Государственное образовательное учреждение
высшего профессионального образования
Петрозаводский государственный университет
Математический факультет

Кафедра информатики
и математического обеспечения

Отчет о ходе выполнения курсовой работы

Разработка приложений под ОС Maemo/MeeGo

Исполнитель:
студент группы 22305
_____ В. В. Мошанин

Научный руководитель:
инженер (совм.), ст. преподаватель
_____ А. В. Бородин

Оценка научного руководителя:
_____

Представлена на кафедру
« ___ » _____ 2010 г.
Оценка оформления
и сроков представления отчёта:
_____

Петрозаводск — 2010

Оглавление

Введение

С появлением новых языковых средств программирования, создание приложений под различные мобильные ОС становиться всё проще, а их возможности только увеличиваются. Без внимания не обходятся и развлекательные приложения.

Имея опыт работы с профессиональными музыкальными программами для создания и обработки музыки, я решил изучить рынок развлекательных музыкальных приложений под системы Maemo/Meego. Оказалось, что таковых не так уж и много.

Так и родилась идей создания приложения, обрабатывающей звук в реальном времени, с помощью которой пользователь смог бы себя ощутить DJ, имея в руках только свой мобильный телефон. Данная программа будет иметь такие функции, как управление частотами воспроизводимой звуковой дорожки в реальном времени, регулирование скорости воспроизведения, панорамирование звука и много другое. Но главной отличительной чертой данного приложения будет являться наличие винилового проигрывателя, управление которым будет осуществляться с помощью сенсорного экрана. Это позволит имитировать управление настоящими DJ установками.

Для разработки данного приложения я решил выбрать библиотеку Qt, так как это кросс-платформенный инструментарий разработки ПО, с широкими возможностями мультимедиа. Данная библиотека включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, и звуковыми интерфейсами. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Так же, отличительной чертой Qt является наличие качественной документации, чтобы делает процесс разработки более лёгким и продуктивным.

Глава 1

Обзор мультимедийных классов библиотеки QT

1.1 Класс QSound

Данный класс обеспечивает доступ платформенных аудио средствам. Qt обеспечивает работу самой востребованной аудио операции в GUI приложениях: Асинхронно воспроизводит звуковой файл. Самый простой полноценный способ использования функции play():

```
QSound::play("mysounds/bells.wav");
```

Также существует другой способ. Создать QSound объект из звукового файла сначала, а потом вызвать функцию play():

```
QSound bells("mysounds/bells.wav");
bells.play();
```

Созданный объект QSound может быть вызван с помощью его имени filename() и суммарного количества повторов. Количество повторов может быть изменено с помощью функции setLoops(). Во время воспроизведения файла, функция loopsRemaining() возвращает оставшееся количество повторов. С помощью функции isFinished() можно выяснить закончил ли файл воспроизводиться или нет.

Звуки, воспроизводимые с помощью создания объекта QSound, могут требовать больше памяти, чем функция static play(), но этот метод более эффективный и быстрый. Используйте функцию static isAvailable, чтобы определить существуют ли аудио устройства на платформе.

1.2 Класс QAudioDeviceInfo

Класс QAudioDeviceInfo предоставляет интерфейс для запроса аудио устройств и их функциональных возможностей. QAudioDeviceInfo позволяет запрашивать для аудио устройств - таких, как звуковые карты и USB-гарнитур – которые в настоящее время доступны в системе. Список доступных аудио устройств зависит от платформы или аудио плагинов.

Вы можете также запрашивать, какие форматы каждое устройство поддерживает. Формат, в этом контексте является множеством, состоящим из описания порядка байтов, количества каналов, кодеков, частоты дискретизации, и типа семплов. Формат представляется классом QAudioFormat.

Значения параметров поддерживаемых устройством могут быть выбраны с supportedByteOrders(), supportedChannelCounts(), supportedCodecs(), supportedSampleRates(), supportedSampleSizes(), и supportedSampleTypes(). Поддерживаемые комбинации зависят от платформы, аудио-плагинов и аудио возможности устройства. Если вам необходимо использовать определенный формат, вы можете проверить, поддерживает ли устройство ли его с помощью функции isFormatSupported(), или получить самую точную информацию о том какой формат поддерживает устройство с помощью функции nearestFormat(). Например:

```
QAudioFormat format;
format.setFrequency(44100);
...
format.setSampleType(QAudioFormat::SignedInt);
QAudioDeviceInfo
info(QAudioDeviceInfo::defaultOutputDevice());
if (!info.isFormatSupported(format))
format = info.nearestFormat(format);
```

QAudioDeviceInfo используется в Qt для построения классов, которые общаются с устройствами – так же как и QAudioInput, и QAudioOutput. Статические функции defaultInputDevice функций(), defaultOutputDevice(), и availableDevices() позволяют получить список всех доступных устройств. Устройства выбираются в соответствии со значением QAudio::Mode enum. Возвращаемые значения QAudioDeviceInfo действительны только для QAudio::Mode. Например:

```
foreach(const QAudioDeviceInfo deviceInfo,
QAudioDeviceInfo::availableDevices(QAudio::AudioOutput))
qDebug() « "Device name: ≪< deviceInfo.deviceName();
```

См. также QAudioOutput и QAudioInput.

