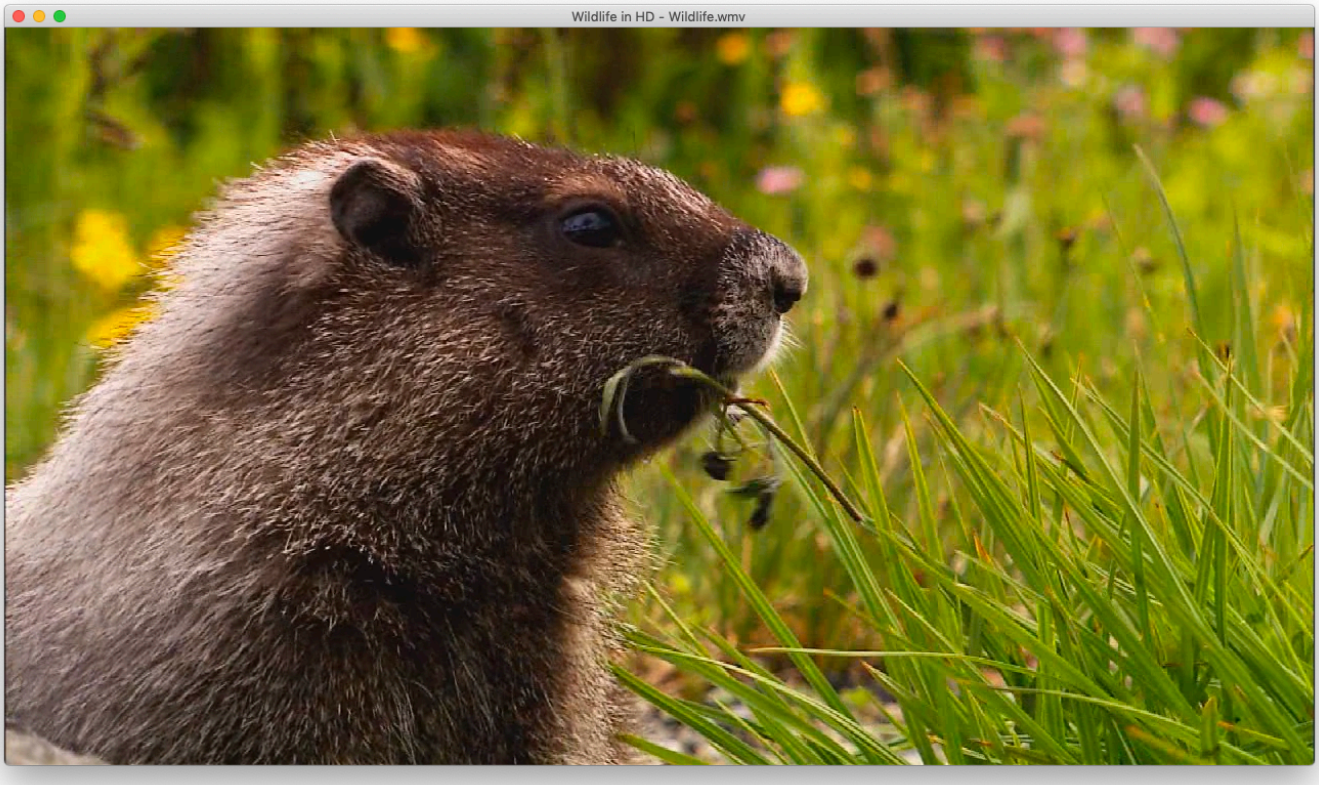
redCV and FFmpeg: Video reading

As previoulsy explained, Red and FFmpeg collborate rather well for video and audio recording. In this new post, we'll focus on video reading.

A simple approach: Use ffplay tool

A simple, but efficient way, is to call *ffplay tool* which is included in FFmpeg framework. Just open a terminal session, and give the name of the movie to read.

```
ffplay Wildlife.wmv
```



While running movie, different options can be used form controlling the ouput:

```
f: for fullscreen
```

```
m: toggle audio
```

```
p, space: pause/resume the movie
```

