# ZX81 video encoder and player v0.3

## Quick intro

An encoder to convert videos to the RVC-format, playable on the ZX81.
A player for the ZX81 to play RVC files.
RVC = Retro Video compression, Character based
RVC images are 1/6th the size of ordinary hires images. It all comes down to uncomressing the data on-the-fly with the hardware acceleration available, the character set!

## Requirements

The player requires a ZX81 with a ZXpand classic or ZXpand+, together with a UDG4ZXpand or Chroma81 to get 128 user definable characters. Source code is included, use ZX-IDE to assemble it.

The encoder is distributed as C++ source code, developed on a Linux-machine. Should compile on most modern platforms, but no effort has been spent to test that.
There is also a windows binary and required DLLs included. I am a complete novice at cross-compiling like this, so anything might be wrong!

## Usage of enclosed player and video files

### Physical ZX81

Connect the required HW and make sure it is configured for 128 character mode.
Add CAM25WS.P and CAM25WS.RVC to the same folder on the SD-card and start the video:

- LOAD “CAM25WS”

### EightyOne emulator

The tested v1.12 unfortunately plays the videos slower in EO, but the images look correct.
Settings:

- RAM pack = 32k
- chr$ generator = CHR$16
- high resolution = none
- ZXpand+ = yes
- Enable RAM in 8k-16k = yes
- NTSC TV = no

Add the player CAM25WS.P file and the video CAM25WS.RVC to the SD-card folder and start the video with:

- LOAD “CAM25WS”

## Enclosed videos

The enclosed videos are encoded from the Blender Foundation sponsored films Big Buck Bunny and Caminandes 3: Llamigos, both released under the Creative Commons Attribution 3.0 license.
The movie files used can be found at:
bbb_sunflower_1080p_60fps_normal.mp4
caminandes_llamigos_1080p.mp4

Each video has a player of its own, they share the name: BBB25WS.P is the player for the BBB25WS.RVC video file that should reside in the same folder when run.

# Background

Since releasing my second player and video for Bad Apple, I've been working on enhancing the encoder to work on ANY video material. A lot of time has been spent learning about new algorithms and testing what fits best in the scope of this encoder. I coded this in C++ to reap the speed benefits of compiled code. This is also my first real multi-threaded program to get even more speed :D

# Encoding

This is not a complete manual, just a few examples on how to use the programs.
NOTE: when writing to an RVC file, the encoder overwrites the file if it already exists.

## Example 1

- Convert a video to raw grayscale, from 60fps to 25fps, scale to fullscreen. The result is anamorphic, but that’s OK.
  - ffmpeg -i BigBuckBunny_sunflower_1080p_60fps_normal.mp4 -r 25 -an -pix_fmt gray -vf "scale=256:192" -f rawvideo BBB25FS.RAW
- Encode the resulting raw video to a ZX81 RVC-file, fullscreen, use 8 threads for faster encoding on an aging i7 CPU. Do a clip from frame 1514, 1091 frames long.
  - rrvcencode --start 1514 --frames 1091 --threads 8 --image fs BBB25FS.RAW BBB25FS.RVC
- Change the parameters in the player source and assemble with ZX-IDE, rename the result to BBB25FS.P
- Copy the file BBB25FS.RVC and BBB25FS.P to an SD-card, insert into the ZXpand and start with LOAD "BBB25FS"
- Enjoy the show!

## Example 2

- The video editing program VirtualDub2 for windows is a good fit for the conversion as it has a raw video export. The same input file can be converted to the proper aspect ratio 16:9 at 256x144 pixels, and 30fps.
- Encode the resulting raw video to a sequence of TGA-files, widescreen, use 8 threads. Only convert 1091 frames from frame 1514.
  - ./rvcencode --start 1514 --frames 1091 --threads 8 --output seq bbb25ws.raw bbb25ws
- Convert the resulting TGA files to an animated GIF. Play the GIF at 30fps.
  - ffmpeg -r 30 -i bbb25ws%06d.tga bbb.gif
- View the result on a modern computer when away from the ZX81 :)

# Technical

Streaming video data from the ZXpand and compressing video to a character based mode is challenging. The ZXpand classic limits the playback framerate for fullscreen video because it can't move data any faster than roughly 512 bytes per frame! Getting 8 bytes, or 64 pixels, set every 16 Z80-cycles is a VERY good deal. My goal has always been to play fullscreen at 25 frames per second.

It would be easy to make a player exclusively for the Zxpand+, it can move even more data per second. But that is another project, this is all about being compatible with the Zxpand classic and its limits.

Two formats are supported, fullscreen (256x192 pixels) and widescreen (256x144 pixels).

Fullscreen format uses 1024 bytes per frame. 768 bytes are used for the image and 256 bytes for updating the character set. Every frame gets 32 new character definitions, and pushes out the 32 oldest. 16 of the 32 characters are locked to greyscale patterns every fourth frame. 3 frames of old characters plus 1 frame of the current makes for a rather pleasing video with only 128 unique characters per frame.
50Hz mode can play 25 frames per second. 60Hz mode can only play 15fps, there is not enough free CPU-time to reach the desired 30fps.

Widescreen format uses 768 bytes per frame. 576 bytes for the image and 192 bytes for updating the character set. 32 of the 128 characters are locked to greyscale patterns while each frame updates 24 characters, so there are 3 frames of old characters plus 1 frame of the current here as well.
50Hz mode can play 25 frames per second. 60Hz mode can play 30fps

The encoder takes 8-bit greyscale video at 256x192 or 256x144 pixels and splits it into 8x8 blocks. These blocks are carefully chosen, compressed and dithered to be represented by 128 characters and also fit the bandwidth limits of the ZXpand classic.

The player has to be assembled for each new video file to set the required filename, number of frames and image format.

The player is hardcoded to produce the framerate chosen, regardless of the HW setting of the ZX81.

The encoder source code is not pretty, but it is decently commented (I hope). The player source code is generously commented, but is a cut-and-paste job from several older projects and could use a cosmetic tune-up. I hope to fix all this in the future.

2019-09-01
NollKollTroll