RTViz Voxel File Format

Mitsubishi Electric ITA
Real Time Visualization

1.0 Introduction

This document describes the `vox1999a` file format, a simple and flexible format for voxel data storage, exchange and input and output filter targeting. One of the design goals is that minimum effort would be required to read and render voxel data stored in such a format by Mitsubishi's VolumePro family of volume rendering hardware, while not making the file format specific to the VolumePro.

The proposed file format is operating system and processor independent. It is capable of storing 1, 8, 16, 32 and 64 bit volumes, each of which can contain multiple fields. Multiple volumes can be stored in a single file.

Volumes stored in this format are assumed to be parallelepiped, consisting of slices of voxels with equal distance between the slices.

Although it is recognized that compression is useful in better utilizing the disk space and reducing file loading and transmission time, as of this draft, no compression is proposed.

2.0 File Structure Overview

The proposed voxel file format consists of three logical sections. The file begins with a file header that identifies the file as being the proposed voxel file format and contains descriptions and copyright notices pertaining to the data in the file. The header is followed by a volume description section that describes the first volume and the voxel data in it. The volume data section follows immediately after the end of the descriptor. Each datum in the data section represents a voxel containing one or more fields (for example, an RGB voxel). The volume description and volume data sections may be repeated to provide for multiple volumes in one file.

The file structure can be depicted as follows

File Header	(signature, number of volumes, copyrights, descriptions)
Volume Description	(volume dimension, voxel size, voxel fields)
Volume Data	(voxel data streams, always end at a byte boundary).
Volume Description	
Volume Data	
...	

The file header and the volume description sections contain text in ASCII encoding, organized as multiple name-value pairs called descriptors. Each descriptor must be on a separate line, and the descriptor must be complete on that line (except for the two descriptors that accept multiple fields delimited by parentheses). The name is case sensitive, and leading spaces before the name are allowed, with a few exceptions noted. There must be one or more spaces between the name and value. In the following definitions of descriptors, we use the following parameter types, as shown in the following table:

Name	**Description**
`int`	An integer value. Currently no negative values are useful.
`float`	A floating point value, in a form suitable for "atof".
`word`	Either a sequence of non-blank characters (which must not start with an ASCII double-quote (hex value 0x22)) or a `quoted-string` as described below.
`rest-of-line`	A sequence of characters terminated by the end of line.
`quoted-string`	A sequence of characters starting and ending with an ASCII double-quote (hex value 0x22). This sequence must be complete on one line, and may contain the character sequence backslash double-quote (hex values 0x5c 0x22) which represents the single character double-quote.

The end of line indicator is ASCII newline, hex value 0x0a. Blank characters may be blanks (ASCII value 0x20) or tabs (ASCII value 0x04), but not newlines, except as noted for specific descriptors.

All vox1999a file readers must accept all descriptors described in this document without error. (A warning may be given for non-processed descriptors, but the file reader must be able to continue processing the file.)

3.0 File Header

The voxel data file starts with a file header that identifies the file as being in the `vox1999a` format and describes the content of the file. The file header ends when the end-of-header marker is encountered.

3.1 Signature - required - singular

```
Vox1999a
```

The signature must appear at the beginning of the file (bytes 0-7) and be followed immediately by the end of line.

3.2 Comments - optional - multiple

Comments are introduced by `//` (double slash) and terminated by the end of line. The double slashes **must** be the first two characters on the line; you cannot have a comment following a descriptor.

Comments can appear anywhere after the signature, in the file header or in any volume description.

3.3 Volume count - optional - singular

```
VolumeCount int
```

This descriptor specifies the number of volumes stored in the file. If the `int` is 0 or this descriptor is not present, an arbitrary number of volumes are stored in the file. The last volume is followed by the end of file.

3.4 Title - optional - multiple

```
Title rest-of-line
```

This descriptor can be used to hold the descriptions of the overall data, such as the origin of the data, date it was acquired/generated etc. This descriptor ends at a new line. Multiple title descriptors can be used in the file header:

```
Title CT scan at XY hospital on Jan-01-2000
Title 120kV, 300mA, 40DFOV
```

3.5 Copyright notice - optional - multiple

```
Copyright rest-of-line
```

This descriptor is used to place copyright notice. Similar to the title descriptor, multiple copyright notice descriptors can be present in the file header.