1.3 Класс QAudioFormat

Данный класс хранит информацию об аудио параметрах.

Показывает, как организован аудио поток, и как его интерпретировать. Кодировка зависит от кодека, который используется в потоке.

Кроме этого, QAudioFormat содержит другие параметры, описывающие организацию звуковой информации, такие как, частота, количество каналов, размер семпла, тип семпла, порядок байтов.

Вы можете получить аудио форматы совместимые с устройством, используя функции QAudioDeviceInfo. Этот класс также может запросить доступные значения параметров, поэтому вы можете установить параметры самостоятельно.

1.4 Класс QAudioInput

Класс QAudioInput обеспечивает интерфейс для получения аудио-данных от аудио устройства.

Аудио ввод можно создать с помощью устройства аудио ввода, заданного по умолчанию. Кроме того, можно создать QAudioInput с конкретными QAudioDeviceInfo. При создании аудио входа, вы также должны указать фомат в QAudioFormat, который будет использоваться для записи (смотри описание класса QAudioFormat).

Чтобы записать файл: QAudioInput позволяет записывать звук с аудио устройства ввода. Конструктор этого класса, по умолчанию будет использовать аудио устройство ввода, заданного по умолчанию, но вы можете также указать QAudioDeviceInfo для конкретного устройства. Вы также должны указать формат в QAudioFormat, в которой вы хотите записать.

Запуск QAudioInput осуществляется посредством вызова функции Start() и классом QIODevice, открытым для записи. Например, для записи в файл:

```
QFile outputFile; // class member.
QAudioInput* audio; // class member.
outputFile.setFileName("/tmp/test.raw");
outputFile.open( QIODevice::WriteOnly | QIODevice::Truncate );
QAudioFormat format;
// set up the format you want, eg.
format.setFrequency(8000);
format.setChannels(1);
format.setSampleSize(8);
format.setCodec("audio/pcm");
format.setByteOrder(QAudioFormat::LittleEndian);
format.setSampleType(QAudioFormat::UnSignedInt);
QAudioDeviceInfo info = QAudioDeviceInfo::defaultInputDevice();
if (!info.isFormatSupported(format))
qWarning()«"default format not supported try to use nearest";
format = info.nearestFormat(format);
```

```
audio = new QAudioInput(format, this);
QTimer::singleShot(3000, this, SLOT(stopRecording()));
audio->start(outputFile);
// Records audio for 3000ms
```

Чтобы проверить поддерживается ли указанный формат устройством ввода, используется QAudioDeviceInfo:: isFormatSupported(). В случае, если есть какие-то проблемы, использование функции error(), поможет посмотреть, что пошло не так. Остановка записи stopRecording().

```
void stopRecording()
audio->stop();
outputFile->close();
delete audio;
```

В любой момент времени, QAudioInput будет находится в одном из четырех состояний: активное, приостановленное, остановленное или простоя. Эти состояния определяются QAudio::State enum. Изменить состояние можно непосредственно через suspend(), resume(), stop(), reset(), и start(). Текущее состояние сообщает функция state(). QAudioOutputwill сигнализирует, когда изменяется состояние(stateChanged()).

QAudioInput предоставляет несколько способов измерения времени, прошедшего с момента запуска функции start(). Функция processedUSecs() возвращает длину потока в микросекундах записанных данных, т. е. не учитывает, когда аудио ввод был приостановлен или был в состоянии простоя. Функция elapsedUSecs() возвращает время, прошедшее с запуска start() независимо от такого в каком состоянии находился QAudioInput.

Если ошибка может произойти, вы можете извлечь причины с помощью функции error(). Возможные причины ошибки описываются в QAudio::Error enum. QAudioInput вступит в StoppedState когда произойдет ошибка. Подключение функции StateChanged(), поможет обработать сигнал об ошибке:

```
void stateChanged(QAudio::State newState)
switch(newState)
case QAudio::StopState:
if (input->error() != QAudio::NoError)
// Error handling
else
break;
```

1.5 Класс QAudioOutput

QAudioOutput класс предоставляет интерфейс для отправки звуковых данных в устройство вывода звука. Вы можете создать аудио выход с помощью устройства аудио вывода, заданного по умолчанию. Кроме того, можно создать QAudioOutput с конкретными QAudioDeviceInfo. При создании аудио выхода, вы также должны указать формат в QAudioFormat, который будут использоваться для воспроизведения (смотри описание класса QAudioFormat).

