Multimediální frameworky

David Bařina

12. dubna 2013

Obsah

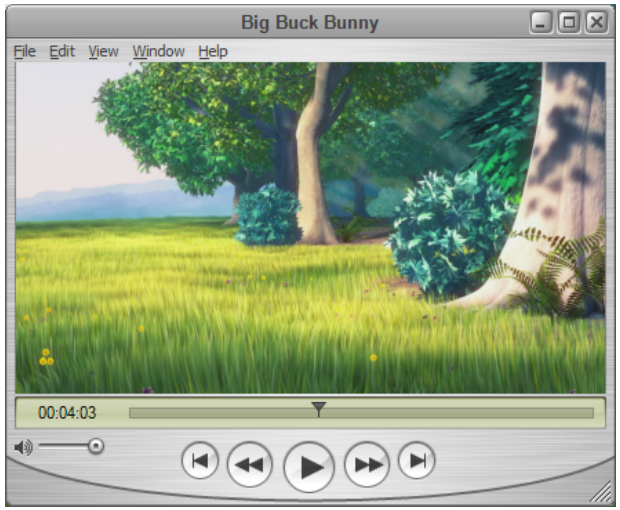
Multimédia



- multimédia:

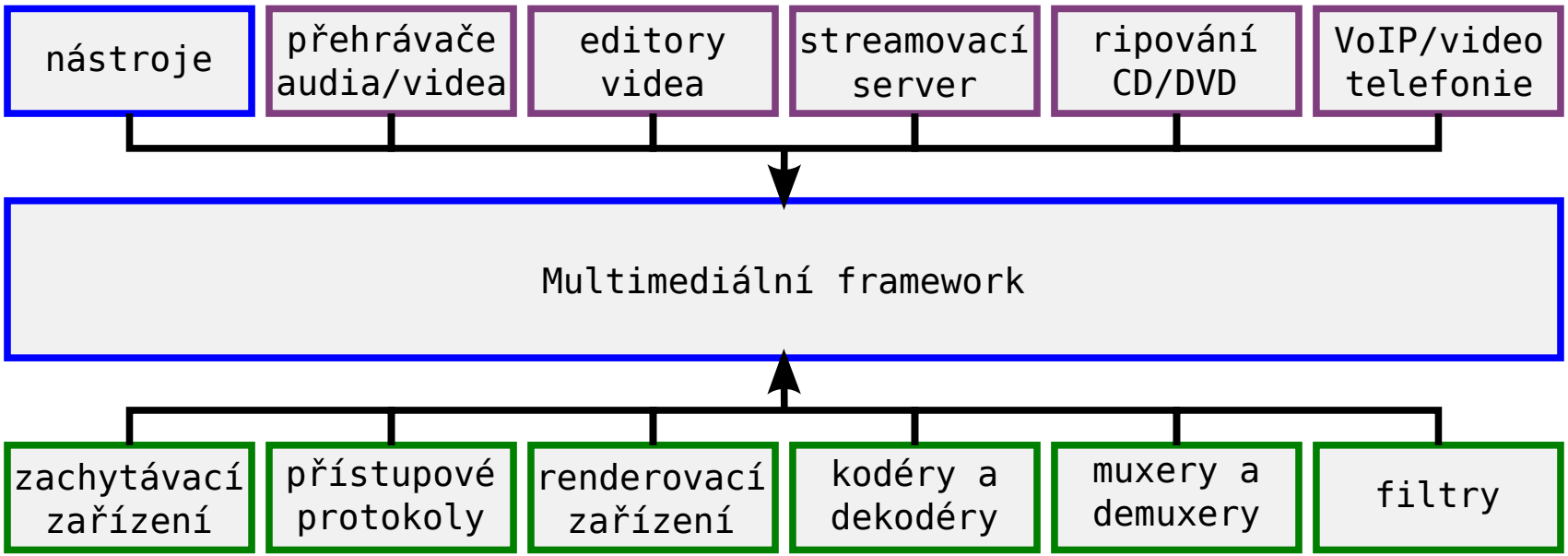
  text, zvuk, statický obraz, **video**, metainformace, ...
- potřeba:
  - získávat (kamera),
  - ukládat (pevný disk, komprese),
  - vyhledávat (podle popisu),
  - přehrávat,
  - upravovat (střih videa), ...
- ukládání: kontejner + kodeky

Multimediální framework

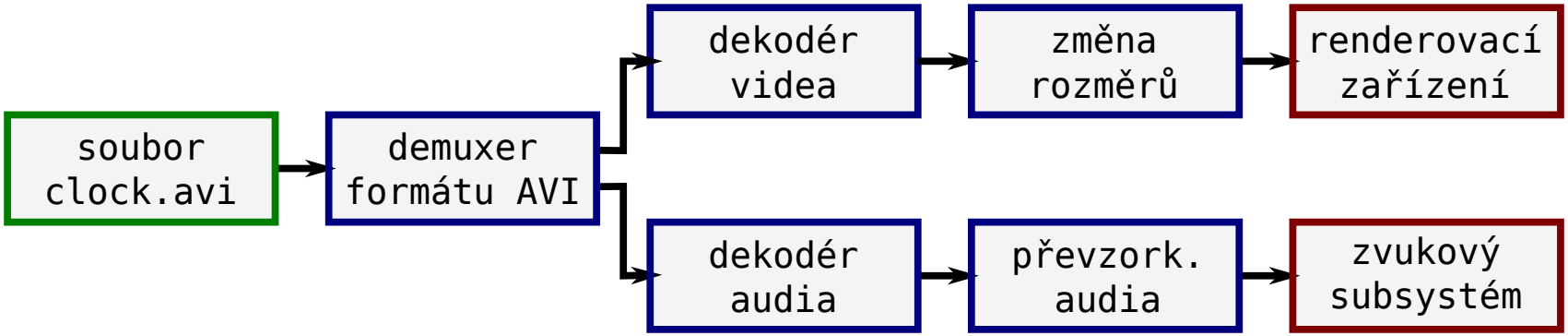


- vše zaobaluje
- knihovny (API), nástroje (přehrávač, CLI)
- formáty: kontejnery, kodeky, protokoly, ...
- požadavky: modularita, široká podpora formátů, intuitivní použití, dokumentace, výkon, platforma, ...
- problém: žádný neumí vše