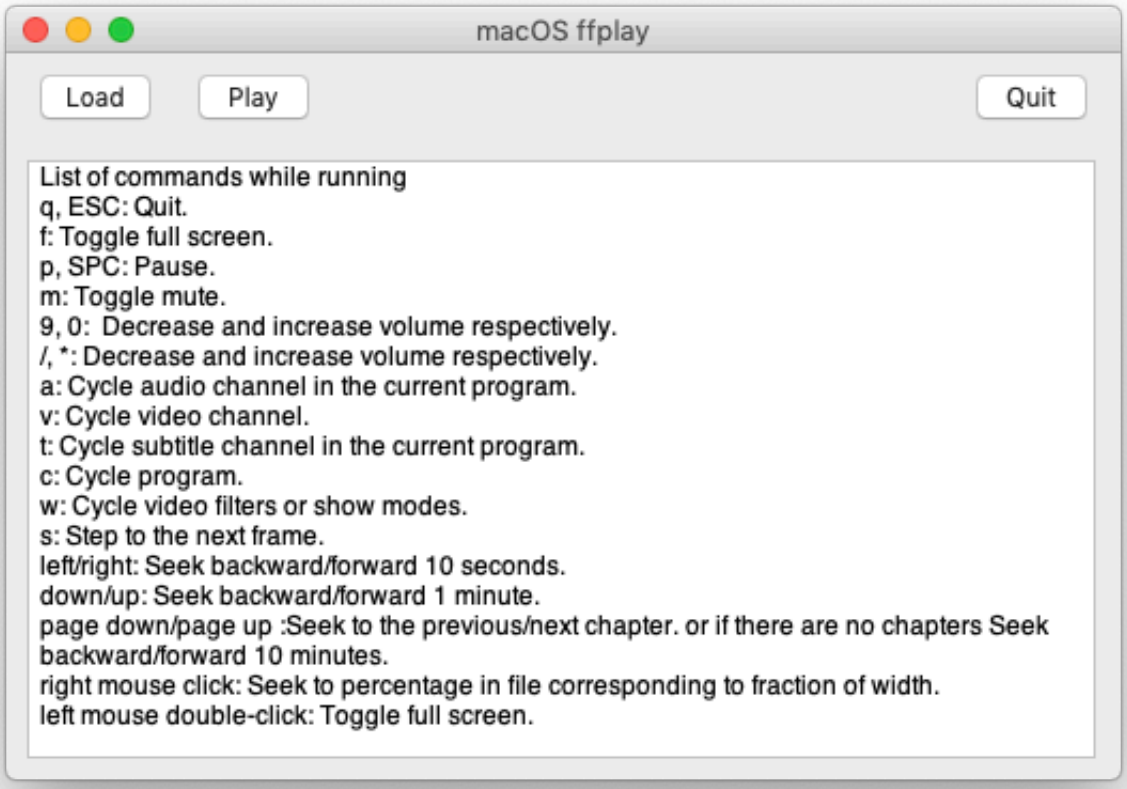
```
/, *: decrease and increase volume respectively
```

During the reading of the movie, ffplay returns a lot of information about the video:

```
Input #0, asf, from 'Wildlife.wmv':
  Metadata:
    SfOriginalFPS   : 299700
    WMFSDKVersion   : 11.0.6001.7000
    WMFSDKNeeded    : 0.0.0.0000
    comment         : Footage: Small World Productions, Inc; Tourism New Zealand | P
roducer: Gary F. Spradling | Music: Steve Ball
    title           : Wildlife in HD
    copyright       : © 2008 Microsoft Corporation
    IsVBR           : 0
    DeviceConformanceTemplate: AP@L3
  Duration: 00:00:30.09, start: 0.000000, bitrate: 6977 kb/s
    Stream #0:0(eng): Audio: wmav2 (a[1][0][0] / 0x0161), 44100 Hz, 2 channels, fltp
, 192 kb/s
    Stream #0:1(eng): Video: vc1 (Advanced) (WVC1 / 0x31435657), yuv420p, 1280x720,
5942 kb/s, 29.97 fps, 29.97 tbr, 1k tbn, 1k tbc
  25,37 A-V: -0,031 fd=   0 aq=   62KB vq= 2751KB sq=    0B f=0/0
```

Of course, this command can be integrated in Red code as parameter of call function, and GUI program can be developped to avoid command-line use (see */video/ffplay.red* code). It is important to **use /shell refinement for call**, since ffplay uses the terminal to display different information:

```
call/shell "ffplay Wildlife.wmv"
```



This simple approach is very confortable for reading and listening all supported video files and, this is the way I prefer for reading movies without using VLC or Quick Time Player.

