Multimedia API for KDE 4

Matthias Kretz

former KView maintainer

workend on aRts/KDE Multimedia stuff

Student of Physics and Computer Science at the University of Heidelberg

aKademy 2005, Developers Conference
Outline

1. Introduction
   - Where We’re Coming From
   - Media Frameworks

2. KDE Multimedia Efforts
   - aKademy ’04
   - Why KDEMM

3. The design of KDEMM at this point
   - new API for KDE4
   - User Visible Features
   - API Usage
Introduction

Where We’re Coming From

Media Frameworks

KDE Multimedia Efforts

aKademy ’04

Why KDEMM

The design of KDEMM at this point

new API for KDE4

User Visible Features

API Usage

Summary

Appendix
For KDE 2 aRts was adopted as the soundserver and mediat framework
- best available framework at that time
- works well for a lot of people
- has a rich featureset
- the possibilities of MCOP were never really exploited in KDE
Known Problems

- not maintained since a considerable time
- design issues
- debugging artsd or aRts applications is hard
- new scheduler core made a lot of people reject aRts because of its glib usage
- using aRts in your application isn’t easy enough
- developing for the aRts core is even harder
- not enough adoption outside of KDE
- whatever you’re favorite complaint is...
Multimedia API for KDE 4

Matthias Kretz

Introduction

Where We’re Coming From

Media Frameworks

KDE Multimedia Efforts

aKademy ’04

Why KDEMM

The design of KDEMM at this point

new API for KDE4

User Visible Features

API Usage

Summary

Appendix
libxine

- mature
- good support for many formats
- playback-engine only
gstreamer

- mature
- good support for many formats
- clean C++ API
- completely network transparent
- supports sharing of media resources that seems very attractive for desktop usage (example: the same music playing in two rooms)

Michael Repplinger: “Since we are, from our side, very interested that NMM becomes a backend for the multimedia-part in KDE4, I will try to create a working version as fast as possible.”
NMM
Network Multimedia Middleware

- clean C++ API
- completely network transparent
- supports sharing of media resources that seems very attractive for desktop usage (example: the same music playing in two rooms)

Michael Repplinger: “Since we are, from our side, very interested that NMM becomes a backend for the multimedia-part in KDE4, I will try to create a working version as fast as possible.”
Multimedia API for KDE 4

Matthias Kretz

Introduction
Where We’re Coming From
Media Frameworks

KDE Multimedia Efforts
aKademy ’04
Why KDEMM

The design of KDEMM at this point
new API for KDE4
User Visible Features
API Usage

Summary
Appendix

- Helix
- mplayer
- aKode
- DirectX?
Multimedia API for KDE 4
Matthias Kretz

Outline

1 Introduction
   - Where We’re Coming From
   - Media Frameworks

2 KDE Multimedia Efforts
   - aKademy ’04
   - Why KDEMM

3 The design of KDEMM at this point
   - new API for KDE4
   - User Visible Features
   - API Usage

Appendix

Summary
The largest gathering of KDE Multimedia developers in history

We had talks about media frameworks

We talked about our options:

1. Adopt one of the existing media frameworks as the KDE framework
2. Adopt one of the existing media frameworks and write an API that is more suited to KDE developers (C++, signals/slots, Qt style API) around it
3. If we write our own API, why not make the implementation of the API completely independent, thereby allowing different media frameworks to implement the API’s functionality
The largest gathering of KDE Multimedia developers in history

We had talks about media frameworks

We talked about our options:

1. Adopt one of the existing media frameworks as the KDE framework

2. Adopt one of the existing media frameworks and write an API that is more suited to KDE developers (C++, signals/slots, Qt style API) around it

3. If we write our own API, why not make the implementation of the API completely independent, thereby allowing different media frameworks to implement the API’s functionality
The largest gathering of KDE Multimedia developers in history

We had talks about media frameworks

We talked about our options:

