



MeeGo™



# Разработка мультимедийных приложений для MeeGo

Иванов Сергей,  
ННГУ ВМК ФОТ БС



# Библиотеки для разработки

- **Gstreamer** – фреймворк написанный с использованием библиотеки Glib
- **FFmpeg** – фреймворк на основе библиотек libavcodec и libavformat
- **Media Application Framework(MAFW)** – фреймворк основанный на gstreamer для работы с мультимедийными сервисами в Maemo/MeeGo
- **Phonon** – модуль к библиотеке QT для работы с мультимедиа



# Gstreamer

**Gstreamer** - мультимедийный фреймворк написанный на языке программирования C с использованием библиотеки Glib. **Он установлен в MeeGo по умолчанию.** Эта библиотека является основой для большинства мультимедийных приложений, таких как видео-редакторы, медиа-плееры, программы записи и т.д.



# gstreamer tools

gst-inspect  
gst-launch  
gst-editor

media player

VoIP & video conferencing

streaming server

video editor

(...)

## gstreamer core framework

### pipeline architecture



media agnostic  
base classes  
message bus  
media type negotiation  
plugin system  
utility libraries  
language bindings

protocols  
- file:  
- http:  
- rtsp:  
- ...

sources  
- alsa  
- v4l2  
- tcp/udp  
- ...

formats  
- avi  
- mp4  
- ogg  
- ...

codecs  
- mp3  
- mpeg4  
- vorbis  
- ...

filters  
- converters  
- mixers  
- effects  
- ...

sinks  
- alsa  
- xvideo  
- tcp/udp  
- ...

3rd party plugins

## gstreamer plugins

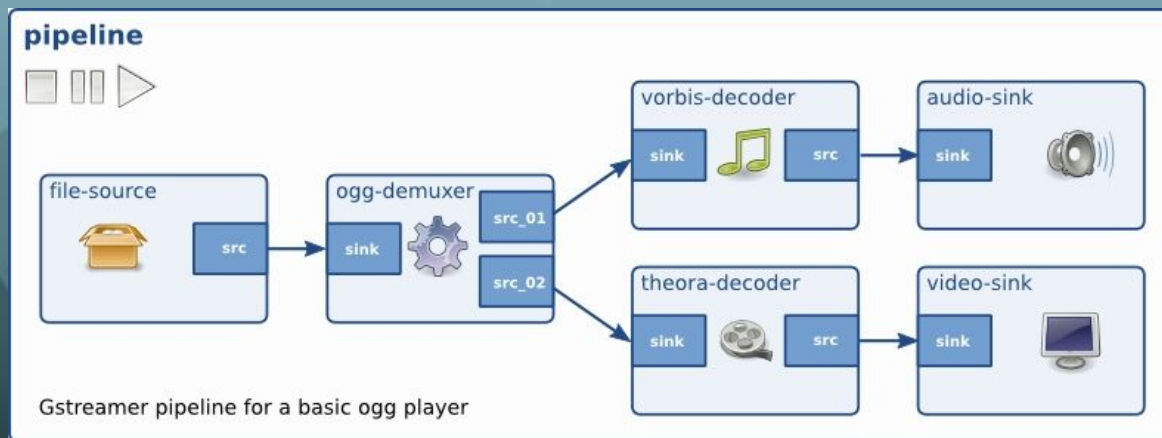
gstreamer includes over 150 plugins

# multimedia applications

## 3rd party plugins

**Основа GStreamer - наборы плагинов**, которые позволяют использовать различные компоненты в составе программы, расширяя функциональность. **Принцип работы GStreamer - конвейер**, состоящий из различных плагинов. В конвейере происходит преобразование потоков данных.

Простейший **конвейер** укладывается в формулу «**Входящий поток - Обработка - Исходящий поток**». Усложнение и, как следствие, расширение функционала происходит во втором звене. Отдельные плагины различаются по выполняемым ими функциям. Для выполнения поставленной задачи плагины отбираются в определенной последовательности, которая и составляет конвейер.



