. http://arstechnica.com/information-technology/2008/01/nokia-buys-trolltech-will-become-a-patron-of-kde/ (20.02.2012)

[Obi12] Obiltschnig, G.: Designing and Building Portable Systems in C++. Applied Informatics. http://www.appinf.com/en/company/papers.html (15.3.2012)

[Ora12] Oracle: How to Set the Look and Feel. Oracle. http://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html (25.4.2012)

[QtJ11] Qt Jambi. http://qt-jambi.org/ (24.5.2012)

[QtP11a] Qt Project: Qt Creator Manual. Qt Project, 2011. http://doc.qt.nokia.com/qtcreator-2.4/index.html (20.02.2012)

[QtP11b] Qt Project: Qt Designer Manual. Qt Project, 2011. http://qt-project.org/doc/qt-4.8/designer-manual.html (20.02.2012)

[QtP11c] Qt Project: Supported Platforms. Qt Project, 2011. http://qt-project.org/doc/qt-4.8/supported-platforms.html (20.02.2012)

[Wik12a] Wikipedia: Human Interface Guidelines. Wikipedia, 2012. http://en.wikipedia.org/wiki/Human_interface_guidelines (20.4.2012)

[Wik12b] Wikipedia: Event Loop. Wikipedia 2012. http://en.wikipedia.org/wiki/Event_loop (20.3.2012)

[Wik12c] Wikipedia: Software portability. Wikipedia, 2012. http://en.wikipedia.org/wiki/Software_portability (21.4.2012)

[Wik12d] Wikipedia: Endianness. Wikipedia, 2012. http://en.wikipedia.org/wiki/Endianness (21.4.2012)

[Wik12e] Wikipedia: Qt (framework). Wikipedia 2012. http://en.wikipedia.org/wiki/Qt_%28framework%29 (20.02.2012)

[Wik12f] Wikipedia: Webkit. Wikipedia, 2012. http://en.wikipedia.org/wiki/Webkit (20.4.2012)

[Wik12g] Wikipedia: GTK+. Wikipedia, 2012. http://en.wikipedia.org/wiki/Gtk%2B (25.4.2012)

[Wik12h] Wikipedia: OpenSolaris Desktop. Wikipedia 2012. http://en.wikipedia.org/wiki/OpenSolaris_Desktop (25.4.2012)

[Wik12i] Wikipedia: Java (programming language). Wikipedia 2012. http://en.wikipedia.org/wiki/Java_%28programming_language (25.4.2012)

[Wik12j] Wikipedia: Observer pattern. Wikipedia 2012.
http://en.wikipedia.org/wiki/Observer_pattern (20.3.2012)