1. Adopt one of the existing media frameworks as the *KDE framework*

2. Adopt one of the existing media frameworks and write an API that is more suited to KDE developers (C++, signals/slots, Qt style API) around it

3. If we write our own API, why not make the implementation of the API completely independent, thereby allowing different media frameworks to implement the API’s functionality
The largest gathering of KDE Multimedia developers in history

We had talks about media frameworks

We talked about our options:

1. Adopt one of the existing media frameworks as *the KDE framework*

2. Adopt one of the existing media frameworks and write an API that is more suited to KDE developers (C++, signals/slots, Qt style API) around it

3. If we write our own API, why not make the implementation of the API completely independent, thereby allowing different media frameworks to implement the API’s functionality
What is KDEMM?

- It’s a boring name I invented to have some name to call my project. I’m open for better name suggestions
- It’s a library that is to reside in kdelibs (currently in branches/work/kdemm)
- It’s targeted for release with KDE 4
A first version of KDEMM
make_it_cool branch in CVS
Did not provide all the features I would have liked to support
Design was limited when it came to video functionality
KDEMM was fully functional with two working backends (aRts and aKode)
The New KDEMM
or: how university can be useful sometimes

I was unsatisfied with the design of KDEMM
At university I was allowed to do a redesign of KDEMM as an internship in software engineering
I’d like to give credit to my supervisor Lars Borner who helped me a lot to be able to apply the concepts of software engineering on my API design
Multimedia API for KDE 4
Matthias Kretz

Outline

1. Introduction
   - Where We’re Coming From
   - Media Frameworks

2. KDE Multimedia Efforts
   - aKademy ’04
   - Why KDEMM

3. The design of KDEMM at this point
   - new API for KDE4
   - User Visible Features
   - API Usage
What we get with KDEMM

1. High Level Multimedia API is a good thing as it facilitates the integration of media capabilities into all kinds of programs (this motivates alternatives 2 or 3)
2. No lock-in to one specific media framework
3. Choice
4. Independence from ABI changes of the media frameworks
What we get with KDEMM

1. High Level Multimedia API is a good thing as it facilitates the integration of media capabilities into all kinds of programs (this motivates alternatives 2 or 3)

2. No lock-in to one specific media framework

3. Choice

4. Independence from ABI changes of the media frameworks
What we get with KDEMM

1. High Level Multimedia API is a good thing as it facilitates the integration of media capabilities into all kinds of programs (this motivates alternatives 2 or 3)
2. No lock-in to one specific media framework
3. Choice
4. Independence from ABI changes of the media frameworks
What we get with KDEMM

1. High Level Multimedia API is a good thing as it facilitates the integration of media capabilities into all kinds of programs (this motivates alternatives 2 or 3)
2. No lock-in to one specific media framework
3. Choice
4. Independence from ABI changes of the media frameworks
1. Will KDEMM slow down the multimedia experience?

- KDEMM doesn’t target pro-audio (we cannot guarantee the performance of the media framework the backend uses)
- most critical is play/pause/stop
- thin wrapper for critical calls
- there’s no overhead for the decoding process - only setup calls to the media framework might take a few cycles longer (i.e. while playing back a media file the CPU load is the same with or without the KDEMM layer)
Common Questions

1. Will KDEMM slow down the multimedia experience?
   - KDEMM doesn’t target pro-audio (we cannot guarantee the performance of the media framework the backend uses)
   - most critical is play/pause/stop
   - thin wrapper for critical calls
   - there’s no overhead for the decoding process - only setup calls to the media framework might take a few cycles longer (i.e. while playing back a media file the CPU load is the same with or without the KDEMM layer)
Common Questions

What about link times?
- startup link time could actually be less since the bigger part of the linking is done after startup (dlopen) (KDEMM libs should be lighter on an app than the media framework libs)
- done right, the dlopen comes at a point where the user doesn’t notice

Will KDEMM add more instability (crashes)?
- KDEMM is not supposed to become \textit{that} complicated
- in the end we might be able to work around common pitfalls in a media framework for once in the backend
What about link times?

