Mediaplayer loppuraportti

Sisällysluettelo

Yleiskuvaus

Tässä kappaleessa avataan yleisellä tasolla projektiaihetta ja sen perustoteutusta. Mediasoitin on ohjelma joka toistaa kaikenlaista mediaa (musiikki ja ääni). Se toimii graafisella ja helppokäyttöisellä käyttöliittymällä, johon voi tuoda omat mediatiedostot ja toistaa niitä tietokoneen ruudulla ja kaiuttimissa. Mediasoittimen on myös tarkoitus helpottaa media ylläpitoa ja järjestystä selkeällä käyttöliittymällä, metadatan muokkausmahdollisuudella sekä tiedostojen järjestämisellä esimerkiksi aakkosjärjestykseen. Mediasoittimessa voi toistaa medialistaa, muuttaa soiton järjestystä sekä vapaasti valita soitettava media. Perus play/stop toiminnallisuuden lisäksi soittimesta löytyy myös sekoitus toiminto.

Ohjelman rakenne

Tässä kappaleessa perehdytään ohjelman rakenteesta. Käyttöliittymän ympärille rakennetaan monta pienempää komponenttia, jotkat tuovat ikkunaan toiminnallisuuksia. player.cpp-tiedostoon tulee kaikki median kontrolloimiseen liittyvät ominaisuudet, joita kutsutaan käyttöliittymäkomponenttien luokista.

Yleis arkkitehtuuri

Ohjelma koostuu main.cpp, player.cpp ja ui.cpp sekä sen eri komponenttitiedostoista. Jokaiselle ohjelmamme .cpp tiedostolle löytyy vastaavat .hpp tiedostot, joissa määritellään kukin luokka ja niiden metodit. Cpp tiedostoissa luodaan implementaatiot. Main.cpp käynnistää ohjelman luomalla uuden applikaation (QtApplication) ja täyttää sen Player (player.cpp) sekä UI (ui.cpp) luokilla. UI luokassa luodaan uudet komponentit (PlaylistModel, MenuBar, ToolBar, CenterBar, ControlBar,

Actions), joita avataan tarkemmin kappaleessa Luokat.

Luokat

Player luokka

Player luokka mahdollistaa kappaleiden toistamisen ja pysäyttämisen. Player luokka perii QWidget luokan. Player luokalla on 11 metodia. setMedia metodilla asetataan player luokalle soittolista, addMedia metodilla lisätään asetettuun soittolistaan mediatiedostoja. Play metodi toistaa soittolistan mediasisältöä, pause metodi keskeyttää toistettavaa mediasisällön. getVolume metodi palauttaa äänenvoimakkuuden, setVolume metodi mahdollistaa äänenvoimakkuuden säädön ja clearPlaylist metodilla voidaan tyhjentää, setMedia metodilla asetettu soittolista.

Lisäksi Player luokalla on kaksi 'getter' metodia, getQMediaPlayer ja getQMediaPlaylist, joiden avulla pääsee käsiksi Player luokan QMediaPlayeriin ja QMediaPlaylistiin. getQMediaPlayer metodilla on määre const, sillä sitä ei voi metodin getQMediaPlayer avulla muuttaa, mutta getQMediaPlaylist metodin avulla voidaan muuttaa Player luokan QMediaPlaylistia (playlist). Player luokalla on kaksi private osoitinta, joihin pääsee luokan ulkopuolella käsiksi edellä mainituilla 'getter' metodeilla.

Lisäksi luokalle on implementoitu constructor ja destructor.