Для воспроизведения файла: Чтобы начанать воспроизводить аудио поток нужно вызвать функцию start() с QIODevice, открытым для воспроизведения. После этого QAudioOutput будет получать данные, которые требуется от ввода-вывода устройства. Пример фоспроизведения файлов:

```
QFile inputFile; // class member.
QAudioOutput* audio; // class member.
inputFile.setFileName("/tmp/test.raw");
inputFile.open(QIODevice::ReadOnly);
QAudioFormat format;
// Set up the format, eg.
format.setFrequency(8000);
format.setChannels(1);
format.setSampleSize(8);
format.setCodec("audio/pcm");
format.setByteOrder(QAudioFormat::LittleEndian);
format.setSampleType(QAudioFormat::UnSignedInt);
QAudioDeviceInfo info(QAudioDeviceInfo::defaultOutputDevice());
if (!info.isFormatSupported(format))
qWarning()«"raw audio format not supported by backend, cannot play audio.";
return;
audio = new QAudioOutput(format, this);
connect(audio,SIGNAL(stateChanged(QAudio::State)),SLOT(finishedPlaying(QAudio::State)));
audio->start(inputFile);
```

Файл начнет воспроизводится, если аудиосистема и устройства вывода поддерживают файл. Если ничего не происходит, то проверить в чем дело можно с помощью функции error().

```
void finishedPlaying(QAudio::State state)
if(state == QAudio::IdleState)
audio->stop();
inputFile.close();
delete audio;

]
```

В любой момент времени, QAudioOutput будет в одном из четырех состояний: активное, приостановленное, остановленное или простоя. Эти состояния определяются QAudio::State enum. Изменения состояния сообщается через сигнал stateChanged(). Этот сигнал можно использовать, например, обновляя графический интерфейс приложений. Изменить состояние можно непосредственно через suspend(), resume(), stop(), reset(), и start().

Когда аудио поток воспроизводится, вы можете установить интервал уведомления в миллисекундах с setNotifyInterval(). Этот интервал определяет время между двумя эмиссиями (выбросами) Notify() сигнала. Все зависит от положения в потоке, т. е. если QAudioOutput в SuspendedState или IdleState, Notify() сигнал не издается. Функция elapsedUSecs() вернет время с момента начала воспроизведения, независимо в каком состоянии находился QAudioOutput.

Если происходит ошибка, её тип можно узнать с помощью функции error(). Когда происходит ошибка, состояние переходит в QAudio:: StoppedState. Проверить наличие ошибок можно при подключении функции StateChanged():

```
void stateChanged(QAudio::State newState)
switch (newState)
case QAudio::StopState:
if (output->error() != QAudio::NoError)
// Perform error handling
else
// Normal stop
break;
```

Смотри так же QAudioInput и QAudioDeviceInfo.

Глава 2

Обзор мультимедийных фреймворков

2.1 Обзор фреймворка Phonon

Phonon — мультимедийный фреймворк, входит в состав Qt.

Особенности Phonon API написан на языке программирования C++ с использованием парадигм объектно-ориентированного программирования. Механизм использования интерфейса Phonon основан на графовых связях между источниками (MediaObject) и выводящими устройствами (AudioOutput, VideoOutput). Связи между объектами данных и устройств вывода реализуются с помощью путей (Path). Библиотека также поддерживает звуковые эффекты.

Плюсы:

1) Кросс-платформенность.
2) Простота использования.
3) Предоставление разработчикам возможности создания API-независимых приложений для воспроизведения видео и аудио данных.

Минусы:

1) Отсутствует поддержка работы с устройствами аудио- и видеозахвата.
2) Отсутствует доступ к видеобуферу для наложения видео эффектов в реальном времени.

2.2 Обзор фреймворка GStreamer

GStreamer — мультимедийный фреймворк, входит в состав в Qt.

Особенности GStreamer написан на языке программирования C и использующий систему типов GObject. GStreamer является «ядром» мультимедийных приложений, таких как видеоредакторы, потоковые серверы и медиаплееры. В изначальный дизайн заложена кроссплатформенность; GStreamer работает на Unix-подобных системах, а также на Microsoft Windows, OS/400 и Symbian OS. GStreamer предоставляет биндинги для других языков программирования таких, как Python, C++, Perl, GNU Guile и Ruby. GStreamer является свободным программным обеспечением, с лицензией GNU LGPL.