- startup link time could actually be less since the bigger part of the linking is done after startup (dlopen) (KDEMM libs should be lighter on an app than the media framework libs)
- done right, the dlopen comes at a point where the user doesn’t notice

Will KDEMM add more instability (crashes)?

- KDEMM is not supposed to become *that* complicated
- in the end we might be able to work around common pitfalls in a media framework for once in the backend
Common Questions

2 What about link times?
- startup link time could actually be less since the bigger part of the linking is done after startup (dlopen) (KDEMM libs should be lighter on an app than the media framework libs)
- done right, the dlopen comes at a point where the user doesn’t notice

3 Will KDEMM add more instability (crashes)?
- KDEMM is not supposed to become *that* complicated
- in the end we might be able to work around common pitfalls in a media framework for once in the backend
2 What about link times?
   • startup link time could actually be less since the bigger part of the linking is done after startup (dlopen) (KDEMM libs should be lighter on an app than the media framework libs)
   • done right, the dlopen comes at a point where the user doesn’t notice

3 Will KDEMM add more instability (crashes)?
   • KDEMM is not supposed to become *that* complicated
   • in the end we might be able to work around common pitfalls in a media framework for once in the backend
Common Questions

4. Won’t KDEMM users miss out on a lot of features of the media framework?
   - They will not be able to use all features the media framework provides, yes
   - The real question is, will the majority of KDE applications be (easily) able to achieve what they want

5. If we allow multiple backends, won’t we see a lot of half-finished backends instead of only one rock-solid adaptor?
   - Now that is mostly a question of manpower, I’d say
   - amaroK seems to have had this problem and then they disabled most backends to stabilize a few of them
Common Questions

4. Won’t KDEMM users miss out on a lot of features of the media framework?
   - They will not be able to use all features the media framework provides, yes
   - The real question is, will the majority of KDE applications be (easily) able to achieve what they want

5. If we allow multiple backends, won’t we see a lot of half-finished backends instead of only one rock-solid adaptor?
   - Now that is mostly a question of manpower, I’d say
   - amaroK seems to have had this problem and then they disabled most backends to stabilize a few of them
Common Questions

4 Won’t KDEMM users miss out on a lot of features of the media framework?
   - They will not be able to use all features the media framework provides, yes
   - The real question is, will the majority of KDE applications be (easily) able to achieve what they want

5 If we allow multiple backends, won’t we see a lot of half-finished backends instead of only one rock-solid adaptor?
   - Now that is mostly a question of manpower, I’d say
   - amaroK seems to have had this problem and then they disabled most backends to stabilize a few of them
Common Questions

4. Won’t KDEMM users miss out on a lot of features of the media framework?
   - They will not be able to use all features the media framework provides, yes
   - The real question is, will the majority of KDE applications be (easily) able to achieve what they want

5. If we allow multiple backends, won’t we see a lot of half-finished backends instead of only one rock-solid adaptor?
   - Now that is mostly a question of manpower, I’d say
   - amaroK seems to have had this problem and then they disabled most backends to stabilize a few of them
Multimedia API for KDE 4

Matthias Kretz

Outline

1. Introduction
   - Where We’re Coming From
   - Media Frameworks

2. KDE Multimedia Efforts
   - aKademy ’04
   - Why KDEMM

3. The design of KDEMM at this point
   - new API for KDE4
   - User Visible Features
   - API Usage

Summary

Appendix
Requirements for a KDE Multimedia API

- Qt/KDE style API
- easy to use and understand
- API independent from the used media framework
- fully functional on all platforms
- solve integration and configuration for the KDE user
What we’re not trying to do

- write another media framework
- create a framework for pro-audio tools
- create a UNIX wide solution
Architecture

