How to use eli module for Python

Heiko Noordhof

July 3, 2008

How to use eli module for Python
by Heiko Noordhof

Published 2007

Contents

1 Introduction

The eli module for python can be used to create and edit "edit list" (.eli) files. These files are used by the MJPEG-tools to describe modifications to video files without actually changing the video files. An .eli can be played directly or converted to a movie file using the MJPEG-tools .

This document shows how to use the python module "eli" by example. Without the MJPEG-tools For more information about how to use the MJPEG-tools refer to the MJPEG HOWTO or start reading this article on linux.com .

2 What is an edit list (.eli) file?

An edit list file (eli file) is a text file that describes which frames, of which video file (.avi, .mpeg, .mov etc.) to play in sequence. This way you can edit a movie by editing a relative simple text file. When finished you can generate the final movie using the MJPEG-tools.

An eli file has a list of paths to video files (called *file list*) and a list of *frame blocks* that are played in secuence. *Frame blocks* refer to a movie file in the file list and have a *start frame* and a *stop frame*.

Example 2.1 Example eli file:

```
LAV Edit List                ❶
PAL                          ❷
3                            ❸
/pathto/video1.avi           ❹
./../tubes/./film3.mpeg      ❺
/pathto/movie2.mov
0 100 200                    ❻
1 1000 1999
2 400  600  # comment        ❼
1 1 1000                     ❽
```

❶ Header, 'Magic number'. Simply identifies this file as MJPEG eli file.

❷ Video type. Can be either *PAL* or *NTSC*.

❸ Number of files in the *file list*.

❹ The first video file in the *file list* . It has index `0` implicitly. The files in the *file list* are referred to by zero-based indices.

❺ Two more paths to video files (index `1` and `2`) follow to make a total of 3 files, as indicated by the length of the *file list* on line 3.

❻ The first *frame block*. Three integers on a single line, seperated by white space:

- `0`: Refers (zero-based) to the first file in the *file list*: `/pathto/video1.avi`.
- `100`: Start frame. Start playing the file from frame 100.
- `200`:: Stop frame. Play the file until frame 200 (inclusive).

❼ The MJPEG-tools only read the first three integers from the *frame block* line. Everything more is ignored and can be used as comment. Not even the comment character '`#`' is needed.

❽ Note this *frame block* plays another part of the same file as the the second *frame block* (same index).

3 Using the python eli module

The python eli module helps editing *eli files* interactively in a python shell, or as a backend module in you own eli-editing scripts or maybe even application scripts. The eli module keeps track of the video files needed in the *file list* and the indices in the *frame blocks*as video files are deleted or added.

3.1 Initializing

There is only one object class of real interest in the eli module. This class is called `Eli` (note uppercase 'E'). So, the recommended way to start using the eli module in python is:

```
>>> from eli import Eli
>>>
```

3.2 Creating eli objects

You can create an empty eli object and append and insert *frame blocks* like this:

```
>>> e = Eli()
>>> e.append("/pathto/video1.avi", 100, 200)                ❶
>>> e.append("/pathto/movie2.mov", 400, 600, "# commment")  ❷
>>> e.append("./../tubes/./film3.mpeg", 1, 1000)
>>> e.insert(1, "../tubes/film3.mpeg", 1000, 1999)          ❸
>>>
```

❶ A frame block to play can be appended by calling the eli method `append`. The file is automatically added to the *file list* of the eli object and the correct index to the file is generated.

❷ Optionally some comment can be passed as a 4th parameter.

❸ You can call the method `insert` to insert a *frame block* at a specific place. Pass the frmae block index as the 1st parameter (here: `1`)

Note that, though the string containing the file path is different from the previous method call, it actualy refers to the same file. This is automatically resolved and merged with the other. The indices to the file list for these frame blocks will be the same as we will see later.

An eli object can be printed like most python objects. The string output is like the contents of an eli file.

```
>>> print e
LAV Edit List
PAL
3
/pathto/video1.avi
../tubes/film3.mpeg
/pathto/movie2.mov
0        100      200
1       1000     1999
2        400      600    # commment
1          1     1000
>>>
```

The object can of course be saved to an eli file:

```
e.save("/tmp/example.eli")
```

An eli object can also be created by loading it from a file. This can be done in two ways:

```
e = Eli(file("/tmp/example.eli"))
e = Eli("/tmp/example.eli")
```

The first way is to pass a `file` object to the `Eli()` constructor. The second is to pass a string to the constructor. The subtle difference is that in the second way the string argument is detected to be a path to an eli file from the fact that the string ends in ".eli". So if you have a eli file with a name not ending in ".eli", the second way would cause an exception (error). The first way always works.

This is done because you can also create an eli object from a string:

```
>>> elistr = '''LAV Edit List
PAL
3
/pathto/video1.avi
../tubes/film3.mpeg
/pathto/movie2.mov
0        100      200'''
>>>
>>> e = Eli(elistr)
```

Note that the *file list* has 3 files, two of them are not used in this eli object. This managed automatically by the `Eli` class. When you print or save this object, the unneeded file are removed from the list:

```
>>> print e
LAV Edit List
PAL
1
/pathto/video1.avi
0        100      200
>>>
```

The last way to create an eli object is to make a *deep copy* of another:

```
>>> f = Eli(e)
>>> print f
LAV Edit List
PAL
1
/pathto/video1.avi
0        100      200
>>>
```

3.3 Editing eli objects