Плюсы:

1) Кросс-платформенность.
2) Широкие возможности для создания видео редакторов и аудио плееров.
3) Предоставление разработчикам возможности создания API-независимых приложений для воспроизведения видео и аудио данных.
4) Стабильность

Минусы:

1) Практически нет

Глава 3

Постановка задачи

3.1 Приложение

Основной целью данной работы является разработка приложения для ОС Maemo/MeeGo. В данном случае разрабатывается мультимедийное приложение:

1) Цель: создать приложение, для обработки звука в реальном времени.

2) Назначение: развлекательное музыкальное приложение, которое позволит пользователю ощутить себя DJ, имея в руках только свой мобильный телефон.

3) Предметная область: развлекательное музыкальное приложение

3.2 Функции

Важнейшие требуемые функции:

1) Реализовать программу, которая сможет воспроизводить мелодию, останавливать воспроизведение, и воспроизводить в обратную сторону.

2) Реализовать эмулятор винилового проигрывателя, управление которым будет осуществляться с помощью сенсорного экрана.

3) Возможность регулировать громкость воспроизведение, также реализовать кнопку выключение/включения звука.

4) Возможность регулирования скорости воспроизведения мелодии.

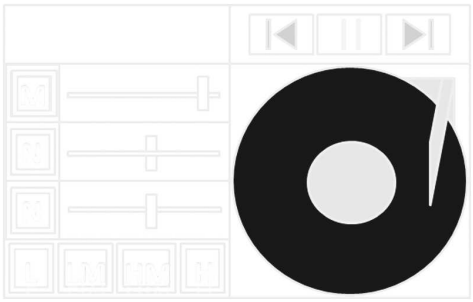
5) Возможность регулировки чувствительности управления виниловым проигрывателем.

6) Возможность управления отдельными частотами звука в реальном времени.

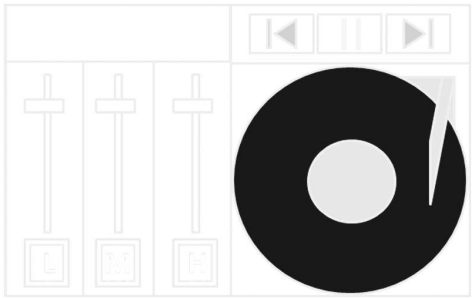
7) Возможность панорамирования звука в реальном времени.

3.3 Внешний вид проигрывателя

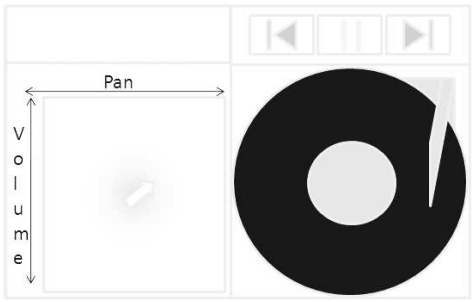
На рис.1 можно увидеть примерный интерфейс будущего приложения.



Окно управления частотами звука (см. рис.2): Данное окно позволяет контролировать громкость воспроизведения низких (L) - (16-250 Гц), средних (M) (250-2000 Гц) и высоких (H) (2000-20000 Гц) частот.



Окно панорамирования звука в реальном времени (см. рис.3): С помощью передвижения курсора по сенсорному экрану пользователь сразу управляет двумя параметрами звука: громкостью и панорамированием. Передвигая курсор вверх или вниз, громкость будет соответственно увеличиваться и уменьшаться. Передвигая курсор влево или вправо, звук будет перемещаться либо в левый или в правый канал соответственно.



Глава 4

Текущие результаты

На данный момент получены результаты

1) Разобраны классы библиотеки Qt, работающие с мультимедиа и звуком. Среди них такие классы как QSound, QAudioDeviceInfo, QAudioFormat, QAudioInput, QAudioOutput.

2) Разобраны мультимедиа фрэймворки, интегрированные и работающие с Qt. Отдельное внимание уделялось таким фреймворкам как Phonon и GSteamer.

3) Поставлена задача создания мультимедиа приложения, обрабатывающая звук в реальном времени. Началось проектирование программного обеспечения.

Библиографический список использованной литературы

[1] Ж. Бланшет, М. Саммерфилд *Qt 4: Программирование GUI на C++. 2-е дополненное издание.* — М.: «КУДИЦ-ПРЕСС», 2008. — С. 736.

[2] Макс Шлее *Qt 4.5 Профессиональное программирование на C++.* — СПб.: «БХВ-Петербург», 2010. — С. 896.

[3] Земсков Ю.В. *Qt 4 на примерах.* — СПб.: «БХВ-Петербург», 2008. — С. 608.

[4] Nokia QT Documentetion < http://doc.qt.nokia.com/ >

[5] Daniel Molkentin *The Book of Qt 4: The Art of Building Qt Applications* — No Starch Press, 2007 г. — С. 440.

[6] Mark Summerfield *Advanced Qt programming. Creating great software with C++ and QT4* — Addison-Wesley, 2010. — С. 499.