Multimedia frameworks

David Bařina

April 17, 2013

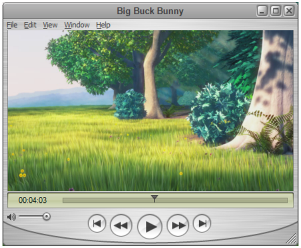
Contents

Multimedia

- multimedia:

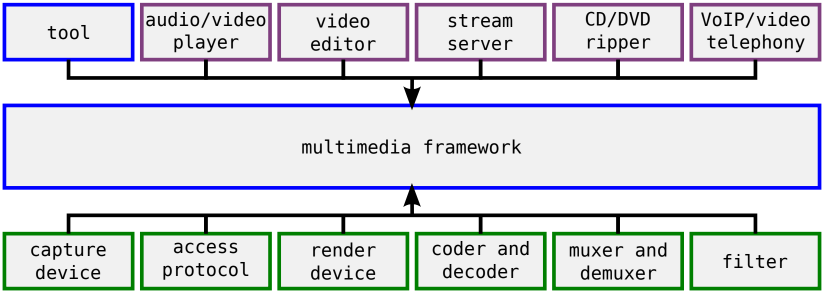
  text, audio, still image, **video**, metadata, . . .
- needs:
  - acquire (camera),
  - store (hard drive, compression),
  - search (by description),
  - play,
  - edit (video editing), . . .
- store: container + codecs

Multimedia frameworks

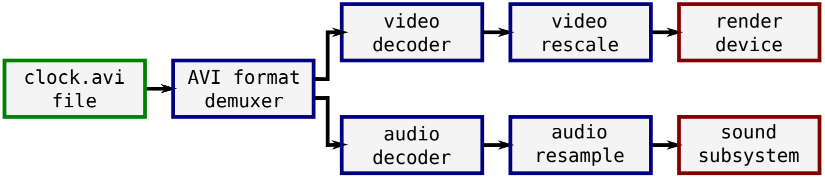


- encapsulates multimedia processing
- libraries (API), tools (player, CLI)
- formats: containers, codecs, protocols, . . .
- requirements: modularity, format support, intuitive use, documentation, performance, platform, . . .
- issue: noone supports every feature

Multimedia frameworks



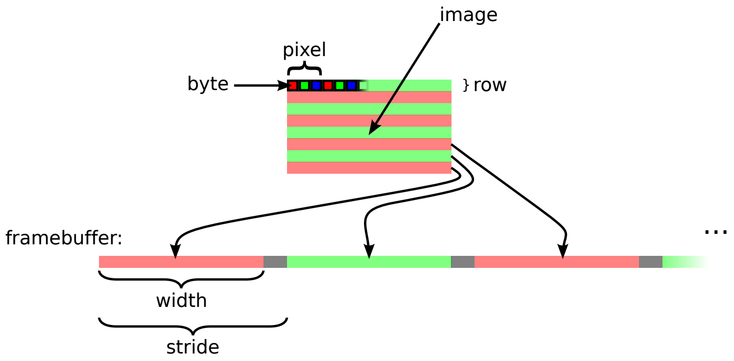
Filter graph



- data transfer models
  - push – source continuously produces data, next filter passively receives
  - pull – filter actively requests data (parser from source)
- data passed in buffers
- states: stopped, paused, running

Notions

- color model (RGB, $Y'C_bC_r$)
- pixel format (RGB24)
- framebuffer



Pixel format

- RGB24 (RGB888), BGR24
- chroma subsampling
- planar formats (separated)
  $R_0 R_1 R_2 \ldots G_0 G_1 G_2 \ldots B_0 B_1 B_2 \ldots$
  IYUV (4:2:0), I422 (4:2:2)
- packed formats
  $R_0 G_0 B_0 R_1 G_1 B_1 \ldots$
  RGB24, YUY2 (4:2:2), UYVY (4:2:2)

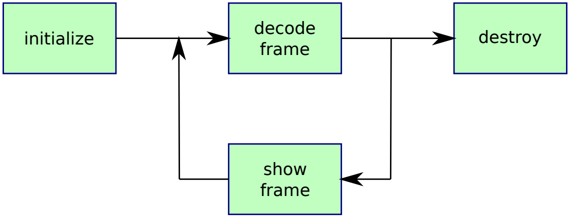
Multimedia frameworks

important frameworks:

- 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, . . .