3.6 Attribute - optional - multiple

Attribute word rest-of-line

This descriptor is available for application specific purposes, for ASCII data related to the entire file. All `vox1999a` file readers should store all attribute word/string pairs for application retrieval.

3.7 Data - optional - multiple

```
Data word int
```

This descriptor is available for application specific purposes, for binary data related to the entire file. The integer argument represents the size in bytes of the data, which must appear after the end of header marker. All `vox1999a` file readers must process all Data descriptors to allow proper computation of file offsets for subsequent file sections. All `vox1999a` file readers should store all data word/size/binary-data triplets for application retrieval.

3.8 End of header - required - singular

```
##\f
```

This sequence signals the end of the file header. It must appear on a single line with no leading spaces, and be followed immediately by the end of line. “`\f`” denotes a form feed (ASCII value 0x0c). Application specific binary data blocks follow immediately after the end of this line, in the order that the `Data` descriptors appear in the file header, and with no padding.

4.0 Volume Description

The volume description section starts with a start-of-descriptor marker and terminates with an end-of-descriptor marker. The volume description section is used to describe the characteristics of the volume and the voxels. Singular and multiple refer to a single volume description, not to the entire `vox1999a` file.

4.1 Start of descriptor - required - singular

```
##
```

This sequence signals the start of the volume description. It must appear on a single line with no leading spaces, and be followed immediately by the end of line. It must follow immediately after the end-of-header marker if there is no application specific binary data, or follow immediately after that data if it is present.

4.2 Volume size - required - singular

```
VolumeSize int int int
```

This descriptor describes the size of the volume. The three arguments are the number of voxels in the x, y and z dimensions of the volume. Voxels are stored in a slice-by-slice fashion along the z-axis, with the index x running the fastest.

4.3 Voxel size - required - singular

```
VoxelSize int
```

This descriptor specifies the size of voxel data stored in the volume data section immediately following the descriptor block. The value is the number of bits per voxel. Valid values are 1, 8, 16, 32, and 64. All `vox1999a` file readers must be able to process all these sizes sufficiently to allow proper computation of file offsets for subsequent file sections. A `vox1999a` file reader is not required to fully support all sizes, particularly sizes 1 and 64.

4.4 Endian-ness - required - singular

```
Endian word
```

This descriptor specifies the endianess of the voxel data. The argument must be either `L` or `B` to indicate little-endian and big-endian modes, respectively. All `vox1999a` file readers must be able to convert the voxel data into the endian mode supported by the system on which the reader is being run.

4.5 Volume scale - optional - singular

```
VolumeScale float float float
```

This descriptor describes the scale of the three axis. The three floating point numbers represent the voxel spacing between adjacent voxels in the each direction.

This descriptor is for calibration information associated with the volume data and is not intended to be used in rendering, for which the model matrix should be used.

4.6 Volume position - optional - singular

```
VolumePosition float float float
```

This descriptor describes the position of the volume. The three floating point numbers, are the position of the voxel at index (0,0,0), relative to some calibration point or application specified origin (the landmark for a CT scan for example). Like the `VolumeScale` descriptor, this descriptor is not intended to be used for rendering. To position the volume in a graphical “World Space”, the model matrix should be used.

Given the scale of the axes and the position of the volume, the location of a voxel at (`x,y,z`) relative to the calibration point/application specific origin is

```
   Location = (x*Sx + Px, y*Sy + Py, z*Sz + Pz)
```

where `Sx`, `Sy` and `Sz` are the arguments to `VolumeScale` and `Px`, `Py` and `Pz` are the arguments to `VolumePosition`.

4.7 Voxel Field - Field 0 required, others optional - multiple

```
Field int (
        Position int
        Size int
        Name word
        [Format word]
        [Offset float]
        [Scale float]
        [Description quoted-string]
        )
```

The first line of this descriptor must include `Field` and the field number (the first argument). The name-value pairs (“field specifiers”) inside the parentheses can appear in any order, and the end of line indicator can appear anywhere that blank space is acceptable. The close parenthesis must be followed by the end of line indicator, possibly after some blank space. None of the field specifiers may appear more than once in a field descriptor.

