Definition of curve bundles format (`*.bundles`) used in *BrainVISA*

Yann Cointepas

June, 30th 2004

1 Introduction

This document describes the main format used by *BrainVISA* and *Anatomist* to represent name set of curves. The aimed audience is programmers who wish to generate bundles with their own software and use them with *BrainVISA*/*Anatomist* [1].

2 Format description

The bundle format is composed of two files. One human readable file with the extension `.bundles` and one data file with the extension `.bundlesdata`. The `*.bundles` file contains a generic header with the same format as `*.minf` files of *BrainVISA*. See the "*Definition of meta-information format (`*.minf`) used in* BrainVISA " document to for a desctiption of the `*.minf` format.

A `*.bundles` file contains the following fields :

format (required): Format identifier. A format identifier uses the following pattern: `<format_name>.<major_version>.<minor_version>`. To date, `'bundles_1.0'` value is required.

space_dimension (optional, default = 3): Space dimension. Must be 3 for `bundles_1.0`.

curves_count (required): Number of fibers in data file (required). Must be ¿ 0 for `bundles_1.0`.

data_file_name (optional, default = `'*.bundlesdata'`): File where to find the data. Must be `'*.bundlesdata'` for `bundles_1.0`.
A star (`*`) in the file name is replaced by the header file name without its extension. This file contains a series of curves. Each curve is a series of points. Each point is a series of `space_dimension` coordinates.
In ascii mode:

[1] See `http://brainvisa.info` for more information about *BrainVISA*/*Anatomist*.

- A *coordinate* is a decimal number.
- A *point* is `space_dimension` space-separated *coordinates*.
- A *curve* is a comma separated series of *points*.
- There is one and only one *curve* per line.

In binary mode

- A *coordinate* is a 8 bytes floating point number (a `double` in C or C++).
- A *point* is the concatenation of `space_dimension` *coordinates*.
- A curve is a 4 bytes integer representing the number of points and a series of points.

3 Fields description

mode*:** The format can be written either as an *ascii* text file or as a *binary* file. The ***mode is used to identify the representation it can have three values :

- `'ascii'`: the file is in text format.
- `'binarABCD'`: the file is in binary format and uses *big-endian* byte order for numbers (such as Motorola or Sun processors for example).
- `'binarDCBA'`: the file is in binary format and uses *little-endian* byte order for numbers (such as Intel processors for example).

textureType*:** The file format was created with the possibility to include a texture. But this is *never used* since textures are represented in a separate format. In *ascii* mode his field should always contain `'VOID'` or, in *binary* mode, a ***U32 containing 4 (which is the size of the following string) followed by the four characters `'VOID'`.

polygonDimension*:** This field is an ***U32 containing the number of points of each polygon. The following values are supported in *Anatomist* and *Aims*:

3: Polygons are composed of three points (they are triangles). This is the recomanded value for surfaces because other values may not be supported by all *BrainVISA* processing tools.

4: Polygon are composed of four points. This is supported in *Anatomist* but may not be supported in every *BrainVISA* processing tools.

2: This value is used to represent segments in a 3D space. Each ”polygon” is composed of two 3D points.

numberOfTimeSteps*:** The mesh format can represent several meshes at different time steps. This is a ***U32 representing the number of time steps.

timeSteps*:** This field contains ***numberOfTimeSteps times the following structure :

instant*:** a ***U32 representing a time instant.

***vectorOf*<*vertex*>:** contains all the vertices which are used to build polygons.

***vectorOf*<*normal*>:** contains the normals of the surface at each vertex. It must have the same size as ***vectorOf*<*vertex*>** or be empty.

vectorOf*<*texture*>:** must be an empty vector (i.e a ***U32 containing 0).

***vectorOf*<*polygon*>:** contains the polygons which represent the surface.

vertex*:** is 3D a point. In *ascii* mode it has the following syntax: `'('` ***FLOAT `','` ***FLOAT*** `','` ***FLOAT*** `')'`. In binary mode it is represented by three ***FLOAT***.

normal*:** is a normalized vector. In *ascii* mode it has the following syntax: `'('` ***FLOAT `','` ***FLOAT*** `','` ***FLOAT*** `')'`. In binary mode it is represented by three ***FLOAT***.

polygon*:** is a set of ***polygonDimension points. Each point is represented by a ***U32*** which is an index in ***vectorOf*<*vertex*>**. The first vertex index is *zero*, the second is one, etc. In *ascii* mode it has the following syntax: `'('` ***U32*** `','` ***U32*** `','` ... `','` ***U32*** `')'`. In binary mode it is represented by a series of ***polygonDimension*** elements of type ***U32***.

U32*:** A 32 bits wide unsigned integer (between 0 and 4294967295). In *ascii* mode it is written as a decimal number. In *binary* mode it is represented on four bytes with the choosen byte order (see ***mode above).

FLOAT*:** A 32 bits wide real number (maximum 3.40282347e+38). In *ascii* mode it is written as a decimal number. In *binary* mode it is represented on four bytes with the choosen byte order (see ***mode above).

vectorOf*<field>:** where ***field is a field type. It represents a fixed length vector of elements of type ***field***. It contains the size of the vector (i.e. the number of elements) as a ***U32*** followed by the elements.

***space*:** A byte with one of the ascii value for a space, a tabulation or a carriage-return.

4 Examples

Here is an example of an *ascii* mesh file containing a tetrahedron.

```
ascii
VOID
3
1
0
4 (-0.8,0.8,0) (0.8,8e-1,0) (-1,-1,0) (0,0,1)
4 (-0.8,0.8,0) (0.8,8e-1,0) (-1,-1,0) (0,0,1)
0
4 (0,1,2) (0,3,1) (1,3,2) (2,3,0)
```

Here is an example of an *ascii* mesh file containing a linear spiral.

```
ascii
VOID
2
1
0
16
(10, 0, 0) (7.07, 7.07, 0.4) (0, 10, 0.8)
(-7.07, 7.07, 1.2) (-10, 0, 1.6) (-7.07, -7.07, 2.0)
(0, -10, 2.4) (7.07, -7.07, 2.8) (10, 0, 3.2)
(7.07, 7.07, 3.6) (0, 10, 4.0) (-7.07, 7.07, 4.4)
(-10, 0, 4.8) (-7.07, -7.07, 5.2) (0, -10, 5.6)
(7.07, -7.07, 6.0)
0
0
15
(0,1) (1,2) (2,3) (3,4) (4,5) (5,6) (6,7) (7,8) (8,9)
(9,10) (10,11) (11,12) (12,13) (13,14) (14,15)
```