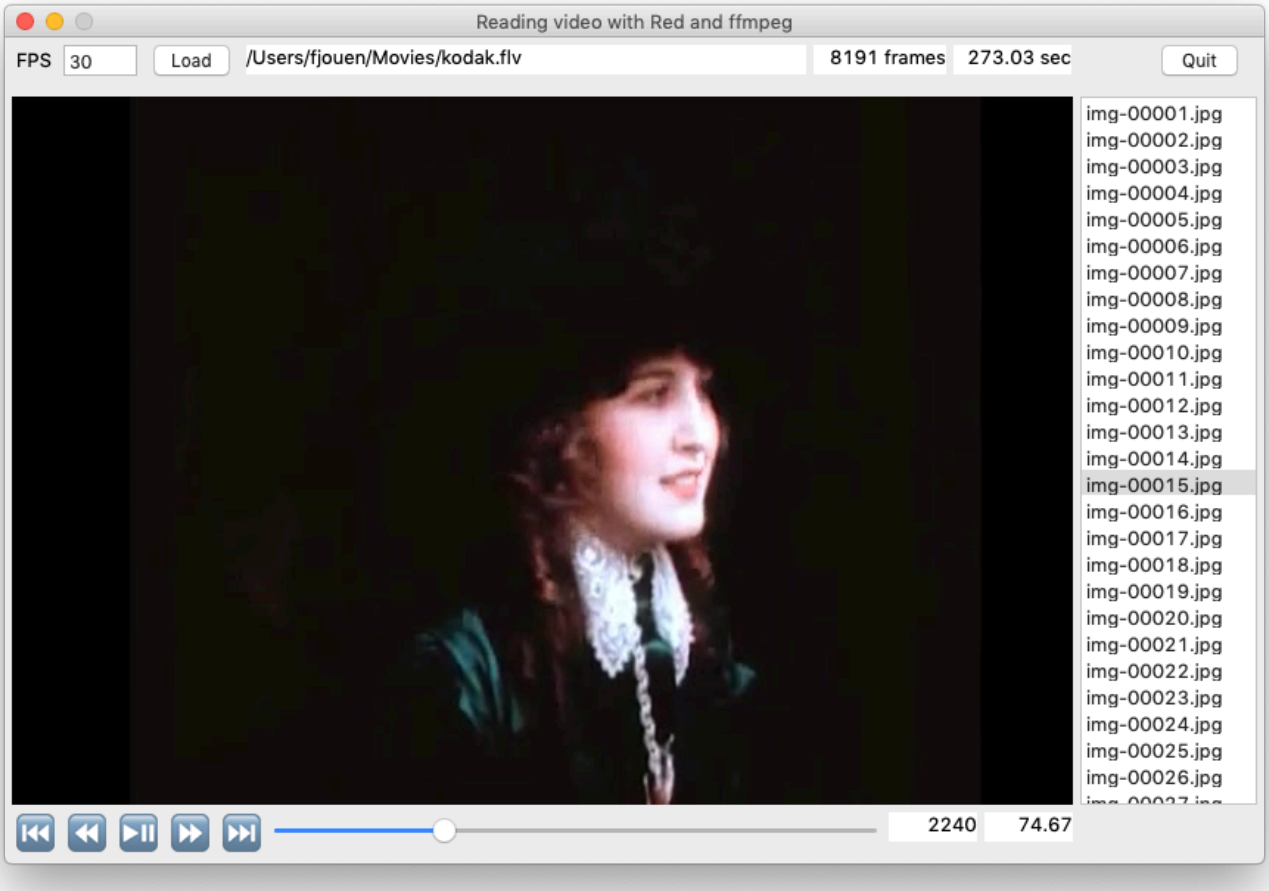
A second approach: Extract frames from a movie

In many applications, I don't need tp process audio channel, but I have to focus on images in order to make redCV image processing on time-lapse videos.

To turn a video to number of images, run the command below. The command generates the files named image0001.png, image0002.png and so on.

```
ffmpeg -i filename image%000d.png
```

This command is included in *video/movies.red* program and, associated to Red code for creating an efficient video reader with a clean GUI interface.



The code is very simple, but contains some intersting functions.

First of all, to create elegant navigation buttons, we profite the fact that Red supports unicode string :)

```
; for Unicode string with images
makeUnicodeStr: func [s [string!] c [char!] return: [string!]][
    rejoin [s form c]
]

s1: makeUnicodeStr "" #"^(23EE)"    ;First frame
s2: makeUnicodeStr "" #"^(23EA)"    ;Next frame
s3: makeUnicodeStr "" #"^(23E9)"    ;Previous frame
s4: makeUnicodeStr "" #"^(23ED)"    ;Last frame
s5: makeUnicodeStr "" #"^(23EF)"    ;Play/Stop movie
```

The second impportant function concerns the generation of line-command for FFmpeg.

```
generateCommands: func [] [
    blk: copy []
    blk: rejoin [
        "/usr/local/bin/ffmpeg"
        " -y"
        " -i '" fileName "'"
        " -f image2"
        " -s 720x480"
        " -q:v 1"
        " -r " fps
        " " dataDir
        "img-%05d.jpg"
    ]
    form blk
]
```

FFmpeg options are very simple:

`"/usr/local/bin/ffmpeg" ;location of ffmepg binary`
`" -y" ;automatically replace image files`
`" -f image2" ;The image file muxer writes video frames to image files`
`" -s 720x480" ;output size`
`" -q:v 1" ;use fixed quality scale (1 to 31, 1 is highest)`
`" -r " frames per sec ;fps (mandatory for .vmw files)`
`" " dataDir ;temp destination directory`
`"img-%05d.jpg" ;automatic file name numbering`

The rest of the code is pure Red and very easy to understand. First of all, you have to select a movie file. When done, FFmpeg is called to create, in a temprary directory, all jpg images corresponding to the number of frames contained in the movie. You can also play with the FPS to create more or less images. Navigation buttons, slider and, text-list faces can be used to directely access any image. When you press the Play/Stop button, for a complete reading, text-list face is disabled. All code is here: https://github.com/ldci/ffmpeg/video/movies.red.