Multimediální framework



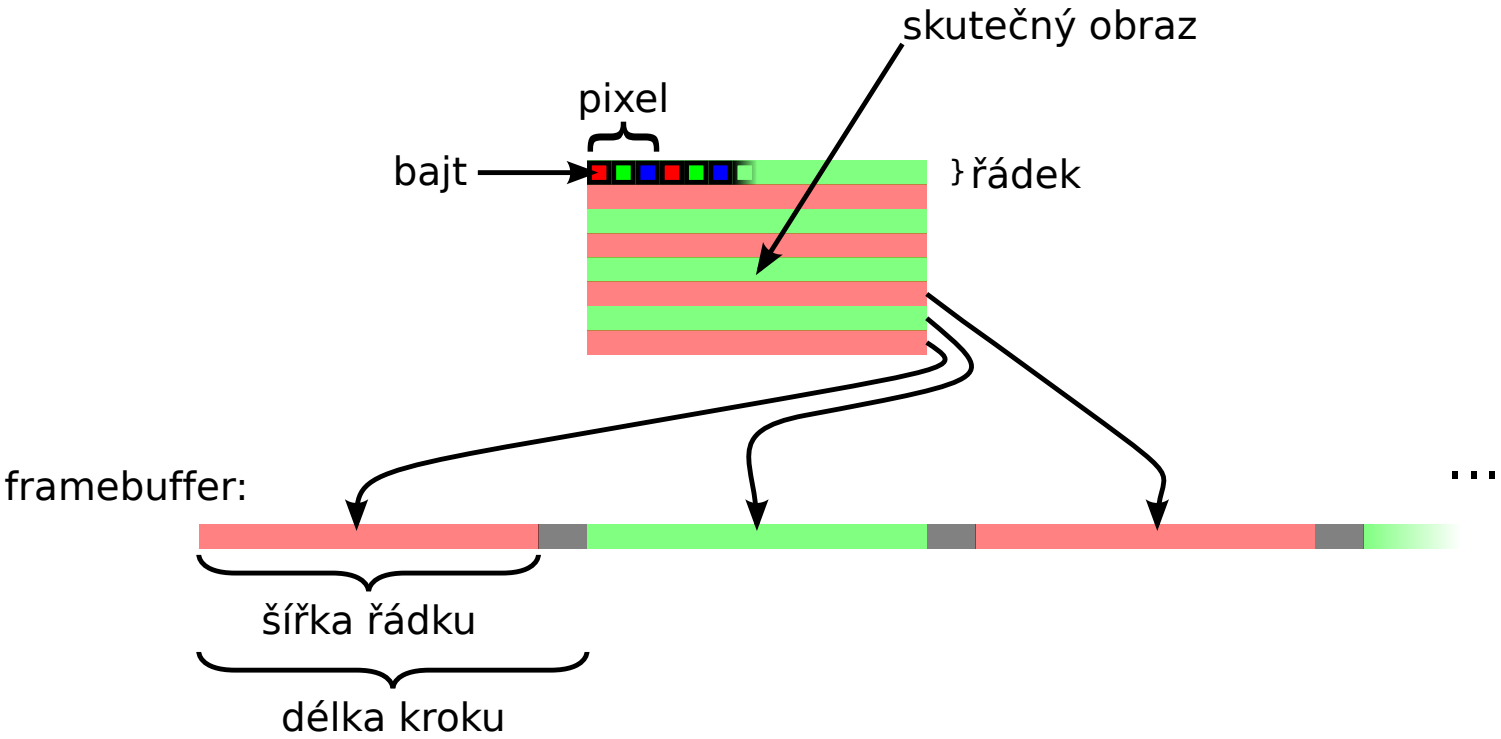
Graf filtrů



- modely pro přenos dat
  - push – zdroj neustále produkuje data, další filtr pasivně přijímá
  - pull – filtr aktivně požaduje data (parser od zdroje)
- data předávána v bufferech
- stavy: zastaven, pozastaven, spuštěn

Pojmy

- barevný model (RGB, $Y'C_bC_r$)
- formát pixelu (RGB24)
- framebuffer



Formát pixelu

- RGB24 (RGB888), BGR24
- podvzorkování
- planární formáty (odděleně)
  $R_0 R_1 R_2 \ldots G_0 G_1 G_2 \ldots B_0 B_1 B_2 \ldots$
  IYUV (4:2:0), I422 (4:2:2)
- prokládané formáty
  $R_0 G_0 B_0 R_1 G_1 B_1 \ldots$
  RGB24, YUY2 (4:2:2), UYVY (4:2:2)

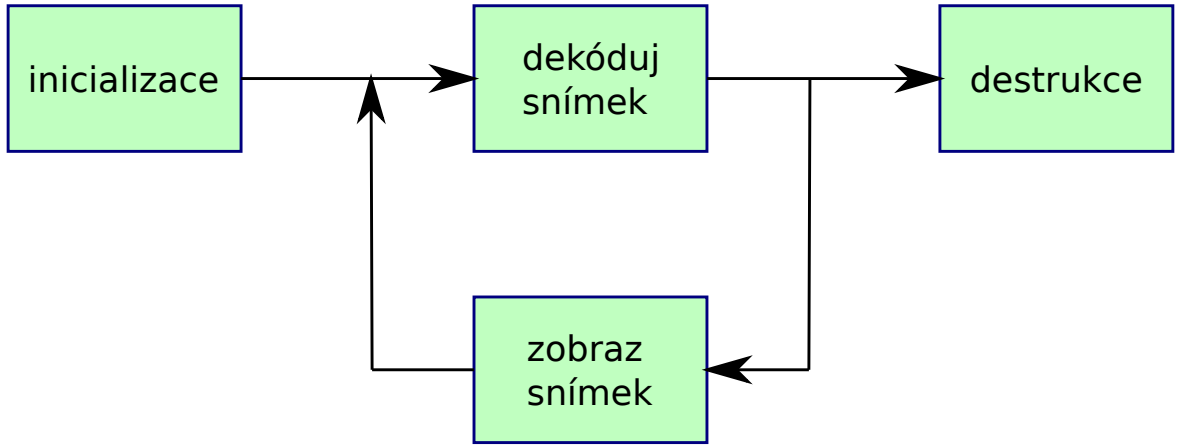
Multimediální framework

některé významné frameworky:

- Video for Windows (VirtualDub, Media Player)
- DirectShow (WMP, BSPlayer, Media Player Classic)
- FFmpeg (MPlayer, VLC, ffdshow)
- QuickTime (QuickTime)
- Media Foundation (Windows Media Player 11/12)
- GStreamer
- xine, libvlc, Phonon a další. . .