Player, codec

video player:



codec functions:

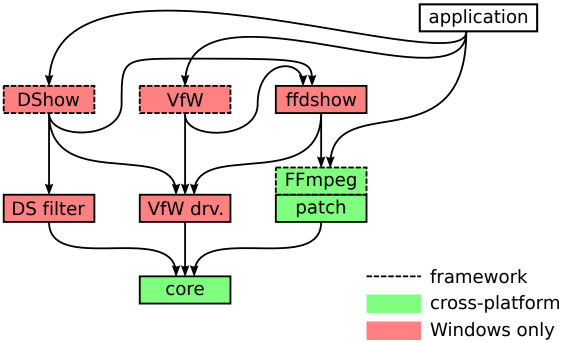
- initialization (memory allocation – delta frames, parameters)
- estimation of compressed image size
- frame compression
- frame decompression

Codec

- library vs. framework plugin
- context (public and private part)
- functions
  - `compress`, `decompress`
  - `get_size`
  - `query`
  - `create`, `destroy`

Codec – example

core + VfW driver + DShow filter + FFmpeg patch



Video for Windows

- Video for Windows (VfW) / Video Compression Manager (VCM)
- developed by Microsoft as a reaction to QuickTime (Apple)
- first version (ver. 1.0), November 1992
- own file format Audio Video Interleave (AVI)
- successor was DirectShow
- documentation on MSDN

Opening AVI file

```
LONG hr;
PAVIFILE pfile;

AVIFileInit();

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

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

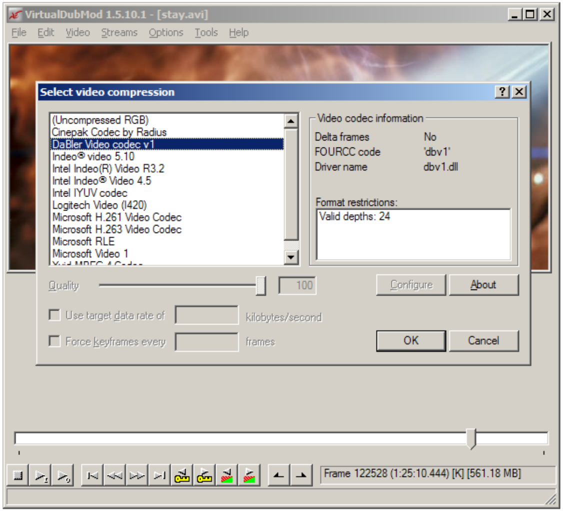
Video for Windows

Codec skeleton