Playlistmodel luokka

Playlistmodel luokka perii QAbstractItemModel luokan ja ottaa parametriksen Player luokan. Luokalla on kahdeksan metodia. rowCount, columnCount, index, parent, data ja headerData metodit ovat Qt:n vaatimia metodeja, jotta QAbstractItemModel tyyppisiä luokkia voidaan käyttää. rowCount palauttaa luokan private muuttujan m_data_matrix rivimäärän ja columnCount sarakkeiden lukumäärän. index metodi palauttaa QModelIndex tyyppisen luokkamuuttujan, jota Qt vaatii, jotta model/view tyyppistä tietorakennetta voidaan käyttää. index metodi palauttaa kyseisen indeksikohdan, kun luokkaa vastaavaa QTableView instansia klikkaa. data metodi palauttaa niin ikään QModelIndex tyyppisen luokkamuuttujan. QModelIndex:n avulla voidaan paikantaa dataa mallistamme. dataMetodi ottaa parametreikseen referenssin QModelIndexiin, sekä roolimuuttujan, joka kertoo halutaanko mallia katsoa (Qt::DisplayRole) tai esimerkiksi muokata Qt::EditRole. Malliamme ei voi muokata. headerData metodi palauttaa QVariant tyyppisen muuttujan, joka on yleiskäsite, kaikille Qt:n muuttujille. headerData ottaa parametreikseen kokonaisluvun section, Qt::Orientation tyyppisen muuttujan sekä kokonaislukumuuttujan role vastaavalla tavalla kuin data metodi. headerData metodi vastaa taulukon otsikkorivi ja sarakkeiden täydentämisestä.

Luokalla on myös metodit setPlaylist ja clearData, jotka molemmat ovat void tyyppisiä eivätkä siten palauta mitään. setPlaylist metodilla täytetään luokan private muuttuja m_data_matrix. m_data_matrix on QVector<QVector> tyyppinen muuttuja, jossa riveillä on mediatiedoston tiedostopolku, alkuperäinen indeksi soittolistassa, mediatiedoston metadatat (otsikko, artisti, album ja median pituus). setPlaylist metodissa luodaan TagLib ulkoisessa kirjastossa oleva Tag luokka, jonka avulla päästään käsiksi mediatiedoston metadataan.

Lisäksi luokalla on Player luokkaan osoitin, jonka avulla päästään käsiksi Player luokan soittolistaan.

Jotta Playlistmodel luokan dataa voidaan muuttaa, Qt edellyttää, että slotteihin lisätään metodit beginRemoveItems ja endRemoveItems, joiden nimistä voi päätellä, että niillä on tarkoitus signaloida milloin rivien poistaminen mallista alkaa ja milloin se loppuu.

PlaylistModel luokalla on lisäksi julkinen (public) slot sortByCol, jonka avulla voidaan järjestää m_data_matrix otsikon, albumin, artistin tai median pituuden mukaan.

UI luokka

UI luokka perii QMainWindow luokan, jolle Qt on luonut valmiin pohjan eri QWidgettien sijoittamiselle näytölle (layout). UI luokalla kuten kaikilla projektimme luokilla on constructor ja destructor sekä yksi metodi, initComponents (void). UI luokalla on yksityisiä (private) osoittimia luokkiin Player, PlaylistModel, Menubar, Toolbar, Controlbar, Centerbar ja Actions. UI luokassa luodaan uudet instassit kaikkiin luokkiin paitsi Player luokkaan. Instanssien luominen tapahtuu intiComponents metodissa. Lisäksi UI luokassa asetetaan teema ja ulkoasun värimaailma. Luotu instanssi Menubar asetaan UI luokan menuksi kutsumalla QMainWindow:n setMenuBar metodia. Vastaavasti Toolbar ja Controlbar instassit asetaan UI luokan työkalukomponenteiksi kutsumalla QMainWindow:n addToolBar metodia antamalla sille paramaterina tieto siitä, minne ikkunan kohtaan kyseinen Toolbar halutaan sijoittaa (top tai bottom). Centerbar instassi asetetaan UI luokan keskimmäiseksi komponentiksi kutsumalla QMainWindow:n setCentralWidget. Actions luokan instanssi hyödyntävät Menubar ja Toolbar luokat.

Actions luokka

Actions luokka perii QObject luokan ja sille annetaan macro Q_OBJECT, joka takaa, että Qt:n sisäänrakennettu SIGNAL/SLOT systeemi toimii. Actions luokka ottaa parametrikseen UI luokan (QMainWindow), Player luokan, PlaylistModel luokan ja Centerbar luokan osoittimina. Actions luokalla on yksi metodi getActions, joka palauttaa listan kaikista hyödyllisitä toiminnoista. Luokalla on yksi julkinen (public) osoitinmuuttuja toolButtonAction. Yksityistiä (private) sloteista löytyy metodit addTracks, changeIndex, addToPlaylist, removePlaylist, sortPlaylistByTitle, sortPlaylistByArtist, sortPlaylistByAlbum, sortPlaylistByLength, savePlaylist ja exit. Kaikki metodit ovat void tyyppisiä.