Přehrávač, kodek

přehrávač videa:



funkce kodeku:

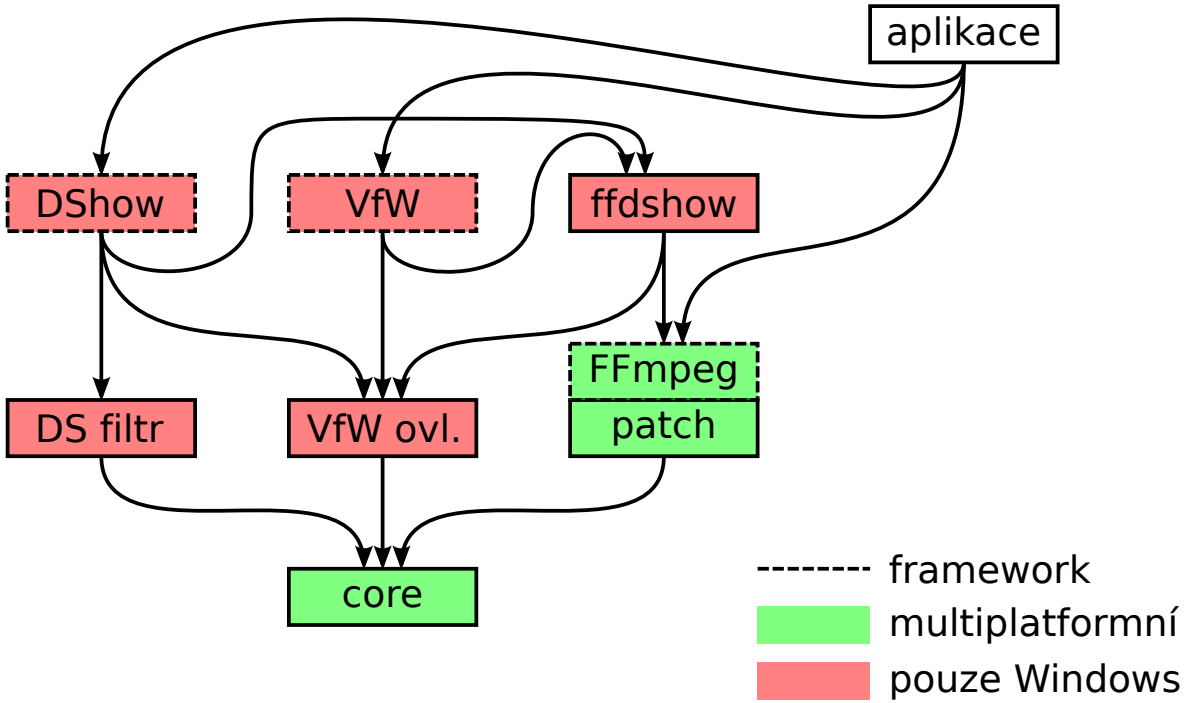
- inicializace (alokace paměti – rozdílové snímky, parametry)
- odhad maximální velikosti zkomprimovaného snímku
- komprese snímku
- dekomprese snímku

Kodek

- knihovna vs. plugin do frameworku
- kontext (veřejná a privátní část)
- funkce
  - `compress`, `decompress`
  - `get_size`
  - `query`
  - `create`, `destroy`

Kodek – příklad

jádro + ovladač pro VfW + filtr pro DShow + patch na FFmpeg



Video for Windows

- Video for Windows (VfW) / Video Compression Manager (VCM)
- vyvinul Microsoft jako reakci na QuickTime (Apple)
- první verze (verze 1.0), listopad 1992
- vlastní souborový formát Audio Video Interleave (AVI)
- nástupcem se stal DirectShow
- dokumentace na MSDN

Otevření souboru

```
LONG hr;
PAVIFILE pfile;

AVIFileInit();

hr = AVIFileOpen(&pfile, szFile, OF_SHARE_DENY_WRITE, 0L);
if (hr != 0) {
        return;
}

AVIFileRelease(pfile);
AVIFileExit();
```