Field specifiers have the following characteristics:

Name	**Required?**	**Argument type**	**Default Value**	**Description**
`Position`	yes	`int`	N/A	bit position in the voxel; 0 is the least significant bit
`Size`	yes	`int`	N/A	number of bits in the field
`Name`	yes	`word`	N/A	single word name for this field
`Format`	no	`word`	u	field data format; may be ‘u’ or ‘f’
`Offset`	no	`float`	0.0	offset to add to the field to get an application specific value; this does not affect the voxel data read from the file and presented to the application
`Scale`	no	`float`	1.0	scale factor by which to multiply the field to get an application specific value; this does not affect the voxel data read from the file and presented to the application
`Description`	no	`quoted-string`	none	textual description of the field; application specific

For example, for a 32 bit RGBA volume with red at the least significant byte and alpha the most significant byte, the following can be specified to describe the voxels in the volume:

```
Field 0 (Position 0  Size 8 Name Red)
Field 1 (Position 8  Size 8 Name Green)
Field 2 (Position 16 Size 8 Name Blue)
Field 3 (Position 24 Size 8 Name Alpha)
```

In general, this file format does not attach any meaning to the voxel fields, which can be anything (gradients, distance maps, segmentation information). Only Field 0 must be specified. For example, if a 12 bit CT scan data are stored in the upper 12 bits of a 16 bit quantity, the following field descriptor is sufficient:

```
Field 0 (Position 4 Size 12 Name CT_scan)
```

The `Format` specifier determines how the field should be interpreted. The valid values are `u` and `f`, with the default being `u`. If the format is `u`, then the field represents an unsigned integer value, ranging from 0 to (2^{Size}-1). The format `f` indicates IEEE floating point; this requires that the field size be 32. All `vox1999a` file readers must support format `u`; format `f` is not required to be supported.

Two optional field specifiers (`Offset` and `Scale`) are available to be used to convert the field value to the physical value it represents. The offset and scale should be used as follows:

```
    PhysicalValue = Offset + Scale * fieldValue
```

`Offset` and `Scale` are assumed to be 0.0 and 1.0 if they are not present in the field descriptor. These specifiers must not affect the value of the field a `vox1999a` file reader presents to the application; they are intended solely for application usage.

For example, whereas the values for a 12 bit field can range from 0 to 4095, the CT numbers they represents are actually from -1024 to about 3071. With `Offset`, the field can be specified as follows:

```
Field 0 (Position 4 Size 12 Name CT_Data Offset -1024)
```

This allows the application to adjust the voxel field values to the proper physical values for any purpose it might have; presentation to the user, for example.

All `vox1999a` file readers must accept all the field specifiers described; `vox1999a` file readers should be able to present the field specifiers to the application.

4.8 Model Matrix - optional - singular

```
ModelMatrix (
            f11 f21 f31 f41
            f12 f22 f32 f42
            f13 f23 f33 f43
            f14 f24 f34 f44 )
```

This descriptor specifies a 4x4 model matrix. If this descriptor is not present, it is assumed that the model matrix is the identity matrix. The model matrix is arranged in column major (the order used in OpenGL and VLI). The matrix values may be separated by arbitrary white space (blanks, tabs and end of line indicators) or by a single comma with optional white space before and after it. All `vox1999a` file readers must be able to present the model matrix to the application.

4.9 Attribute - optional - multiple

Attribute word rest-of-line

This descriptor is available for application specific purposes, for ASCII data related to this volume. All `vox1999a` file readers should store all attribute word/string pairs for application retrieval.

4.10 Additional data - optional - multiple

```
Data word int
```

This descriptor specifies additional data that is to follow the voxel data. `word` is an application specified name for the data and `n` is the total size, in bytes, of the data. More than one data descriptor may appear in the descriptor, and the total size of the additional data following the voxel data is the sum of each individual data size. All `vox1999a` file readers must be able to skip this data so the presence of it does not affect multiple volume reading. All vox1999a file readers should store all data word/size/binary-data triplets for application retrieval.

4.11 Comments - optional - multiple

Comments are introduced with `//` (two slashes) and end at the end of line. Comments can appear anywhere in the descriptor block. The double slashes **must** be the first two characters on the line; you cannot have a comment following a descriptor.

4.12 Title - optional - multiple

```
Title rest-of-line
```

This descriptor can be used to hold a description of the volume, in addition to the `Title` descriptors in the file header. Multiple `Title` descriptors may be present in the volume description.

4.13 Copyright - optional - multiple

```
Copyright rest-of-line
```

Additional copyright notice for the volume data. Multiple copyright notices can appear in the descriptor.

