Phonon

Matthias Kretz

former KView maintainer

work on aRts and KDE Multimedia

Student of Physics and Computer Science at the University of Heidelberg

KDE Multimedia Meeting 2006
Outline

What is Phonon?

Design of Phonon
  Core Classes
  Code Examples
  User Visible Features
  How to Write a Backend
Multimedia API

- task-oriented design
- 80/20
- easy multimedia development
- no “competition” for GStreamer/NMM like media frameworks
- in KDE SVN: trunk/KDE/kdelibs/phonon (or branches/work/kdelibs4_snapshot/phonon)
Multimedia API

- task-oriented design
- 80/20
- easy multimedia development
- no “competition” for GStreamer/NMM like media frameworks
- in KDE SVN: trunk/KDE/kdelibs/phonon (or branches/work/kdelibs4_snapshot/phonon)
Multimedia API

- task-oriented design
- 80/20
- easy multimedia development
- no “competition” for GStreamer/NMM like media frameworks
- in KDE SVN: trunk/KDE/kdelibs/phonon (or branches/work/kdelibs4_snapshot/phonon)
Multimedia API

- task-oriented design
- 80/20
- easy multimedia development
- no “competition” for GStreamer/NMM like media frameworks

- in KDE SVN: trunk/KDE/kdelibs/phonon (or branches/work/kdelibs4_snapshot/phonon)
Multimedia API

- task-oriented design
- 80/20
- easy multimedia development
- no “competition” for GStreamer/NMM like media frameworks
- in KDE SVN: trunk/KDE/kdelibs/photon (or branches/work/kdelibs4_snapshot/photon)
Motivation
the user’s perspective

- A user should be able to playback any media without configuration steps
- “power users” want great flexibility
- additional multimedia hardware should be available to all applications without any further steps
- users need to decide what device to use for what purpose/program
- KDE should not get in the way of “media producers”
Motivation
the user’s perspective

- A user should be able to playback any media without configuration steps
- “power users” want great flexibility
- additional multimedia hardware should be available to all applications without any further steps
- users need to decide what device to use for what purpose/program
- KDE should not get in the way of “media producers”
Motivation
the user’s perspective

- A user should be able to playback any media without configuration steps
- “power users” want great flexibility
- additional multimedia hardware should be available to all applications without any further steps
- users need to decide what device to use for what purpose/program
- KDE should not get in the way of “media producers”
Motivation
the user’s perspective

- A user should be able to playback any media without configuration steps
- “power users” want great flexibility
- additional multimedia hardware should be available to all applications without any further steps
- users need to decide what device to use for what purpose/program
- KDE should not get in the way of “media producers”
Motivation

the user’s perspective

- A user should be able to playback any media without configuration steps
- “power users” want great flexibility
- additional multimedia hardware should be available to all applications without any further steps
- users need to decide what device to use for what purpose/program
- KDE should not get in the way of “media producers”
Motivation
the developer’s perspective

- Qt/KDE style API
  - developers need APIs that are straightforward, easy to use and understand
  - applications need a multimedia API that works on UNIX systems (including OS X) and Windows
  - ABI changes should not hinder KDE from using the newest version of some media framework
Motivation
the developer’s perspective

- Qt/KDE style API
- developers need APIs that are straightforward, easy to use and understand
- applications need a multimedia API that works on UNIX systems (including OS X) and Windows
- ABI changes should not hinder KDE from using the newest version of some media framework
Motivation
the developer’s perspective

- Qt/KDE style API
- developers need APIs that are straightforward, easy to use and understand
- applications need a multimedia API that works on UNIX systems (including OS X) and Windows
- ABI changes should not hinder KDE from using the newest version of some media framework
Motivation
the developer’s perspective

- Qt/KDE style API
- developers need APIs that are straightforward, easy to use and understand
- applications need a multimedia API that works on UNIX systems (including OS X) and Windows
- ABI changes should not hinder KDE from using the newest version of some media framework
Phonon Backends

Application

Phonon

Phonon–Xine

Xine
Phonon Backends

- Application
- Phonon
- Phonon–GStreamer
- GStreamer
Outline

What is Phonon?