Существует более 150 плагинов и постоянно пишутся новые



# Классификация плагинов Gstreamer:

- **protocols handling**
- **sources:** аудио и видео потоки
- **formats:** микшеры, демикшеры, субтитры и т.д
- **codecs:** кодеки и декодеры
- **filtres:** конверторы, микшеры, эффекты и т.д.
- **sinks:** аудио и видео потоки



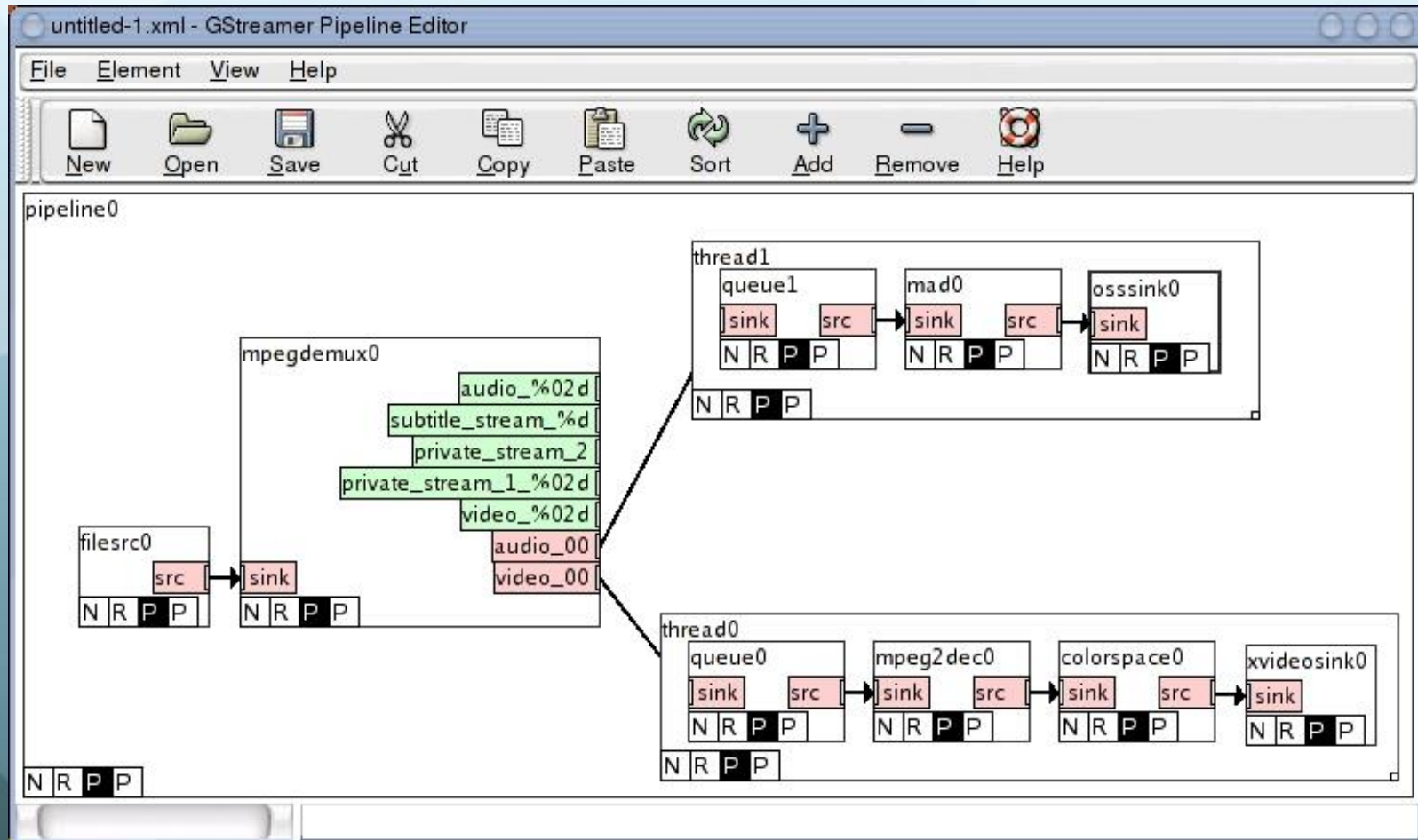
# Программы для работы с Gstreamer

1. **Gst-Inspect** – выводит информацию об установленных плагинах
2. **Gst-Launch** – запускает конвейер(pipeline)
3. **Gst-Editor** – позволяет под “графикой” создавать конвейеры из блоков
4. **Gst-Xmllaunch** – запускает созданные конвейеры с помощью gst-editor
5. **Gst-Visualise** – визуализация при воспроизведении аудио потока