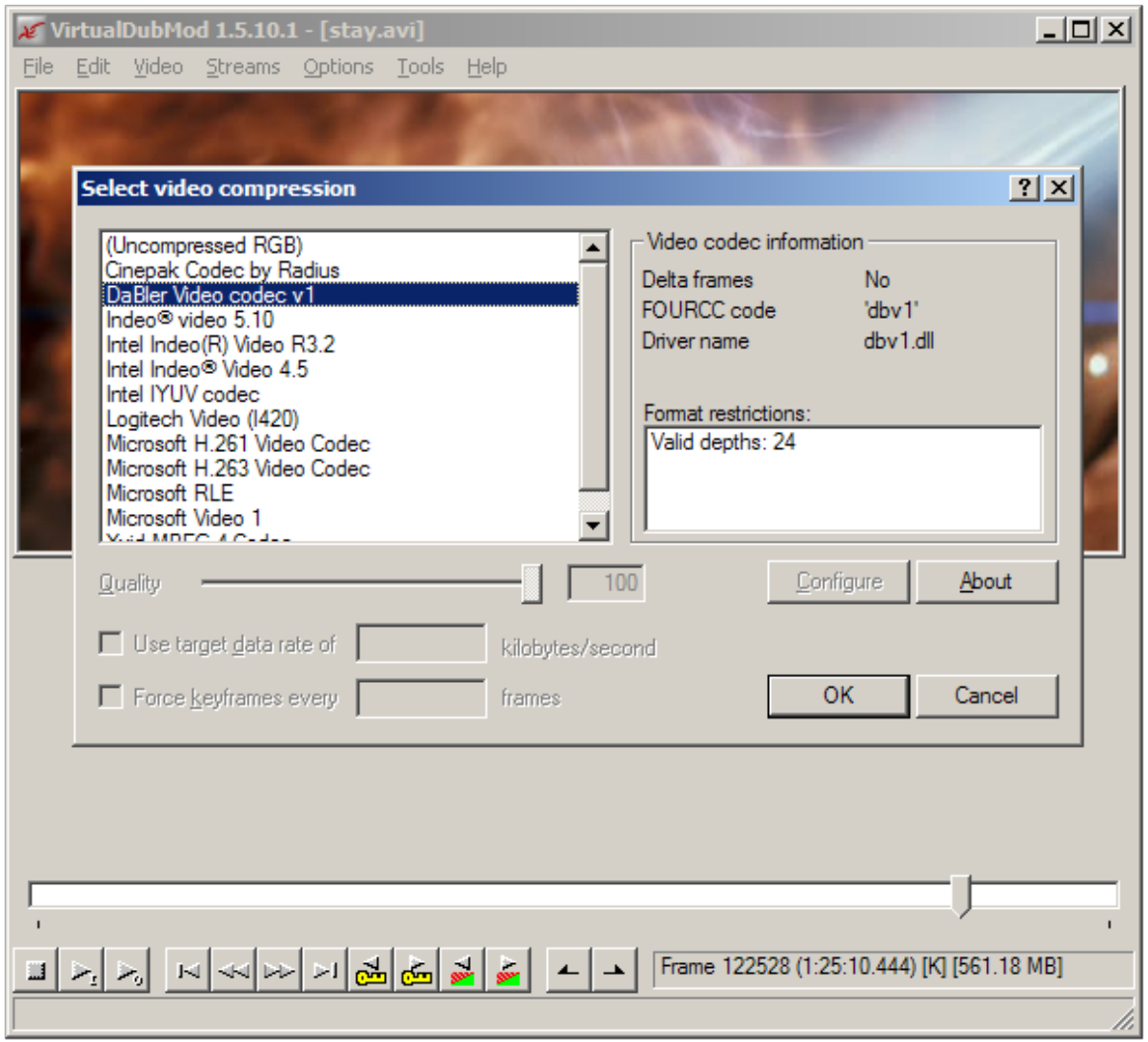
Video for Windows

Kostra kodeku

```
#include <vfw.h>

LRESULT WINAPI DriverProc(
        DWORD dwDriverId,
        HDRVR hdrvr,
        UINT msg,
        LONG lParam1,
        LONG lParam2)
{
        switch(msg)
        {
                case ICM_COMPRESS:
                        // komprimuj snimek
                        return Compress((ICCOMPRESS*)lParam1, (DWORD)lParam2);

                case ICM_DECOMPRESS:
                        // dekomprimuj snimek
                        return Decompress((ICDECOMPRESS*)lParam1, (DWORD)lParam2);
        }
}
```

- kodek: zkompilovat jen plugin

Video for Windows



Video for Windows

- `AVIFileInit` inicializuje knihovnu
- `AVIFileExit` ukončí práci s knihovnou
- `AVIFileOpen` otevře soubor AVI
- `AVIFileRelease` zavře soubor
- `AVIFileGetStream` vrátí vybranou stopu
- `AVIFileCreateStream` vytvoří novou stopu
- `AVIStreamInfo` vrátí informace o stopě
- `AVIStreamReadFormat` vrátí informace o formátu stopy
- `AVIStreamGetFrameOpen` připraví dekompresor
- `AVIStreamGetFrame` dekomprimuje snímek
- `AVIStreamGetFrameClose` ukončí dekompresi
- `AVIStreamOpenFromFile` otevře vybranou stopu
- `AVIStreamSetFormat` nastaví formát stopy
- `AVIStreamRead` přečte komprimovaná data
- `AVIStreamWrite` zapíše data do stopy
- `AVIStreamRelease` uzavře stopu

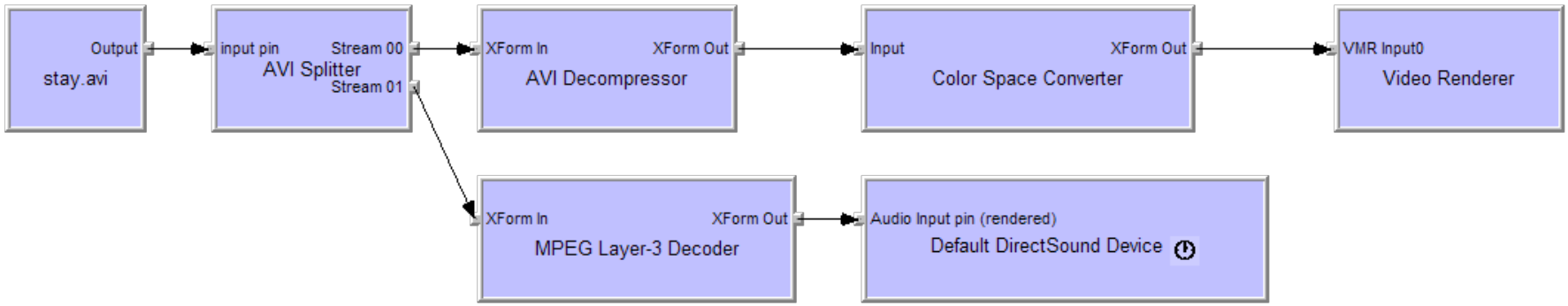
Video for Windows

- `ICM_ABOUT` zobrazit dialog s informacemi
- `ICM_COMPRESS` zkomprimovat snímek
- `ICM_COMPRESS_BEGIN` připravit se na kompresi podle parametrů
- `ICM_COMPRESS_END` konec komprese
- `ICM_COMPRESS_GET_FORMAT` informace o komprimovaném formátu
- `ICM_COMPRESS_GET_SIZE` max. velikost komprimovaného snímku
- `ICM_COMPRESS_QUERY` podpora dekomprimovaného formátu
- `ICM_CONFIGURE` konfigurační dialog
- `ICM_DECOMPRESS` dekomprimuje snímek
- `ICM_DECOMPRESS_BEGIN` připravit se na dekompresi
- `ICM_DECOMPRESS_END` konec dekomprese
- `ICM_DECOMPRESS_GET_FORMAT` info. o dekomprimovaném formátu
- `ICM_DECOMPRESS_QUERY` podpora komprimovaného formátu
- `ICM_GETINFO` informace o kodeku