Design of Phonon
Core Classes
Code Examples
User Visible Features
How to Write a Backend
The Core Classes

MediaObject

AudioPath

process audio in

send to

AudioEffect

send to

AudioOutput

VideoPath

process video in

send to

VideoEffect

send to

VideoOutput
The Core Classes

- MediaObject
  - process audio in
  - AudioPath
    - process audio in
    - AudioEffect
    - send to
    - AudioOutput
  - send to
  - VideoPath
    - process video in
    - VideoEffect
    - send to
    - VideoOutput
The Core Classes

MediaObject
  process audio in
  process video in

AudioPath
  send to
  AudioOutput

VideoPath
  send to
  VideoOutput

AudioEffect

VideoEffect
The Core Classes

MediaObject

- process audio in
- process video in

AudioPath

- send to AudioOutput

AudioEffect

VideoPath

- send to VideoOutput

VideoEffect

AudioOutput

VideoOutput
A Glance at the Phonon Classes

- BackendCapabilities
  - AbstractMediaProducer
    - MediaObject
    - ByteStream
    - AvCapture
  - AbstractAudioOutput
    - AudioOutput
    - AudioDataOutput
  - AbstractVideoOutput
    - VideoWidget
    - VideoDataOutput
  - AudioPath
  - VideoPath
  - AudioEffect
    - VolumeFaderEffect
  - VideoEffect

- NameDescriptionTuple
  - AudioOutputDevice
  - AudioCaptureDevice
  - VideoOutputDevice
  - VideoCaptureDevice
  - AudioEffectDescription
  - VideoEffectDescription

- Ui::VideoWidget
- Ui::SeekSlider
- Ui::VolumeSlider
- Ui::MediaControls
- Ui::EffectWidget
A Glance at the Phonon Classes

- BackendCapabilities
- AbstractMediaProducer
  - MediaObject
  - ByteStream
  - AvCapture
- AbstractAudioOutput
  - AudioOutput
  - AudioDataOutput
- AbstractVideoOutput
  - VideoWidget
  - VideoDataOutput
- AudioPath
- VideoPath
- AudioEffect
  - VolumeFaderEffect
- VideoEffect

- NameDescriptionTuple
- AudioOutputDevice
- AudioCaptureDevice
- VideoOutputDevice
- VideoCaptureDevice
- AudioEffectDescription
- VideoEffectDescription

- Ui::VideoWidget
- Ui::SeekSlider
- Ui::VolumeSlider
- Ui::MediaControls
- Ui::EffectWidget
A Glance at the Phonon Classes

- BackendCapabilities
- AbstractMediaProducer
  - MediaObject
  - ByteStream
  - AvCapture
- AbstractAudioOutput
  - AudioOutput
  - AudioDataOutput
- AbstractVideoOutput
  - VideoWidget
  - VideoDataOutput
- AudioPath
- VideoPath
- AudioEffect
  - VolumeFaderEffect
- VideoEffect

- NameDescriptionTuple
- AudioOutputDevice
- AudioCaptureDevice
- VideoOutputDevice
- VideoCaptureDevice
- AudioEffectDescription
- VideoEffectDescription

- Ui::VideoWidget
- Ui::SeekSlider
- Ui::VolumeSlider
- Ui::MediaControls
- Ui::EffectWidget
A Glance at the Phonon Classes

- BackendCapabilities
- AbstractMediaProducer
  - MediaObject
  - ByteStream
  - AvCapture
- AbstractAudioOutput
  - AudioOutput
  - AudioDataOutput
- AbstractVideoOutput
  - VideoWidget
  - VideoDataOutput
- AudioPath
- VideoPath
- AudioEffect
  - VolumeFaderEffect
- VideoEffect
- NameDescriptionTuple
- AudioOutputDevice
- AudioCaptureDevice
- VideoOutputDevice
- VideoCaptureDevice
- AudioEffectDescription
- VideoEffectDescription
- Ui::VideoWidget
- Ui::SeekSlider
- Ui::VolumeSlider
- Ui::MediaControls
- Ui::EffectWidget
A Glance at the Phonon Classes

- BackendCapabilities
- AbstractMediaProducer
  - MediaObject
  - ByteStream
  - AvCapture
- AbstractAudioOutput
  - AudioOutput
  - AudioDataOutput
- AbstractVideoOutput
  - VideoWidget
  - VideoDataOutput
- AudioPath
- VideoPath
- AudioEffect
  - VolumeFaderEffect
- VideoEffect