addTracks metodi avaa tiedostodialogin samassa kansiossa, jossa projektimme on. Tiedostodialogin avaamisesta vastaavalle funktiolle on annettu parametrina suodatin, joka rajaa mitä tiedostotyyppejä tiedostodialogi-ikkuna näyttää. Vaikka QMediaPlayer tukee useita eri mediatiedostotyyppejä, päätimme rajoittaa tiedostodialogi-ikkunassa näkyviä tiedostotyyppejä vain .mp3, .wav, .ogg ja .mp4 tiedostotyyppeihin. Muuttamalla suodatinlsitaa, voidaan helposti näyttää käyttäjälle useampia QMediaPlayer:n tukemia mediatiedostotyyppejä. addTracks metodilla lisätään mediatiedostopolkuja Player luokan QMediaPlaylistiin kutsumalla Player luokan metodia addMedia. Mediatiedostopolut lisätään myös PlaylistModeliin kutsumalla Player luokan getQMediaPlaylist metodia juuri luodulla uudella QMediaPlaylistillä.

changeIndex metodi vaihtaa Centerbar luokan sisäistä QWidgettiä riippuen onko

mediaplayerillämme toistettavaa medialistaa. Jos medialistassa on dataa, ilmaantuu näytölle taulukko, jossa näkyy mediasoittimen soittolistan sisältö. Centerbar luokka on QStackedWidget, jonka takia, sillä on indeksejä ja niillä indekseillä on erilaisa QWidgettejä.

addToPlaylist, removePlaylist, savePlaylist ja median jäjrestämiseen liittyvät metodit (esim. sortPlaylistByTitle) nimensä mukaisesti muokkaavat mediaplayerimme soittolistaa. savePlaylist metodia ei implementoitu, josta johtuen soittolistoja ei voida tallentaa.

Lisäksi Actions luokalla on yksityisiä osoittimia Player, PlaylistModel, Centerbar ja QMainWindow (UI luokka) ja muuttujia. actionStrings muuttuja on tyypiltään std::vector . actionStrings vektorissa on nimet kaikille toiminoille. actionVector pitää sisällään listan QAction* osoittimista.

Menubar luokka

Menubar luokka perii QMenuBar luokan, ja luokalle on annettu Q_OBJECT macro. Yksityisiä osoittimia luokalla on QMainWindow*, Player* ja Actions*. Julkisia osoittimia ovat kaikki menut.

Menubar luokka ottaa parametreina UI luokan (QMainWindow) Player luokan ja Actions luokan osoittimina.

Toolbar luokka

Toolbar luokka on vastaavanlainen kuin Menubar luokka, mutta sillä erotuksella, että luokan parametreina löytyy myös Menubar luokka. Luokassa asetetaan Actions luokan toiminnot tähän luokkaan.

Centerbar luokka

Centerbar luokka perii QStackedWidget luokan ja sille on määritetty Q_OBJECT macro. Centerbar luokka ottaa parametrikseen osoittimet luokkkiin QWidget, Player, PlaylistModel ja Controlbar. Centerbar luokassa luodaan kaikki vaadittavat elementit ja komponetit QMainWindow:n (UI luokan) centralWidgetille. Luokalla on yksi julkinen metodi hasEnding, jolla tutkitaan päättyykö toistettava mediatiedosto muuhun kuin .mp4. hasEnding metodia hyödynnetään, jotta voidaan näyttää video, jos tiedosto on videotiedosto ja staatinen logokuvamme, jos tiedosto on musiikkitiedosto.

Luokalla on yksityisiä SLOT metodeja; jump, syncCurr ja mediaType. jump metodilla vaihdetaan toistettavaa mediaa riippuen mitä riviä käyttäjä taulukossa klikkaa. syncCurr synkronoi playlistView ja playlistmodel osoittimet. mediaType metodi vaihtaa näkymää (video tai kuva) riippuen hasEnding metodin paluuarvosta (boolean).

Lisäksi luokalla on liuta yksityisiä osoittimia erilaisiin QWidget ja QLayout luokkiin.

