IGB Image Format

Edward J. Vigmond

June 20, 2000

1 Introduction

IGB format is a file format which was developed at the Université de Montréal for their 4D data sets. Since it was developed at a French university, many of the key words are not misspelled English words; they are French words . The IGB format consists of 2 parts: 1) an ascii header describing the file followed by 2) the data written in binary format.

2 Image Header

The header is usually 1024 bytes long and written in ascii. Furthermore, the header is divided into lines of maximum length 80 charactrers which are seperated with new line characters ('\n'). Each line contains keyword and value pairs seperated from each other by a space. A colon seperates the keyword from its value. Keywords are described in the table below:

Keyword	Type	Description	Default	Notes
x	Int	number of pixels in x direction	0	mandatory
y	Int	number of pixels in y direction	0	mandatory
z	Int	number of pixels in z direction	1	Equals 1 for 2D data
t	int	number of time slices	1	
type	string	data type	-	Possible values: *byte,char, short, long float, double, complex, double_complex, rgba, structure, int, uint*
taille	int	structure size	0	For a data type of *structure*, this is its size in bytes, otherwise ignored
struct	char[40]	structure description	""	a 40 character long string describing the structure
systeme	string	system type	-	Recommended values are *big_endian*, and *little_endian.* Only important for determining byte order.
dim_x	float	physical size in x	1	
dim_y	float	physical size in y	1	
dim_z	float	physical size in z	1	
dim_t	float	length of time	1	
fac_x	float	scaling factor in x	1	
fac_y	float	scaling factor in y	1	

fac_z	float	scaling factor in z	1	
fac_t	float	scaling factor for t	1	
org_x	float	x coordinate of first pixel	1	
org_y	float	y coordinate of first pixel	1	
org_z	float	z coordinate of first pixel	1	
org_t	float	time of first pixel	0	
inc_x	float	distance between pixels in x	1	actual x = org_x + pixel_x*inc_x
inc_y	float	distance between pixels in y	1	actual y = org_y + pixel_y*inc_y
inc_z	float	distance between pixels in z	1	actual z = org_z + pixel_z *inc_z
inc_t	float	time between slices	1	actual t = org_t + t*inc_t
facteur	float	scaling factor for data	1	data = zero + data_value*facteur
zero	float	origin for data scaling	0	
aut	char[40]	author	“”	a 40 character long string without spaces
unites	string	units for data	“”	
unites_x	string	units for x	“”	
unites_y	string	units for y	“”	
unites_z	string	units for z	“”	
unites_t	string	units for t	“”	
comment	string	comment	“”	
transparent	string	transparent pixel value	“”	hexadecimal representation of transparent data value without leading 0x

2.1 Example

The following is a sample from a header ignoring all the white space needed to pad it to 1024 characters:
x:50 y:25 t:200 unites:volts type:float transparent:000a30ff
Note that the transparent value is a string whose length is twice the size of the data type. Leading zeros are necessary.

3 Data

Data is written in binary format using, for example in C, the function fwrite(). The data can be of any type which is specified by the *type* keyword in the header. The data may be linearly transformed before writing to the IGB file using a transform of the form

data = (actual_data - zero) * facteur

where facteur and zero are written to the header. The number of bytes written should be equal to x*y*z*t*sizeof(data type). The transparent value refers to the actual data.

4 Routines

To aid in reading and writing IGB headers, C functions and a C++ class are provided.

4.1 C routines

These routines are located in the C subdirectory. One must include “igb.h” to use these functions.

```
int Header_Read_gz( gzFile file, Header *head, Bool_Header *bool_head )
```

Read from the previously opened file and place the header in head and the boolean header in bool_head. Bool_Header is a structure with the same field names as the header but all of type boolean. An entry is true if the corresponding header field is NOT the default value, otherwise it is false. The function returns a 0 iff no error occurred.

```
int Header_Write_gz( gzFile file, Header *head, Bool_Header *bool_head )
```

Write a header to a previously opened file. One must set any fields of head and then set the corresponding field true in bool_head. Any fields which are not set in header should have the corresponding bool_head field set false. This function returns a 0 iff no error occurred.

4.2 C++ Class

The C++ class *IGBheader* is provided. Refer to the file *IGBheader.h* for a list of methods which allow setting and querying of the header fields. To set the transparent value, call the transparent method with a pointer to a variable of the same type as the data which is set to the data value to be ignored:

```
float a=1000;
header->transparent(&a);
```

5 igbhead

A utility program to aid in creating and modifying IGB files is provided. It is called *igbhead.* Only values which differ from the default are stored in the header. To query a header, simply run *igbhead* on the IGB file:

```
igbhead file.igb
```

Individual fields may be set in the header by specifying the appropriate option. To add a header to the data file *data*, the simplest way is something of the form

```
igbhead -xX -yY -zZ --data-type=type --output-file=data.igb data
```

To put an existing header from aa.igb on the file data, add a comment saying we are doing so, and put the result in a new compressed file *bb.igb.gz*:

```
igbhead aa.igb --frankenstein=data -f bb.igb,gz --comment=”stolen head”
```

igbhead recognizes the ".gz" suffix and gzips accordingly. The placement of *aa.igb* is free. Also note that the comment option adds one line of comment to the existing lines. The decapitation option will remove the header, leaving just the data. Therefore, to recover data from *bb.igb.gz*:

```
igbhead --output-file=data -decapitate bb.igb.gz
```

The *systeme* keyword is important in that it determines whether the data is treated as big endian or little endian. To this end, use of any systems other than *big_ endian* and *little_ endian* is deprecated. If there is a problem with word order when displaying an IGB file, use the swap bytes function under image in the menu bar of flounder to correct the problem.