- NameDescriptionTuple
- AbstractAudioOutputDevice
- AbstractVideoOutputDevice
- AudioCaptureDevice
- VideoCaptureDevice
- AudioEffectDescription
- VideoEffectDescription
- Ui::VideoWidget
- Ui::SeekSlider
- Ui::VolumeSlider
- Ui::MediaControls
- Ui::EffectWidget
A Glance at the Phonon Classes

- BackendCapabilities
- AbstractMediaProducer
  - MediaObject
  - ByteStream
  - AvCapture
- AbstractAudioOutput
  - AudioOutput
  - AudioDataOutput
- AbstractVideoOutput
  - VideoWidget
  - VideoDataOutput
- AudioPath
- VideoPath
- AudioEffect
  - VolumeFaderEffect
- VideoEffect
- NameDescriptionTuple
- AudioOutputDevice
- AudioCaptureDevice
- VideoOutputDevice
- VideoCaptureDevice
- AudioEffectDescription
- VideoEffectDescription
- Ui::VideoWidget
- Ui::SeekSlider
- Ui::VolumeSlider
- Ui::MediaControls
- Ui::EffectWidget
A Glance at the Phonon Classes

- BackendCapabilities
- AbstractMediaProducer
  - MediaObject
  - ByteStream
  - AvCapture
- AbstractAudioOutput
  - AudioOutput
  - AudioDataOutput
- AbstractVideoOutput
  - VideoWidget
  - VideoDataOutput
- AudioPath
- VideoPath
- AudioEffect
  - VolumeFaderEffect
- VideoEffect

- NameDescriptionTuple
  - AudioOutputDevice
  - AudioCaptureDevice
  - VideoOutputDevice
  - VideoCaptureDevice
  - AudioEffectDescription
  - VideoEffectDescription

- Ui::VideoWidget
- Ui::SeekSlider
- Ui::VolumeSlider
- Ui::MediaControls
- Ui::EffectWidget
A Glance at the Phonon Classes

- BackendCapabilities
- AbstractMediaProducer
  - MediaObject
  - ByteStream
  - AvCapture
- AbstractAudioOutput
  - AudioOutput
  - AudioDataOutput
- AbstractVideoOutput
  - VideoWidget
  - VideoDataOutput
- AudioPath
- VideoPath
- AudioEffect
  - VolumeFaderEffect
- VideoEffect
- NameDescriptionTuple
  - AudioOutputDevice
  - AudioCaptureDevice
  - VideoOutputDevice
  - VideoCaptureDevice
  - AudioEffectDescription
  - VideoEffectDescription
- Ui::VideoWidget
- Ui::SeekSlider
- Ui::VolumeSlider
- Ui::MediaControls
- Ui::EffectWidget
Outline

What is Phonon?

Design of Phonon
  Core Classes
  Code Examples
  User Visible Features
  How to Write a Backend
Audio Playback

the simplest case

SimplePlayer

```cpp
SimplePlayer* player = new SimplePlayer;
player->play( "file:///home/user/song.ogg" );
```

seek/pause/stop

```cpp
player->seek( milliseconds );
player->pause();
player->stop();
```

volume

```cpp
float volume = player->volume();
volume *= 0.5;
player->setVolume( volume );
```
Audio Playback
the simplest case

SimplePlayer

SimplePlayer* player = new SimplePlayer;
player->play( "file:///home/user/song.ogg" );

seek/pause/stop

player->seek( milliseconds );
player->pause();
player->stop();

volume

float volume = player->volume();
volume *= 0.5;
player->setVolume( volume );
Audio Playback

the simplest case

**SimplePlayer**

```cpp
SimplePlayer* player = new SimplePlayer;
player->play( "file:///home/user/song.ogg" );
```

**seek/pause/stop**

```cpp
player->seek( milliseconds );
player->pause();
player->stop();
```

**volume**

```cpp
float volume = player->volume();
volume *= 0.5;
player->setVolume( volume );
```
Audio Playback
with more control

define the output

```cpp
output = new AudioOutput;
output->setCategory( Phonon::MusicCategory );
path1 = new AudioPath;
path1->addOutput( output );
```

simple MediaObject use

```cpp
media1 = new MediaObject;
media1->addAudioPath( path1 );
media1->setUrl( "file:///home/user/song.ogg" );
media1->play();
```
Audio Playback
with more control

