Cpfg, Lpfg, L-studio – file formats

Colourmap

Colourmap file is a binary file. It should have an extension `.map`. The must be at least 768 bytes long. Triples of consecutive bytes are treated as R, G and B values of colours. First 256 colours are read. If the file is longer the remainder of the file is ignored.

Material

Material file is a binary file. It contains one or more records of the form:

```
struct materialrecord
{
  unsigned char id;
  unsigned char transparency;
  unsigned char ambient[3];
  unsigned char diffuse[3];
  unsigned char emission[3];
  unsigned char specular[3];
  unsigned char shininess;
};
```

`id`	Material number.
`transparency`	Transparency value applied to all material components (ambient, diffuse, emission and specular). 0 is opaque (alpha value equal to 1.0), 255 is full transparency (alpha equal to 0)
`ambient`	R, G and B values of ambient component.
`diffuse`	R, G and B values of diffuse component
`emission`	R, G and B values of emission component
`specular`	R, G and B values of specular component
`shininess`	Shininess parameter. Must be in the range [0, 128]

Function format description

Original format, as implemented in cpfg4.0:

```
range: 0.0 1.0
points: num of points
x1 y1
x2 y2
…
xn yn
```

Format ver. 1.01 introduced in L-studio v. 2.1, cpfg v. 6.5

```
fver 1 1
name: name
samples: samples
flip: on|off
points: num of points
x1 y1
```

x_2 y_2
…
x_n y_n

All subsequent versions of L-studio and cpfg should be able to read the original format as well as the new one.

Gallery format:
funcgalleryver 1 1
items: *items*
$Item_0$
$Item_1$
…

The items in a function gallery must be in format version 1.1 or newer.

Contour format description

Original format, as implemented in cpfg3.0:

n d open|closed
x_1 y_1 z_1
x_2 y_2 z_2
…
x_n y_n z_n

n – number of points, d – dimension

Format ver. 1.01 introduced in L-studio v. 2.1, cpfg v. 6.5
cver 1 1
name: *name*
points: *n_1 n_2*
type: open|closed
x_1 y_1 z_1 m_1
x_2 y_2 z_2 m_2
…
x_n y_n z_n m_n

m_i is multiplicity of a point. n_1 and n_2 in the third line are: number of control points and number of control points including multiplicity respectively.

Format ver. 0.1 is introduced for compatibility with Lars' editor curves:

version: 1.4
contact: 0 0 0
end: 0 0 0
heading: 0 1 0

```
up: 0 0 –1
size: 1
points: n
range: 0.0 1.0
dimension: 4
type: bspline
x1 y1 0 m1
x2 y2 0 m2
…
xn yn 0 mn
```

n is the number of control points. m_i is the multiplicity of control point. L-studio contour editor ignores the lines contact, end, heading, up, size, range, dimension and type when reading. When writing it will always put the default data for these lines as above.
Remark: This format does not support closed contours.

Format ver. 1.02:
Implements bspline with the end point interpolation.

```
cver 1 2
name: name
points: n1 n2
type: {o|c}{r|e}
x1 y1 z1 m1
x2 y2 z2 m2
...
xn yn zn mn
```

The only difference between this and the original format version 1.01 is that the type includes now two independent flags o|c for open|close and r|e for regular or endpoint interpolation type

Format ver. 1.03
Implements “number of samples” field.

```
cver 1 3
name: name
points: n1 n2
type: {o|c}{r|e}
samples: samples
x1 y1 z1 m1
x2 y2 z2 m2
...
xn yn zn mn
```

Gallery format:

```
contourgalleryver 1 1
items: items
Item0
Item1
…
```
The items in a contour gallery must be in format version 1.1 or newer.

Beziér surfaces

Old format is described in the vlab documentation.

Format ver. 1.01

```
sver 1 1
name: name
patches: n
```

Demo file

First four bytes is the checksum:

Random number	Low-byte	Random number	Hi-byte

From this point on, the contents is scrambled in the following way:

Each four bytes are shuffled like this:

1 4 2 3

and then bitwise not (or the other way around).

The files in the demo file are stored as follows:

- Zero-terminated file name,
- Size of the file stored as a little-endian 32-bit integer,
- Contents of the file.

File name that consists of eight characters ‘*’ (ASCII 42) specifies the end of the tar file. The checksum should be calculated as follows: the sum of all bytes stored in the tar file *before* scrambling. Checksum is an unsigned short int (2-bytes long).

This format is implemented in the classes ScrambleFile and UnscarmbleFile (scrmbl.h, scrmbl.cpp).