# Gst-Editor

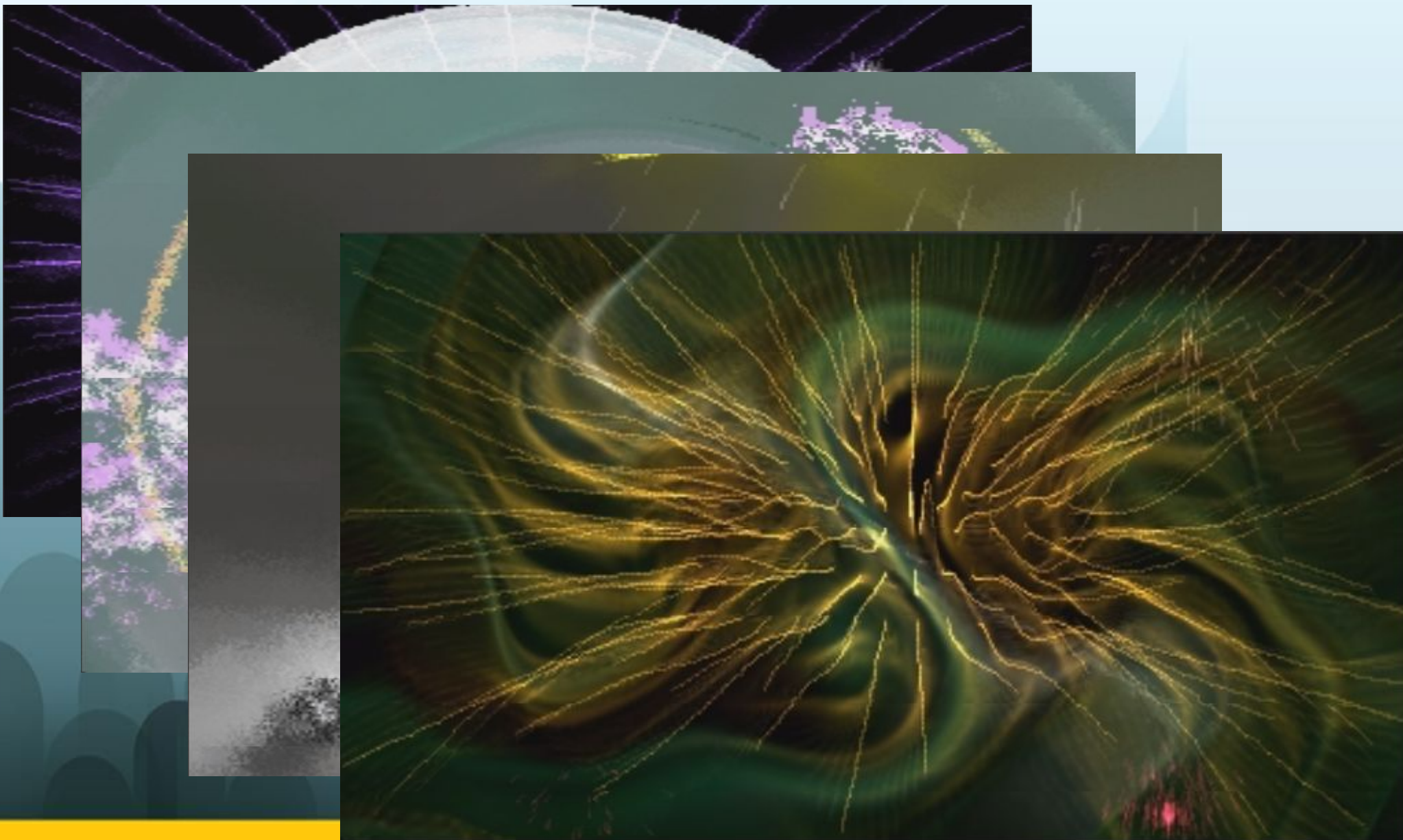
Графический редактор конвейеров





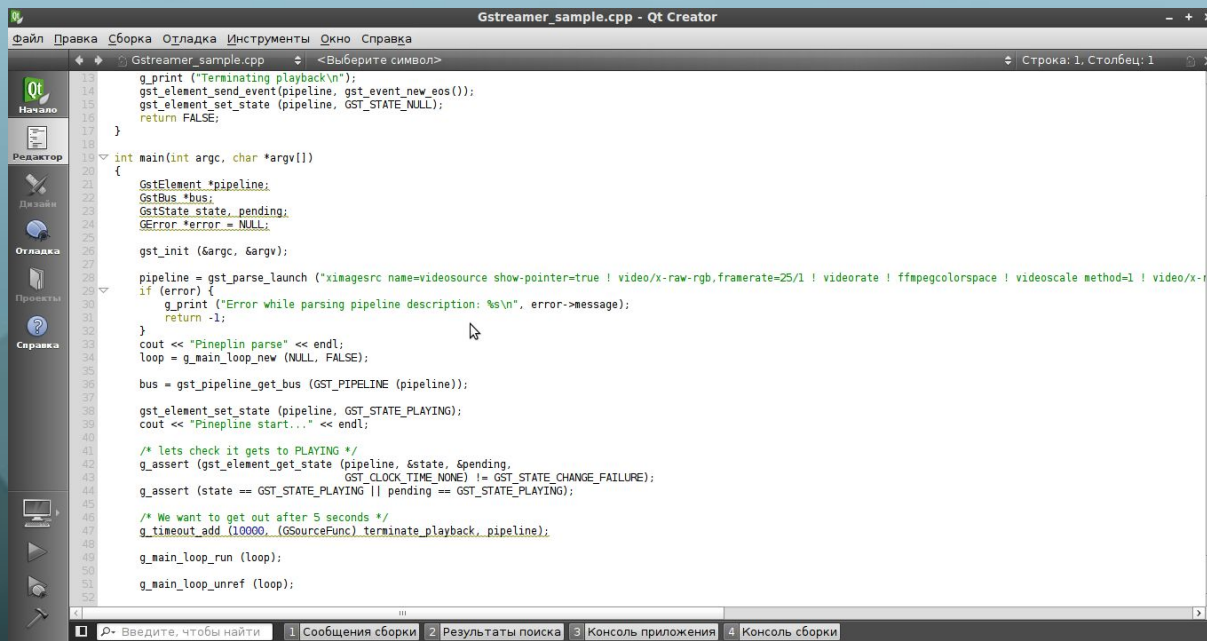
# Gst-Visualise

Эффекты при воспроизведении аудио



# Использование в своих программах

- Вызов существующих приложений с необходимыми параметрами
- Вставка в свой код на C/C++, Python и др.



```
Gstreamer_sample.cpp - Qt Creator
Файл  Правка  Сборка  Отладка  Инструменты  Окно  Справка
Gstreamer_sample.cpp  <Выберите символ>  Строка: 1, Столбец: 1
13  g_print ("Terminating playback\n");
14  gst_element_send_event(pipeline, gst_event_new_eos());
15  gst_element_set_state (pipeline, GST_STATE_NULL);
16  return FALSE;
17  }
18
19  int main(int argc, char *argv[])
20  {
21      GstElement *pipeline;
22      GstBus *bus;
23      GstState state, pending;
24      GError *error = NULL;
25
26      gst_init (&argc, &argv);
27