Controlbar luokka

Controlbar luokka perii QToolbar luokan ja sille on määritelty Q_OBJECT macro. Luokalla on yksi julkinen metodi getPlayButton, joka palauttaa osoittimen playbuttoniin (QPushButton*). Controlbar luokka mahdollistaa painikkeet mediaplayerin median perus toimnnallisuuksille. Lisäksi luokalla on

erilaisia tarpeellisia julkisia SLOT metodeja, joiden avulla voidaan muun muassa siirtää mediaplayer toistamaan seuraavaa mediaa (nextmedia()) tai aktivoida satunnainen soittojärjestys (shuffle()).

Ulkoisten kirjastojen rajapinta

Qt

Qt helpottaa graafisen käyttöliittymän tekoa sekä tuo paljon valmiita ominaisuuksia kuten esimerkiksi raahaa ja pudota-ominaisuuden.

Qt omaa myös multimedia osion joka sopii hyvin meidän projektiin mm. QMediaPlayer luokan ansiosta. Qt multimedia on kuitenkin osoittaunut ongelmalliseksi linux ympäristössä ja on syytä vaihtaa jos ongelmia esiintyy lisää.

TagLib metatietokirjasto

Qt:n oma metatietojen hakuun luotu kirjasto osoittautui vajaavaiseksi eikä täytä kaikkia vaatimuksiamme. Valitsimme mediatiedostojen metadatan hakemiseen ulkoiseksi kirjastoksi TagLib nimisen kirjaston. TagLib -kirjastolle uuden Tag:n luominen onnistuu hyvin, ja sen avulla mediatiedostoista voi kätevästi saada esimerkiksi otsikon, artistin tai albumin nimen.

Ohjeet ohjelman ajamiselle ja käytölle

Asennus koulun koneella

ENGLISH BELOW

Lataa zip tai tar tiedosto painikkeesta projektin juurta kuvaavasta GitLab projektisivulta (tämä).

Pura lataamasi paketti ja avaa terminaali kansiossa, johon purit lataamasi tiedoston.

Suorita seuraavat terminaalikomennot järjestyksessä.

```
cd libs/taglib-1.9.1

cmake -DCMAKE_INSTALL_PREFIX=./bin -DCMAKE_RELEASE_TYPE=Release

make

make install

cd ../../src

qmake -qt=qt5

make

./mediaplayer
```

ENGLISH

Download zip or tar file from this website which represents the root of our project.

Unpack the file which you downloaded.

Open terminal in the directory where you unpacked the files you downloaded.

Use terminal commands above in order to build and eventually to execute our project.

Vaatimukset

Ohjelma toimii ainoastaan Linuxilla

ja tarvitsee toimiakseen lisäksi seuraavat kirjastot:

Qt

Asenna kirjasto heidän kotisivuiltaan

taglib:

Asnna kirjasto heidän kotisivuiltaan

tai suoraan terminalista:

```
wget http://taglib.github.io/releases/taglib-1.9.1.tar.gz

tar -xzvf taglib-1.9.1.tar.gz

cd taglib-1.9.1

sudo apt-get install cmake

cmake -DCMAKE_INSTALL_PREFIX=/usr/local -DCMAKE_RELEASE_TYPE=Release

make

sudo make install
```

Testaaminen

Yksikkötestaus löytyy kansiosta src/tests ja se toimii omana Qt:n projektina. Yksikkötesteissä on tällä hetkellä kaksi yksinkertaista testiä, mutta testejä olisi nyt helppo laajentaa eteenpäin. Tällä hetkellä tests kansiossa testataan player luokan toiminnallisuuksia.

Työskentely loki

Sakari ja Santeri sopivat suullisesti työnjaon. Toteutus jäi Sakarin ja Santerin vastuulle. Tapaamisia

pidettiin noin joka toinen viikko. Tapaamisissa käytiin läpi kunkin henkilökohtaisia panoksia projektin edistämiseksi ja mergettiin master ajan tasalle.

Kuvaus projektin vastuualueista

Vastuualueita jaettiin sitä mukaa, kun edelliset projektin osa-alueet olivat valmiita. Vastuualueet jakautuivat seuraavalla tavalla:

Santeri: player, controlbar, ui, main ja tests Sakari: centerbar, menubar, toolbar, actions ja playlistmodel

Tekijät

Santeri Suitiala
Eelis Pinomaa
Sakari Pöyhiä
Jan Lundström