```
#include <vfw.h>

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

                case ICM_DECOMPRESS:
                        // decompress a frame
                        return Decompress((ICDECOMPRESS*)lParam1, (DWORD)lParam2);
        }
}
```

- codec: compile just the plugin

Video for Windows


VirtualDubMod 1.5.10.1 - [stay.avi]
File Edit Video Streams Options Tools Help
Select video compression
Configure
About
OK
Cancel

(Uncompressed RGB)
Cinepak Codec by Radius
DaBler Video codec v1
Indeo® video 5.10
Intel Indeo(R) Video R3.2
Intel Indeo® Video 4.5
Intel IYUV codec
Logitech Video (I420)
Microsoft H.261 Video Codec
Microsoft H.263 Video Codec
Microsoft RLE
Microsoft Video 1
Xvid MPEG-4 Codec

Video codec information

Delta frames	No
FOURCC code	'dbv1'
Driver name	dbv1.dll

Format restrictions:
Valid depths: 24

Quality 100

☐ Use target data rate of kilobytes/second

☐ Force keyframes every frames

Frame 122528 (1:25:10.444) [K] [561.18 MB]

Video for Windows

- `AVIFileInit` initialize the library
- `AVIFileExit` finish using the library
- `AVIFileOpen` open AVI file
- `AVIFileRelease` close the file
- `AVIFileGetStream` get selected stream
- `AVIFileCreateStream` create new stream
- `AVIStreamInfo` vrátí stream information
- `AVIStreamReadFormat` return stream format
- `AVIStreamGetFrameOpen` prepare a decompressor
- `AVIStreamGetFrame` decomrpess a frame
- `AVIStreamGetFrameClose` finish decompression
- `AVIStreamOpenFromFile` open selecter stream
- `AVIStreamSetFormat` set stream format
- `AVIStreamRead` read compressed data
- `AVIStreamWrite` write data into the stream
- `AVIStreamRelease` close the stream

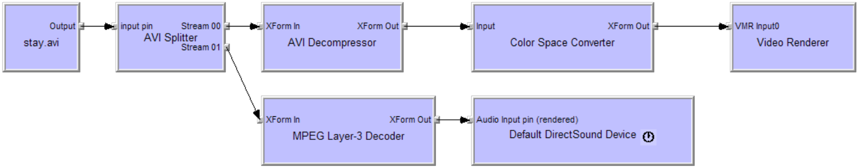
Video for Windows

- `ICM_ABOUT` show dialog with information
- `ICM_COMPRESS` compress a frame
- `ICM_COMPRESS_BEGIN` prepare for compression (parameters)
- `ICM_COMPRESS_END` end of compression
- `ICM_COMPRESS_GET_FORMAT` compressed format information
- `ICM_COMPRESS_GET_SIZE` maximum size of a compressed frame
- `ICM_COMPRESS_QUERY` query to support decompressed format
- `ICM_CONFIGURE` configuration dialog
- `ICM_DECOMPRESS` decompress a frame
- `ICM_DECOMPRESS_BEGIN` prepare for decompression
- `ICM_DECOMPRESS_END` end of decompression
- `ICM_DECOMPRESS_GET_FORMAT` decompressed format information
- `ICM_DECOMPRESS_QUERY` query to support compressed format
- `ICM_GETINFO` return codec information

DirectShow

- DirectShow (DShow, DS)
- predecessor was VfW; successor is Media Foundation
- based on the object model COM (Component Object Model)
- graph composed of filters
- automatic conversion of color models (unlike VfW)
- filters: source, transform, render
- development: Windows SDK (previously DirectX SDK) installed
- GraphEdit utility
- backward compatibility:
  VfW codecs wrapped in AVI Decompressor filter
- formats identified by GUID (FourCC enveloped)
- documentation on MSDN

DirectShow


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

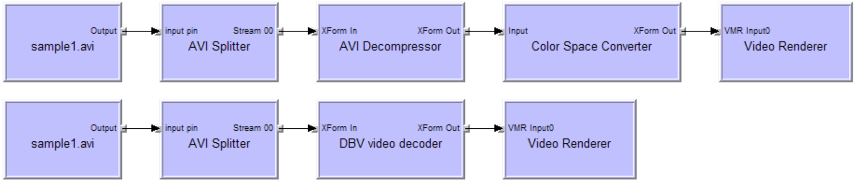
DirectShow

Videa decompressor

```
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);
};
```

- codec: compile just the 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

- free cross-platform software
- used by MPlayer, VLC media player, Avidemux, ffdshow
- libraries:
  - libavutil (math routines, to simplify programming)
  - libavcodec (audio and video codecs)
  - libavformat (muxers and demuxers/splitters for containers)
  - libavdevice (connection with V4L(2), VfW, ALSA)
  - libavfilter (filters)
  - libswscale (rescaling and color space conversion)
- supported formats on `http://www.ffmpeg.org/general.html`
- Libav (FFmpeg fork), `http://libav.org/`

FFmpeg

ffmpeg	recoding of multimedia files
ffserver	stream server
ffplay	simple player based on SDL
ffprobe	prints information from multimedia files

Commands

```
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 – filter graph

1. single filter

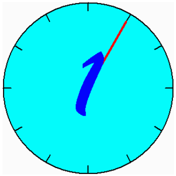
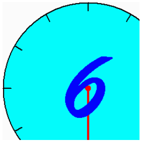
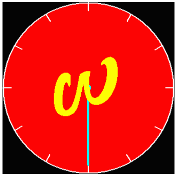
```
ffplay -vf vflip clock.avi
```

2. parameters

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

3. filter chain

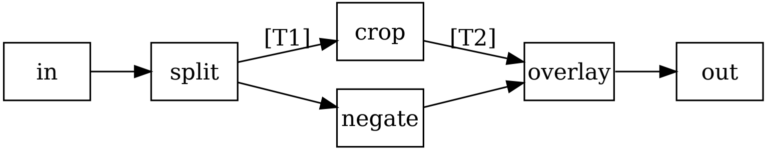
```
ffplay -vf "transpose, negate" clock.avi
```