- layered architecture (Bridge pattern)
  - layer to be used by the application
  - interface layer to be implemented using a media framework
- allows for additions/changes while keeping (binary) compatibility
Multimedia API for KDE 4
Matthias Kretz

Introduction
Where We're Coming From
Media Frameworks

KDE Multimedia Efforts
aKademy '04
Why KDEMM

The design of KDEMM at this point
new API for KDE4
User Visible Features
API Usage

Summary
Appendix

Architecture
Components
Architecture
Components

KDE Multimedia Application

kdemm UI

kdemm core

kdemm interfaces

backend

media framework

kdemm effect widgets
Multimedia API for KDE 4
Matthias Kretz

Introduction
Where We're Coming From
Media Frameworks
KDE Multimedia Efforts
aKademy '04
Why KDEMM

The design of KDEMM at this point
new API for KDE4
User Visible Features
API Usage

Summary
Appendix

Architecture
Components

KDE Multimedia Application
kdemm UI
kdemm core
kdemm interfaces
backend
kdemm effect widgets
media framework

Architecture Components
Multimedia API for KDE 4

Matthias Kretz

Introduction
Where We're Coming From
Media Frameworks

KDE Multimedia
Efforts
aKademy '04
Why KDEMM

The design of KDEMM at this point
new API for KDE4
User Visible Features
API Usage

Summary
Appendix

Architecture
Components

KDE Multimedia Application

kdemm UI

kdemm core

kdemm interfaces

backend

media framework

kdemm effect widgets
The Core

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

Multimedia API for KDE 4
Matthias Kretz

Introduction
Where We're Coming From
Media Frameworks
KDE Multimedia Efforts
aKademy '04
Why KDEMM
The design of KDEMM at this point
new API for KDE 4
User Visible Features
API Usage
Summary
Appendix

The Core

VideoEffect

VideoPath

send to

VideoOutput

process video in

MediaObject

AudioPath

send to

AudioOutput

process audio in

AudioEffect

AudioPath

send to

VideoOutput

process video in
The Core

AudioEffect

process audio in

AudioOutput

send to

VideoPath

process video in

VideoOutput

send to

MediaObject

VideoEffect
The Core

Multimedia API for KDE 4
Matthias Kretz

Introduction
Where We're Coming From
Media Frameworks
KDE Multimedia Efforts
aKademy '04
Why KDEMM
The design of KDEMM at this point
new API for KDE4
User Visible Features
API Usage
Summary
Appendix

AudioOutput
VideoOutput

VideoEffect
send to

VideoPath
send to

AudioPath
send to

AudioEffect

MediaObject

process audio in

process video in
Introduction
Where We're Coming From
Media Frameworks
KDE Multimedia
Efforts
aKademy '04
Why KDEMM
The design of KDEMM at this point
new API for KDE4
User Visible Features
API Usage
Summary
Appendix
Media Object Construction

Factory::self() : Factory

mo : MediaObject

create(url : KURL, parent : QObject) : createIFace() : KDEMM.ifaces.MediaProducer

createMediaObject(url : KURL, parent : QObject) : KDEMM.ifaces.MediaObject

mo : MediaProducer

create(iface : KDEMM.ifaces.MediaProducer, parent : QObject)

setupIFace()

setupIFace()
Internal Use of Interface Objects
KDEMM UI

- position slider
- volume fader
- effects dialog
  - complete dialog for adding/removing effects
  - dialogs for all effects to edit parameters
- video widget
- player controls as toolbar
In order to get a really clean separation (to make apps without GUI able to use kdemmcore) the following is needed:

- kdemmcore may not depend on QtGui
- the part of kdemmifaces that provides the core interfaces may not depend on QtGui
- the backend implementation of that part of kdemmifaces may not depend on QtGui
- the media framework used by the backend implementation needs to be separated into GUI dependent and independent parts
Outline