28      pipeline = gst_parse_launch ("ximagesrc name=videosource show-pointer=true ! video/x-raw-rgb,framerate=25/1 ! videorate ! ffmpegcolorspace ! videoscale method=1 ! video/x-raw,framerate=25/1 ! videoconvert ! video/x-raw,framerate=25/1 ! autovideosink");
29      if (error) {
30          g_print ("Error while parsing pipeline description: %s\n", error->message);
31          return -1;
32      }
33      cout << "Pipeline parse" << endl;
34      loop = g_main_loop_new (NULL, FALSE);
35
36      bus = gst_pipeline_get_bus (GST_PIPELINE (pipeline));
37
38      gst_element_set_state (pipeline, GST_STATE_PLAYING);
39      cout << "Pipeline start..." << endl;
40
41      /* lets check it gets to PLAYING */
42      g_assert (gst_element_get_state (pipeline, &state, &pending,
43                                     GST_CLOCK_TIME_NONE) != GST_STATE_CHANGE_FAILURE);
44      g_assert (state == GST_STATE_PLAYING || pending == GST_STATE_PLAYING);
45
46      /* We want to get out after 5 seconds */
47      g_timeout_add (10000, (GSourceFunc) terminate_playback, pipeline);
48
49      g_main_loop_run (loop);
50
51      g_main_loop_unref (loop);
52  }
```



# FFmpeg

**FFmpeg** - кросс-платформенное решение для записи, конвертации и воспроизведения аудио и видео.

Принцип работы FFmpeg отличается от GStreamer, этот фреймворк также является весьма популярным инструментом разработки.



# Программы для работы с Ffmpeg:

**FFmpeg** – очень быстрый видео и аудио конвертер с возможностью захвата видео и аудио потоков с различных устройств

**FFplay** – простой медиа проигрыватель

**FFprobe** – показывает информацию о мультимедиа потоках

**FFserver** – потоковый сервер для аудио и видео



# Простота использования FFmpeg:

Конвертация аудио.

```
ffmpeg -i birds_song.wav birds_song.mp3
```

Запись видео с вебкамеры.

Захватим видео с вебкамеры и сохраним в MPEG-файле:

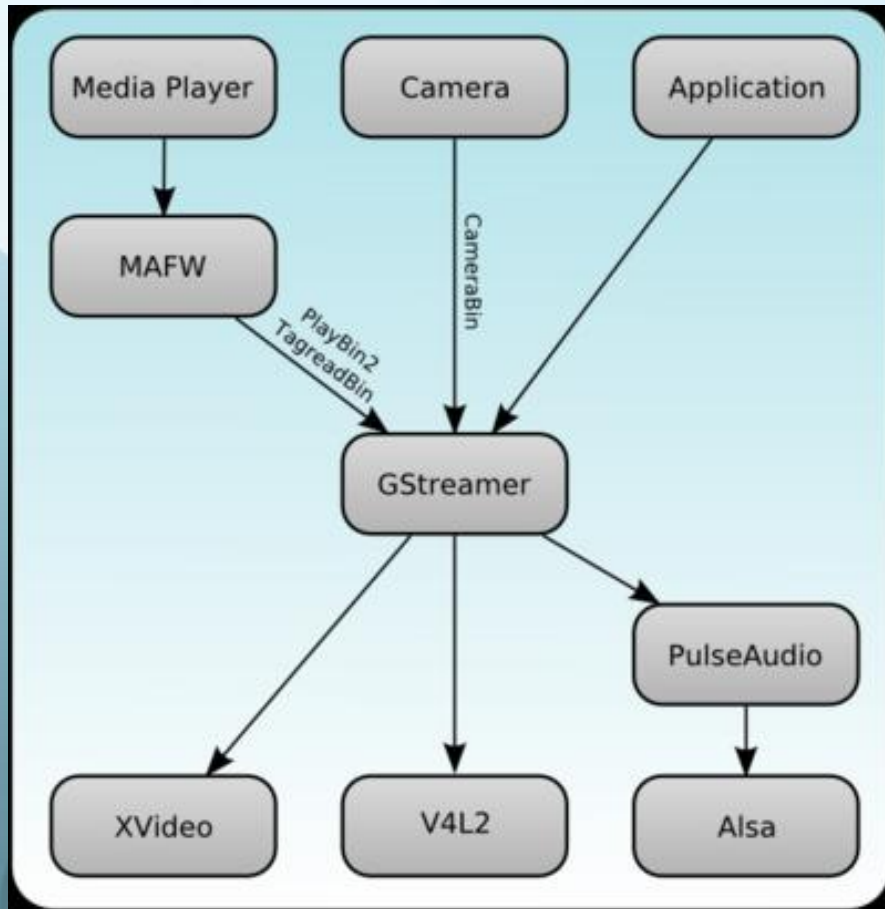
```
ffmpeg -f video4linux2 -s 320x240 -i /dev/video0 out.mpg
```