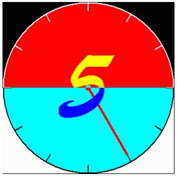




FFmpeg – filter graph

1. named pads, branches

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





FFmpeg

Opening of video stream

```
#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

Player loop

```
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

Codec skeleton

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

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,
};
```

- codec: compile a module + libavcodec + libavformat

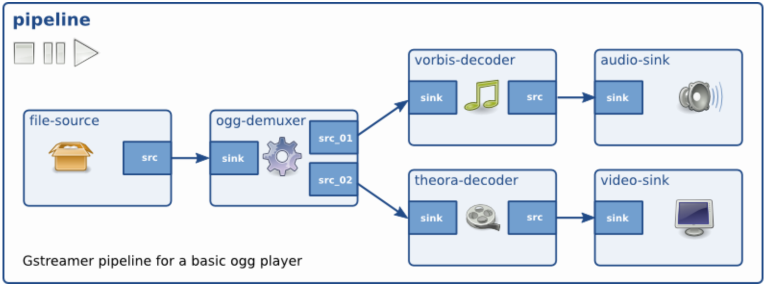
FFmpeg

- `av_register_all` register codecs, muxers, demuxers, protocols
- `avformat_open_input` open input container, read a header
- `avformat_find_stream_info` read information from container
- `av_dump_format` print infromation about a container and streams
- `avcodec_find_decoder` find a decoder according to codec ID
- `avcodec_find_encoder` find a encoder according to codec ID
- `avcodec_alloc_frame` allocate a frame
- `av_read_frame` read one packet (frame) from a container
- `avformat_write_header` write stream header into a container
- `av_write_frame` write packet into a container
- `av_write_trailer` write stream footer into a container
- `avcodec_decode_video2` decode one video frame from a packet
- `avcodec_encode_video` compress video frame into a buffer
- `av_find_best_stream` get selected stream in a container
- `avformat_new_stream` add new stream into a container

GStreamer

- free cross-plarform software, 1999
- based on GLib, primarily for GNOME
- based on filter graph (pipeline), like DirectShow
- tools: `gst-launch`, `gst-inspect`, `gst-editor`
- terminology
  - pads are pins between filters
  - source pad is connected to sink pad
  - data type is negotiated using capabilities
  - element, bin, pipeline
- three packages of plugins: The Good, the Bad and the Ugly

GStreamer



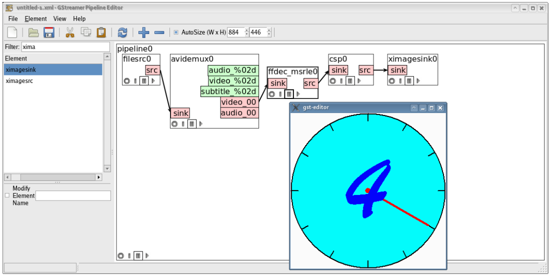
Pipeline construction

```
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
```

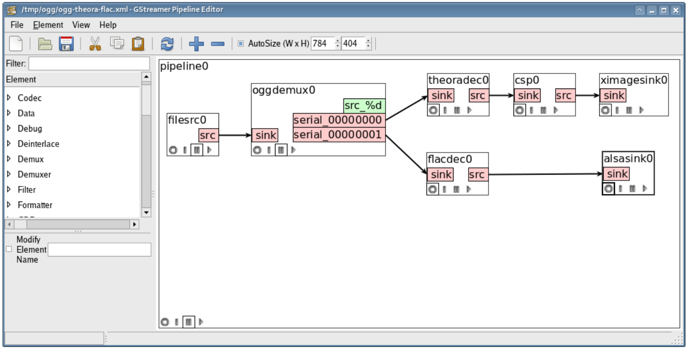
GStreamer

Player

```
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



Save/load 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

restrictions on input and output

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
```

- codec: compile just the plugin

Summary

- multimedia framework (filter graph, framebuffer, pixel format)
- structure of a player, codec (functions)
- Video for Windows
- DirectShow
- FFmpeg
- GStreamer