**define the output**

```cpp
output = new AudioOutput;
output->setCategory( Phonon::MusicCategory );
path1 = new AudioPath;
path1->addOutput( output );
```

**simple MediaObject use**

```cpp
media1 = new MediaObject;
media1->addAudioPath( path1 );
media1->setUrl( "file:///home/user/song.ogg" );
media1->play();
```
Audio Playback  
with more control  

define the output

```cpp
output = new AudioOutput;
output->setCategory( Phonon::MusicCategory );
path1 = new AudioPath;
path1->addOutput( output );
```

simple MediaObject use

```cpp
media1 = new MediaObject;
media1->addAudioPath( path1 );
media1->setUrl( "file://home/user/song.ogg" );
media1->play();
```
Audio Playback with more control

define the output

```cpp
output = new AudioOutput;
output->setCategory( Phonon::MusicCategory );
path1 = new AudioPath;
path1->addOutput( output );
```

**Diagram:**

```
MediaObject -> AudioPath -> AudioOutput
```

simple MediaObject use

```cpp
media1 = new MediaObject;
media1->addAudioPath( path1 );
media1->setUrl( "file:///home/user/song.ogg" );
media1->play();
```
Audio Playback
adding a Fader

recall the AudioPath

```cpp
path1 = new AudioPath;
```

insert the Fader

```cpp
fader1 = new VolumeFaderEffect;
fader1->setVolume( 1.0 );
path1->insertEffect( fader1 );
```
Audio Playback
adding a Fader

**recall the AudioPath**

```cpp
path1 = new AudioPath;
```

![Diagram]

**insert the Fader**

```cpp
fader1 = new VolumeFaderEffect;
fader1->setVolume( 1.0 );
path1->insertEffect( fader1 );
```
Audio Playback
adding a Fader

recall the AudioPath

```cpp
path1 = new AudioPath;
```

insert the Fader

```cpp
fader1 = new VolumeFaderEffect;
fader1->setVolume( 1.0 );
path1->insertEffect( fader1 );
```
Audio Playback
prepare next MediaObject for Crossfade

```cpp
path2 = new AudioPath;
path2->addOutput( output );
media2 = new MediaObject;
media2->addAudioPath( path2 );
media2->setUrl( "file:///home/user/moremusic.ogg" );
fader2 = new VolumeFaderEffect;
fader2->setVolume( 0.0 );
path2->insertEffect( fader2 );
```
Audio Playback
prepare next MediaObject for Crossfade

```cpp
path2 = new AudioPath;
path2->addOutput( output );
media2 = new MediaObject;
media2->addAudioPath( path2 );
media2->setUrl( "file:///home/user/moremusic.ogg" );
fader2 = new VolumeFaderEffect;
fader2->setVolume( 0.0 );
path2->insertEffect( fader2 );
```
Audio Playback
Crossfade 2s before the first song ends

```cpp
void MyPlayer::crossfade(long remaining)
{
    fader1->fadeOut(remaining);
    fader2->fadeIn(remaining);
    media2->play();
}
```

```cpp
media1->setAboutToFinishTime(2000);
connect( media1, SIGNAL( aboutToFinish(long) ), SLOT( crossfade(long) ) );
```
Outline

What is Phonon?

Design of Phonon
  Core Classes
  Code Examples
  User Visible Features
  How to Write a Backend
The Mixer

- Every `AudioOutput` has a volume control
- Volume can be read and written using IPC
- Central “desktop-mixer” can then control the volume of all Phonon applications
- To not let the number of volume controls explode they are be combined into the categories Notifications, Music, Movies, Games and Communication
Devices

- central place for device selection
- select device per category
- applications can override the selection

Select the device preference to be used for audio output. Applications use the given categories below for their output devices (for example all notification sounds from Kopete, kwin and so on are in the "Notifications" category, while media players use the "Music" and "Video" categories depending on the media). This makes it possible to choose the device preference separately for every category.
Devices

- central place for device selection
- select device per category
- applications can override the selection

Select the device preference to be used for audio output. Applications use the given categories below for their output devices (for example all notification sounds from Kopete, kwin and so on are in the "Notifications" category, while media players use the "Music" and "Video" categories depending on the media). This makes it possible to choose the device preference separately for every category.