А теперь то же самое, но еще и со звуком:

```
ffmpeg -f oss -i /dev/dsp -f video4linux2 -s 320x240 -i /dev/video0 out.mpg
```



# Media Application FrameWork



Однако существует несколько задач, которые не затронуты в предыдущих фреймворках. Это связано со сложностью новых приложений, в которых пользователям предоставляются все виды возможных на данный момент мультимедийных служб, таких как: UPnP, Last.Fm, Youtube, и т. д. Для удобства написания приложений для этих сервисов в MeeGo существует The Multimedia Applications FrameWork(MAFW). MAFW предоставляет программистам простой и удобный способ создания современных мультимедиа приложений отвечающих сегодняшним запросам пользователей.

MAFW по сути есть надстройка более высокого уровня над Gstreamer. В описании к MAFW написано, что она позволяет более "просто" работать с мультимедиа сервисами с которыми "не умеет" работать GStreamer.



# Модуль Qt Phonon

**Phonon** — мультимедийный фреймворк для KDE4, который предоставляет API для разработки мультимедиа-приложений.

Входит в состав Qt начиная с версии 4.4.

multimedia  
PHONON



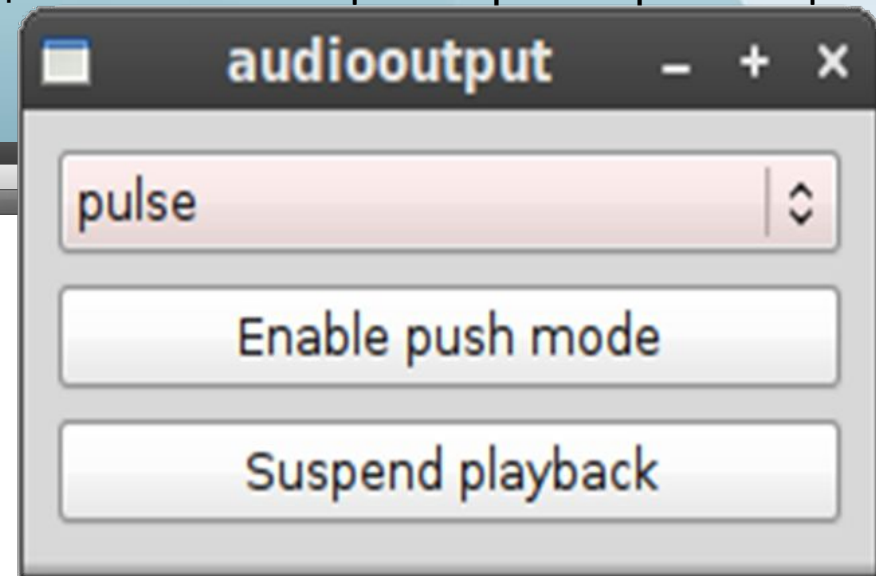
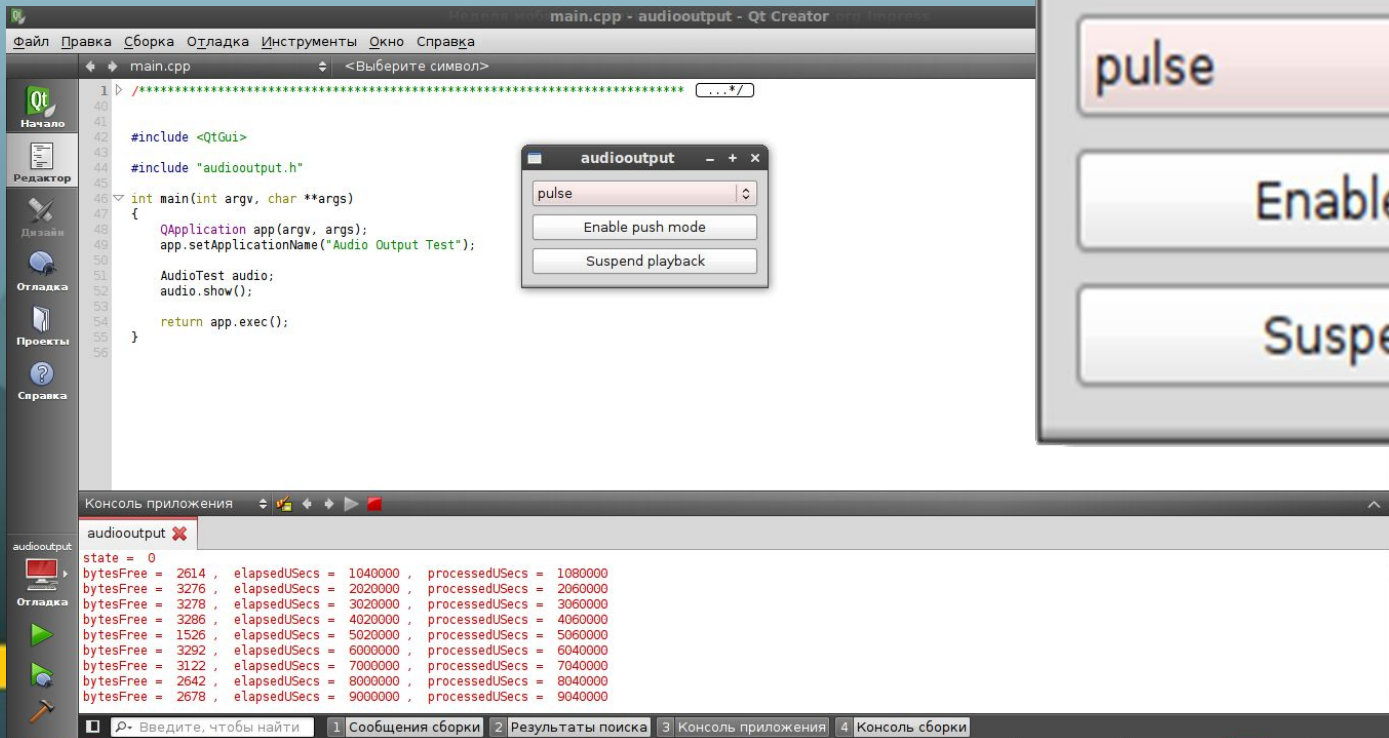
# Использование в своем приложении

Для включения определений классов этого модуля используйте следующую директиву:

```
#include <QtMultimedia>
```

Для линковки приложения с этим модулем, добавьте в ваш qmake файл проекта .pro:

```
QT += multimedia
```





# Интересные ссылки:

**Gstreamer** – <http://www.gstreamer.org>

**FFmpeg** – <http://www.ffmpeg.org>

**MAFW** – <http://www.garage.maemo.org/projects/mafw/>

**Phonon** – <http://phonon.kde.org/>

**Qt** – <http://www.qt.nokia.com>

**Описание классов Qt** – <http://doc.qt.nokia.com/>

**Описание на русском** – <http://doc.crossplatform.ru/>



У вас есть ко мне вопросы?