DirectShow

- DirectShow (DShow, DS)
- nástupce VfW; nástupcem Media Foundation
- založen na objektovém modelu COM (Component Object Model)
- graf složený z filtrů
- automatická konverze barevných modelů (proti VfW)
- filtry: zdrojové, transformační, renderovací
- pro vývoj nutno nainstalovat Windows SDK, dříve DirectX SDK
- utilita GraphEdit
- zpětná kompatibilita: VfW kodeky obaleny filtrem AVI Decompressor
- formáty identifikovány GUID (FourCC obalen)
- dokumentace na MSDN

DirectShow


Output
stay.avi
input pin
Stream 00
AVI Splitter
Stream 01
XForm In
XForm Out
AVI Decompressor
Input
XForm Out
Color Space Converter
VMR Input0
Video Renderer
XForm In
XForm Out
MPEG Layer-3 Decoder
Audio Input pin (rendered)
Default DirectSound Device

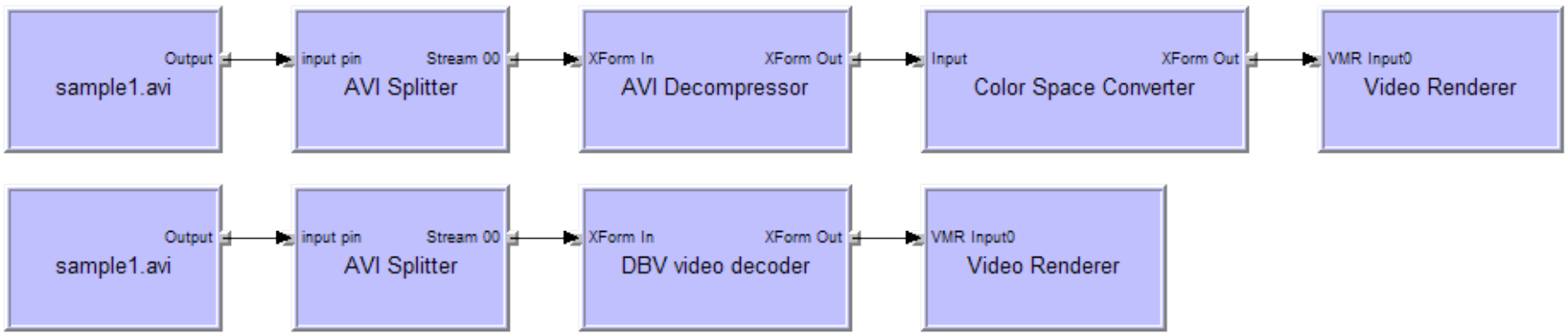
DirectShow

Dekompresor videa

```
class CDBVDecoder: public CVideoTransformFilter, public IDBVDecoder
{
public:
        static CUnknown *WINAPI CreateInstance(LPUNKNOWN punk, HRESULT *phr);
        STDMETHODIMP NonDelegatingQueryInterface(REFIID riid, void **ppv);
        DECLARE_IUNKNOWN;

        CDBVDecoder(LPUNKNOWN punk, HRESULT *phr);

        HRESULT CheckInputType(const CMediaType *mtIn);
        HRESULT GetMediaType(int iPos, CMediaType *pmt);
        HRESULT SetMediaType(PIN_DIRECTION direction, const CMediaType *pmt);
        HRESULT CheckTransform(const CMediaType *mtIn, const CMediaType *mtOut);
        HRESULT DecideBufferSize(IMemAllocator *pima,
                ALLOCATOR_PROPERTIES *pProperties);

        HRESULT Transform(IMediaSample *pIn, IMediaSample *pOut);
};
```

- kodek: zkompilovat jen plugin

DirectShow


Output
sample1.avi
input pin
Stream 00
AVI Splitter
XForm In
XForm Out
AVI Decompressor
Input
XForm Out
Color Space Converter
VMR Input0
Video Renderer
Output
sample1.avi
input pin
Stream 00
AVI Splitter
XForm In
XForm Out
DBV video decoder
VMR Input0
Video Renderer

FFmpeg

- svobodný multiplatformní software
- využívají jej MPlayer, VLC media player, Avidemux, ffdshow
- knihovny:
  - libavutil (matematické rutiny, pro zjednodušení programování)
  - libavcodec (audio a video kodeky)
  - libavformat (muxery a demuxery/splittery pro kontejnery)
  - libavdevice (grabování a renderování přes V4L(2), VfW, ALSA)
  - libavfilter (filtry)
  - libswscale (změna rozlišení a barevného modelu obrazu)
- podporované formáty na `http://www.ffmpeg.org/general.html`
- Libav (fork FFmpegu), `http://libav.org/`

FFmpeg

ffmpeg	překódování multimediálních souborů
ffserver	streamovací server
ffplay	jednoduchý přehrávač založený na SDL
ffprobe	zobrazí informace o multimediálních souborech

Příklady

```
ffmpeg -formats
ffmpeg -codecs
ffprobe clock.avi
ffplay clock.avi
ffplay -f video4linux2 /dev/video0
ffmpeg -i clock.avi -c:v ffv1 output.avi
```