4.14 End of descriptor - required - singular

```
##\f
```

This sequence signals the end of the volume description. It must appear on a single line with no leading spaces, and be followed immediately by the end of line. “`\f`” denotes a form feed (ASCII value 0x0c).

4.15 Examples

The simplest file header and data descriptor for a volume of single field voxels:

```
Vox1999a
##\f
##
VolumeSize 128 128 128
VoxelSize 16
Endian B
Field 0  (Position 0 Size 16 Name Test_Data)
##\f
```

An example of multiple field voxels:

```
Vox1999a
##\f
##
VolumeSize 256 256 300
VoxelSize 32
Endian B
Field 0 (Position 0  Size 16 Name segment)
Field 1 (Position 16 Size 16 Name MRIData)
// a volume consisting of raw MRI data (16bit) and 16 bits of
// segmentation information
##\f
```

5.0 Volume Data

5.1 Voxel Data

Voxel data are always packed with no padding, except for at the end of the volume where the data stream is padded so it ends on a byte boundary. With the information in the data descriptor block, the total size, in bytes, of the volume can be computed as follows:

```
   TotalBytes = Floor( (Nx*Ny*Nz*VoxelSize + 7)/8 )
```

5.2 Additional Data

Additional data, if any, follows immediately after the voxel data. The total size of the additional data is the sum of the sizes of all `Data` descriptors in the volume description.

5.3 End of Data

The end of the volume data section is determined solely by the voxel data size and the sizes of all `Data` descriptors in the volume description. The descriptor for the next volume (if any) must start with a new Start-of-descriptor marker; possibly after some extraneous and uninterpreted characters.

6.0 Appendix

The following is a list of recognized names:

Name	Value(s)	Interpretation
`vox1999a`		signature
`Title`	`rest-of-line`	
`Copyright`	`rest-of-line`	
`//`	`rest-of-line`	comments
`##`		Start of Volume Description Marker
`VolumeCount`	`int`	
`VolumeSize`	`int int int`	
`VolumeScale`	`float float float`	
`VolumePosition`	`float float float`	
`Endian`	`L or B`	
`Attribute`	`word rest-of-line`	
`Data`	`word int`	
`Field n`	`( Position int Size int Name word [Format word] [Offset float] [Scale float] [Description quoted-string] )`	
`ModelMatrix`	`( float float float float float float float float float float float float float float float float )`	
`##\f`		End of Header/Volume Description Marker

7.0 Revision History

04/18/1998 (TCZ) Initial draft and basic structure definition.

02/05/1999 (TCZ) New draft based on software teams feedback on an early version of the proposal dated 04/18/1998.

02/18/1999 (TCZ) Based on software team's review (02/17/1999), removed trailer section, made ModelMatrix 4x4, added title and copyright to volume descriptor, added start-of-descriptor marker.

3/1/1999 (AFV) Cleaned up some small problems with the text, for example, made header text start with a capital letter. Changed explicit "\n" and "newline" text to "end of line". Changed "ASCII" to "ISO Latin-1". Reorganized list of recognized names. Added space between “Field” and field number. Changed “name-value pair” to “descriptor”. Changed “Volume Descriptor” to “Volume Description”. Changed all ‘code’ text to Character Tag “code”, and changed it all to size 12.

6/17/1999 (AFV) Changed ISO Latin-1 back to ASCII. Added Attribute to file header and volume description. Changed text to tables for data types and field specifiers. Coined ‘field specifiers’ to refer to the name-value pairs in a Field descriptor. Clarified that comments cannot appear after a descriptor. Expressly allowed new lines inside parentheses in Field and ModelMatrix descriptors. Added Data descriptor to file header, which now requires and end-of-header marker (##\f). Added lots of MUSTs and SHOULDs to describe required, suggested and non-required support. Did not remove field number from the Field descriptor.

7/15/1999 (AFV) Allow for a `word` to be either a sequence of non-blank characters or a `quoted-string`. Permit the open parenthesis of Field and ModelMatrix descriptors to be on a subsequent line, rather than requiring it to be on the first line. Changed the ModelMatrix syntax to use spaces rather than commas; the text did and does allow either blank space or a comma to separate values. Changed the meta-names `n` and `f` to `int` and `float`. Dropped the meta-name `c` in favor of `word`. (It was used in the Endian descriptor and in the Format field specifier.) Changed `string` to `rest-of-line`.