Notifications:
- internal Soundcard
  - USB Headset

Music:
- internal Soundcard
  - USB Headset

Video:
- internal Soundcard
  - USB Headset

Communication:
- internal Soundcard
Plug Your USB Headset

- hotplugging a device will change outputs to the new device if it is preferred
- consider VoIP
  - call comes in
  - you answer the call using the internal soundcard
  - plug in the USB headset
  - notification shows that the device has been switched
  - you can use your headset for the conversation now
Configuration

or: Hiding tedious configuration work from the user

► applications should not need to configure the sound system, selecting the device to use is enough
► one central place for configuration
► configuration options include
  ► default video output device
  ► default capture devices
  ► whether to use a soundserver (the soundserver will then be started by KDE and made available as a device)
  ► backend specific options
► For system wide integration a shared configuration is needed for the cases where hardware mixing or ALSA dmix is unavailable
Network and Special Routing
or: How to integrate NMM

- NMM provides for a high degree of network transparency
- Out of scope for the Phonon API
- IPC hooks in the NMM backend
- NMM-Phonon control application
Outline

What is Phonon?

Design of Phonon
Core Classes
Code Examples
User Visible Features
How to Write a Backend
Where do I Start?

- Many classes to implement
- Starting point: `Backend` class
  - Mediaframework initialization
  - Works as factory for all other classes
- Then implement `MediaObject`, `AudioPath` and `AudioOutput`: enough for audio playback
Main Audio Playback Classes

**MediaObject**
- read and decode mediafile
- play, pause, stop, seek, tick
- takes multiple Audio- and VideoPaths

**AudioPath**
- defines routing (and signal processing)
- takes multiple AudioOutputs

**AudioOutput**
- defines audio sink
- software volume control
Main Audio Playback Classes

**MediaObject**
- read and decode mediafile
- play, pause, stop, seek, tick
- takes multiple Audio- and VideoPaths

**AudioPath**
- defines routing (and signal processing)
- takes multiple AudioOutputs

**AudioOutput**
- defines audio sink
- software volume control
## Main Audio Playback Classes

### MediaObject
- read and decode mediafile
- play, pause, stop, seek, tick
- takes multiple **Audio- and VideoPaths**

### AudioPath
- defines routing (and signal processing)
- takes multiple **AudioOutputs**

### AudioOutput
- defines audio sink
- software volume control
MediaObject::setUrl
  ▶ start decoding the media to fill audio and video buffers
  ▶ emit the length signal
  ▶ prepare for calls to hasVideo(), seekable(), availableAudioStreams(), availableVideoStreams(), availableSubtitleStreams and totalTime()

MediaObject::add{Audio, Video}Path
  ▶ preprocess audio/video buffers with effects defined for the path
  ▶ prepare for output to the audio device if the AudioOutput is known
- AbstractMediaProducer::play()
  - the backend should start playing immediately
  - the tick signal should be emitted - if possible without polling
- effect change while playing
  - buffers will make reaction sluggish
  - first priority: no dropouts
  - second: no latency - overwrite as much of the buffers with changed audio/video data as possible
subclasses of AbstractMediaProducer

- **ByteStream**
  - same as MediaObject
  - instead of reading media data located using a URL → media data is passed (streamed) from the application (or frontend)

- **AvCapture**
  - Backend i18n(“anbieten”) audio and video capture devices
  - one video and/or one audio capture device can be selected
  - implicitly synchronized
Two Designs

- **1:1**
  - Phonon objects create and hold objects of the media framework
  - good when there's a 1:1 mapping between Phonon classes and media framework classes

- **Phonon objects as description**
  - Phonon objects describe what the application wants
  - Backend object(s) look at what the user wants and wire media framework objects accordingly
Summary

using Multimedia functionality in KDE 4 will be easier for

▶ developers
▶ users

Outlook

▶ network interfaces, DVD-/TV-Support, OSD
▶ backend development
▶ review
▶ more tests, esp. backend “certification” tests