FFmpeg – graf filtrů

1. jediný filtr

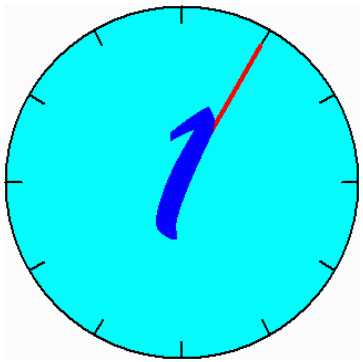
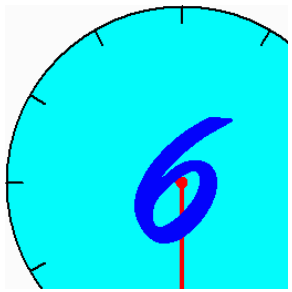
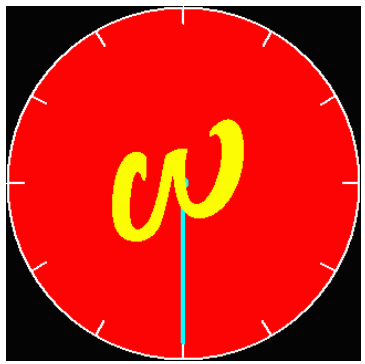
```
ffplay -vf vflip clock.avi
```

2. parametry

```
ffplay -vf crop=256:256:0:0 clock.avi
```

3. řetězec filtrů

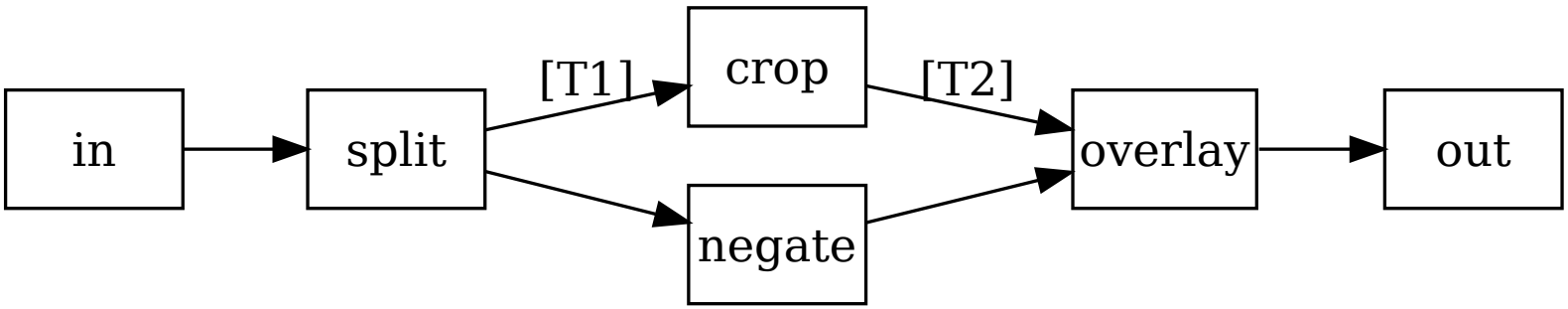
```
ffplay -vf "transpose, negate" clock.avi
```

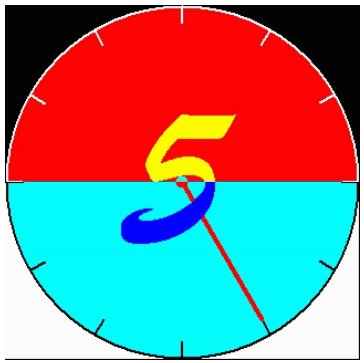




FFmpeg – graf filtrů

1. pojmenované pady, větvení

```
ffplay -vf "[in] split [T1], negate, [T2] overlay=0:H/2
[out]; [T1] crop=iw:ih/2:0:ih/2 [T2]" clock.avi
```





FFmpeg

Otevření video stopy

```
#include <avcodec.h>
#include <avformat.h>

int main(int argc, charg *argv[])
{
        av_register_all();

        AVFormatContext *pFormatCtx;

        if(av_open_input_file(&pFormatCtx, argv[1], NULL, 0, NULL) != 0)
                return -1;

        if(av_find_stream_info(pFormatCtx) < 0)
                return -1;

        AVCodecContext *pCodecCtx;

        if(pFormatCtx->streams[0]->codec.codec_type != CODEC_TYPE_VIDEO)
                return -1;

        pCodecCtx = &pFormatCtx->streams[0]->codec;
```

FFmpeg

Smyčka přehrávače

```
AVPacket pkt;

while( av_read_frame(pFormatCtx, &pkt) == 0 )
{
        if( pkt.stream_index == videoStream)
        {
                int frameFinished = 0;
                if( avcodec_decode_video2(pCodecCtx, pFrame, &frameFinished, &pkt) < 0 )
                        abort();
                if(frameFinished)
                {
                        // sws_scale

                        // avcodec_encode_video2

                        // ...
                }
        }
        av_free_packet(&pkt);
}
```

FFmpeg

Kostra kodeku

```
static int dbv1_decode_frame(AVCodecContext *avctx,
                void *outdata, int *outdata_size,
                const uint8_t *buf, int buf_size)
{
        // dekoduj snimek
}

AVCodec dbv1_decoder =
{
        .name           = "dbv1",
        .type           = CODEC_TYPE_VIDEO,
        .id             = CODEC_ID_DBV1,
        .priv_data_size = sizeof(DBV1Context),
        .init           = dbv1_decode_init,
        .close          = dbv1_decode_close,
        .decode         = dbv1_decode_frame,
        .long_name      = NULL_IF_CONFIG_SMALL("DaBler's␣Video␣codec␣v1"),
        .capabilities   = CODEC_CAP_DR1,
};
```