1. Introduction
   - Where We’re Coming From
   - Media Frameworks

2. KDE Multimedia Efforts
   - aKademy ’04
   - Why KDEMM

3. The design of KDEMM at this point
   - new API for KDE4
   - User Visible Features
   - API Usage
Central place for configuration

software mixing:
  - Try to autodetect (look at driver or open device)
  - If software mixing is needed set up dmix or soundserver

Configuration will then work for all KDEMM apps

For system wide integration a shared configuration is needed
Audio Volume Controls

- Every **AudioOutput** has a volume control
- Volume can be read and written using IPC
- Central “desktop-mixer” can then control the volume of all KDEMM applications
- To not let the number of volume controls explode they should be combined into categories like Notifications, Music, Movies, Games, . . .
Network and Special Routing
or: How to integrate NMM

- NMM provides for a high degree of network transparency
- Too complicated to reflect in the KDEMM API
- IPC hooks in the NMM backend
- NMM-KDEMM control application
Introduction

Where We’re Coming From

Media Frameworks

KDE Multimedia Efforts

aKademy ’04

Why KDEMM

The design of KDEMM at this point

new API for KDE4

User Visible Features

API Usage
Simple KDEMM Test Program

```cpp
m_media = new MediaObject( url, this );
if( m_media->hasVideo() )
{
    m_vout = new VideoOutput( this );
    m_vpath = new VideoPath( m_vout );
    m_vpath->addOutput( m_vout );
    m_media->addVideoPath( m_vpath );
}

m_media->setTickInterval( 100 );

stateChanged( m_media->state() );
```
Simple KDEMM Test Program

```cpp
connect( m_pause, SIGNAL( clicked() ),
         m_media, SLOT( pause() ) );
connect( m_play, SIGNAL( clicked() ),
         m_media, SLOT( play() ) );
connect( m_stop, SIGNAL( clicked() ),
         m_media, SLOT( stop() ) );
connect( m_media, SIGNAL( tick( long ) ),
         SLOT( tick( long ) ) );
length( m_media->totalTime() );
connect( m_media, SIGNAL( length( long ) ),
         SLOT( length( long ) ) );
connect( m_media, SIGNAL( stateChanged( KDEMM::State, KDEMM::State ) ),
         SLOT( stateChanged( KDEMM::State ) ) );
connect( m_media, SIGNAL( finished() ),
         qApp, SLOT( quit() ) );
```
Simple KDEMM Test Program

- play/pause/stop
- DCOP volume control
Summary

- Developing multimedia applications under KDE will be much simplified
- Using multimedia elements in standard (non-multimedia) KDE applications will be much simplified

Outlook
- Most of the interfaces are still to be defined
- The next important step is to get at least one backend implementation done
- KDEMM needs review, a nicer name and more developers interested in getting everything ready for KDE 4
- (Unit) Tests need to be created so that backends can be tested for conformance
I’m not sure KDEMM qualifies as it depends on Qt and KDE libraries. If it still does, this is what I think:

- (2.1.1) Network transparency is backend dependent
  → only one or two backends might qualify (√)
- (2.2.1) Specifying and URL is supported, streaming a media file using `QByteArray` is supported √
- (2.2.2) Should be possible, and can be supported for all backends implementing the `ByteStream` interface √
- (2.2.3) not sure what this means
- (2.2.4) √
- (2.2.5) √
(2.3.1) I’d say this requirement is impossible as long as the Linux Kernel doesn’t provide means for hard real-time applications to work easily, and without a lot of care when developing the client application. We need to talk about this one.

(2.3.2) Again, this might not be possible on Linux.

(2.3.3) KDEMM has it, but it probably doesn’t do it in < 20 ms

(2.3.4) backend dependent (√)

(2.4.1) √

(2.4.2) √

(2.4.3) not planned, but shouldn’t be hard to do

(2.5.1) not planned, but shouldn’t be hard to do

(2.6.1) √