JEVisualizer documentation

Hannes Ihalainen

October 13, 2018

Contents

1 Introduction

JEVisualizer is a program that creates a visualization video for music. The development started, when we wanted to add visualizations for our music on YouTube. One important goal is to develop JEVisualizer in such a way that it can be configured to create very different visualizations - only imagination should be the limit.

JEVisualizer is designed to be used by editing configuration files. N

2 How does it work

The basic idea is that JEVisualizer takes as an input any number of *tracks*, and outputs a video that visualizes those *tracks*. *Atrack* contains a number of *notes*, which get different intensities. Currently the program can extract tracks from the channels of wav-files (using an algorithm that detects frequencies) or from the tracks of mmp-files (Linux multimedia studio -files).

For the output-video, any number of *layers* can be configured. *Alayer* can contain a picture, any number of *drawers* and any number of *filters*.

Adrawer visualizes some notes from some track(s).

Filters are used to create cool effects on layers or output-video.

More about all these in section "How to config".

3 How to config

JEVisualizer contains a number of different config-files.

First is main.config. It contains information about tracks and the name of visualizer-config file.

Second is visualizer.config. It defines basic information about the output-video, like width, height, fps, filename etc. It also contains the layers as filenames.

Thirdly there can be any number of layer-config-files. Each layer-config-file contains information about that layer. It can contain a background-image, information about when the layer is visible, a number of drawers and a number of different filters.

From configuration-files it is also possible to include other configuration files, so there could also be a different config-file for some drawers or filter etc.

3.1 JEVisualizer configuration file format

JEVisualizer uses its own syntax in configuration files.

Here are the basic rules of JEVisualizer configuration syntax: Configuration files are of a simple attribute-value format. Each attribute and value -pair must be on a single line (except when value is inside ””, ” or , when it can contain linebreaks). Basic format is: **attribute=value**.

spaces and tabs	spaces and tabs don’t matter (except in values inside ””, ” or {})
linebreaks	linebreaks do matter, they separate different attribute-value pairs.
commenting	// comments the end of line, multiline comments can be written between /* and */.
””, ” and	Values containing spaces and linebreaks can be set inside ””, ” or {}. ””, ” and {} do note have any functional differences. Inside of ””, ” or {} characters can be escaped using character. Note: p={c={...}} wouldn’t work as expected, something like p={c=\{...\}} should be used.
including other files	Special command: #include=”filename”, includes config file ”filename”.
#-character	#-character in attribute is the delimeter between name and key. (for some attributes there can be different keys...)
UPPER and lower case	Parameter names will be converted to smallcase when reading file. A=b is same as a=b. -# is the delimeter between parameter name and key

Example:

```
PARAMETER1=VALUE
PARAMETER2="VALUE WITH SPACES"
PARAMETER3={
  P=1
  P=2
}
#include="file2.config"
PARAMETER4#KEY1=VALUE1
PARAMETER4#KEY2=VALUE2
```

For further details see implementation in *config.h* and *config.cpp*.

3.2 main.config

Following pa

ant	really busy all the time
chimp	likes bananas
alligator	very dangerous animal, sharp teeth, long muscular tail and a bit of text that is longer than one line and shows the alignment of text quite nicely

3.3 Formulas

Most of the values that are used are given as datatype Formula (defined in formula.h and formula.cpp). The value of a formula is calculated again for every frame. Formula can be either simple constant or it can depend on many variables.
1. $CONSTANT$
simple floating-point number
2. ($CONSTANT$, [$ARRAY\_OF\_VARIABLE_COEFFICIENTS$], MIN_VALUE, MAX_VALUE, $SIN\_COEFFIC$
Value of this type of formula is:
$min(max(CONSTANT+ARRAY_i*VARIABLE_i(foreachi)+SIN\_COEFFICIENT*sin(IN\_SIN), MIN\_VALUE),$
Each parameter of is given as another formula
Formula can have all parameters or n first of them, like for example ($CONSTANT$, [$ARRAY\_OF\_VARIABLE_COEFFIC$

Formulas are defined in following format: Variables the formulas can use are also defined in config-files: $FPV\#i = name$ where *name* is some valid variable (see list below)

Examples of valid formulas: 5 Formula has constant value 5. (5) Formula contains a constant formula with value 5 -¿ does same as formula 5, but is a bit slower to calculated. (5, [1]) Formula has value $5 + FPV\#0$ (0, [], -1, 1, 1, (0, [1])) Formula has value $sin(FPV\#0)$. (Min and max don't have effect since the value of *sin* is always between -1 and 1)

Different variables can be defined in different scopes. Here is a list (this list is not necessarily up to date, but at least we have a list) In visualizer: frame index of the frame fps fps of output-video w width of output-video in pixels h height of output-video in pixels null or 0 unset parameter

Example:

In file visualizer.config:

```
FPV#0=frame
LAYER#0="layer0.config"
```

in file layer0.