- kodek: zkompilovat modul + libavcodec + libavformat

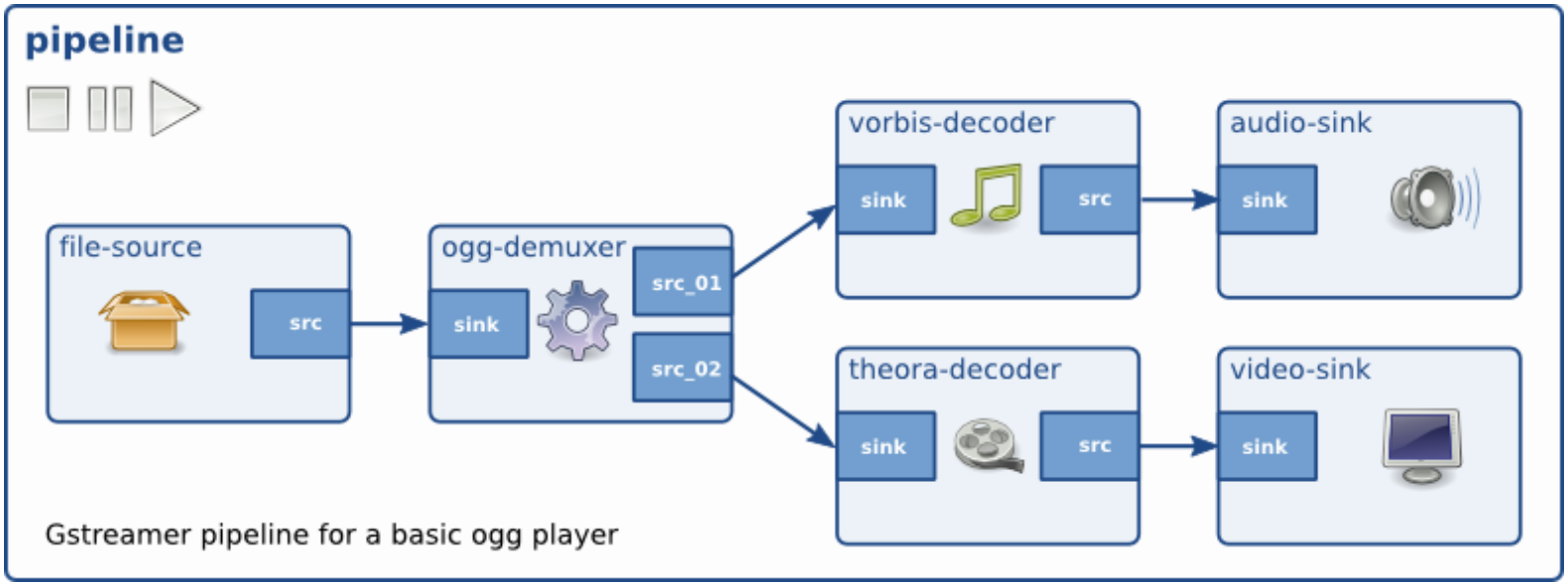
FFmpeg

- `av_register_all` zaregistruje kodeky, muxery, demuxery, protokoly
- `avformat_open_input` otevře vstupní kontejner, přečte hlavičku
- `avformat_find_stream_info` načte z kontejneru informace
- `av_dump_format` zobrazí informace o kontejneru a stopách
- `avcodec_find_decoder` podle ID kodeku najde dekodér
- `avcodec_find_encoder` podle ID kodeku vrátí kodér
- `avcodec_alloc_frame` alokuje snímek
- `av_read_frame` přečte z kontejneru jeden paket (snímek)
- `avformat_write_header` zapíše do kontejneru hlavičku stopy
- `av_write_frame` zapíše do kontejneru paket
- `av_write_trailer` zapíše do kontejneru patičku stopy
- `avcodec_decode_video2` z paketu dekóduje jeden snímek videa
- `avcodec_encode_video` zkomprimuje snímek videa do bufferu
- `av_find_best_stream` vrátí z kontejneru požadovanou stopu
- `avformat_new_stream` přidá do kontejneru novou stopu

GStreamer

- svobodný multiplatformní, 1999
- založen na GLib, primárně pro GNOME
- založen na grafu filtrů (pipeline), jako DirectShow
- nástroje: `gst-launch`, `gst-inspect`, `gst-editor`
- terminologie
  - pads = spoje mezi filtry
  - source pad se propojí do sink pad
  - typ dat se zjistí pomocí capabilities
  - element, bin, pipeline
- tři balíčky pluginů: The Good, the Bad and the Ugly

GStreamer



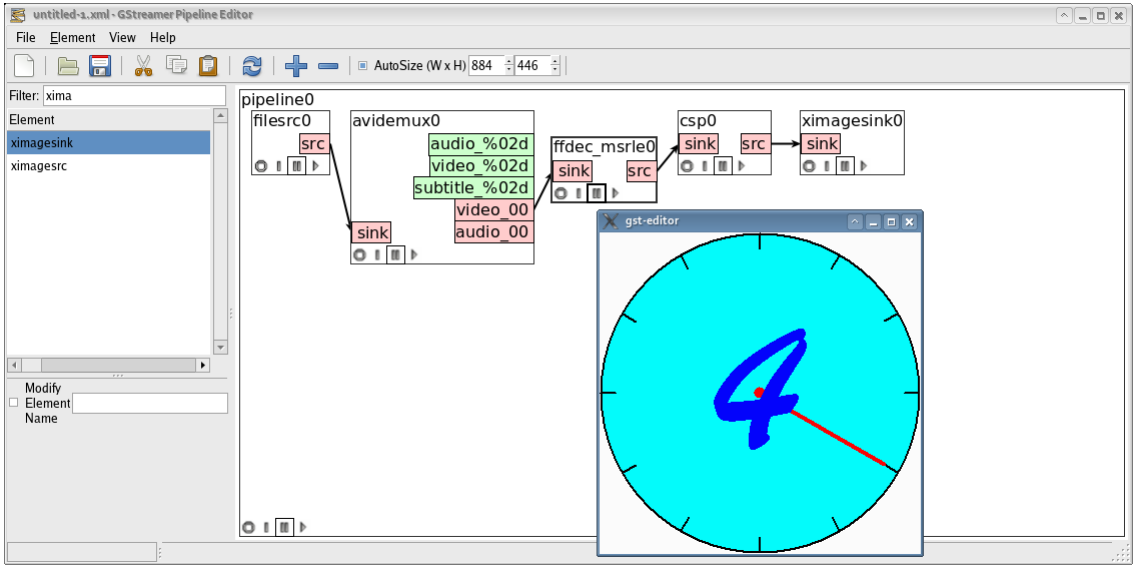
Sestavení pipeline

```
export GST_PLUGIN_PATH=./.libs

gst-launch-0.10 v4l2src device="/dev/video0" ! videoscale ! video/x-raw-yuv,
width=160 ! ffmpegcolorspace ! video/x-raw-gray ! abr2 ! ffmpegcolorspace !
videoscale ! video/x-raw-rgb, width=640 ! ximagesink
```

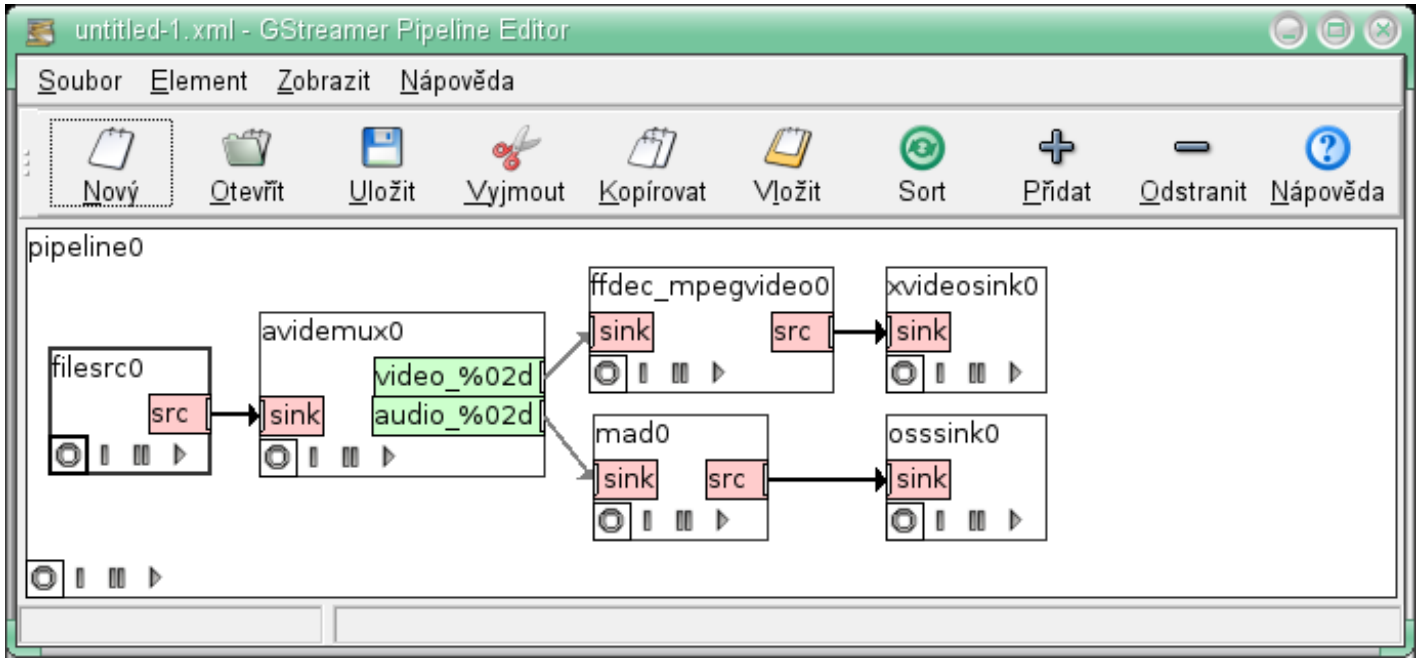
GStreamer

Přehrávač

```
gst-launch-0.10 playbin uri=file:///tmp/clock.avi

gst-launch-0.10 filesrc location=/tmp/clock.avi ! decodebin !
        colorspace ! ximagesink

gst-launch-0.10 filesrc location=/tmp/clock-rle.avi ! avidemux !
        ffdec_msrle ! colorspace ! ximagesink
```



GStreamer – GUI, XML



Uložení/načtení pipeline

```
gst_xml_write_file (GST_ELEMENT (pipeline), fopen ("xmlTest.gst", "w"));

xml = gst_xml_new ();
ret = gst_xml_parse_file(xml, "xmlTest.gst", NULL);
g_assert (ret == TRUE);
pipeline = gst_xml_get_element (xml, "pipeline");
g_assert (pipeline != NULL);
gst_element_set_state (pipeline, GST_STATE_PLAYING);
```

GStreamer

omezení vstupu a výstupu

Capabilities

```
gst-inspect vorbisdec

Pad Templates:
  SRC template: 'src'
    Availability: Always
    Capabilities:
      audio/x-raw-float
                  rate: [ 8000, 50000 ]
              channels: [ 1, 2 ]
            endianness: 1234
                 width: 32
          buffer-frames: 0

  SINK template: 'sink'
    Availability: Always
    Capabilities:
      audio/x-vorbis
```

GStreamer

Plugin

```
$ git clone git://anongit.freedesktop.org/gstreamer/gst-template.git

$ ../tools/make_element abr2

static gboolean abr2_init (GstPlugin * abr2) {
        // ...
}
static GstFlowReturn gst_abr2_chain (GstPad * pad, GstBuffer * buf) {
        // ...
        GstStructure *structure = gst_caps_get_structure (pad->caps, 0);
        gst_structure_get_int (structure, "width", &width);
        gst_structure_get_int (structure, "height", &height);
        // ...
        img.imageData = (char*) GST_BUFFER_DATA(buf);
        // ...
}

$ ./autogen.sh

$ make

$ export GST_PLUGIN_PATH=./.libs

$ gst-launch-0.10 v4l2src device="/dev/video0" ! videoscale ! video/x-raw-yuv,
width=160 ! ffmpegcolorspace ! video/x-raw-gray ! abr2 ! ffmpegcolorspace !
videoscale ! video/x-raw-rgb, width=640 ! ximagesink
```

- kodek: zkompilovat jen plugin

Shrnutí

- multimediální framework (graf filtrů, framebuffer, formát pixelu)
- stavba přehrávače (diagram), kodeku (funkce)
- Video for Windows
- DirectShow
- FFmpeg
- GStreamer