EIAS Version 9.0
FACT file format 2.0 Specification

Table of Contents 1

2 - Introduction 4
3 - General Information 4
- 3.1 - File suffixes and types
- 3.2 - Endian data
- 3.3 - Byte Alignment
- 3.4 - IFF and FORM
- 3.5 - Repeated structures

4 - Top Level structure 6
- 4.1 - File Header

5 - Group data form block (GRUP) 8
- 5.1 - GHDR- Group header form block
- 5.1.1 - Group information block (GINF)
- 5.1.2. - Group hierarchy and linkage information block (GLNK)
- 5.1.3 - Group starting position block (GSTR)
- 5.1.4 - Group motion blur block (GBLR)
- 5.1.5 - Group outline block (GOUT)
- 5.1.6 - Group shading attributes block (GATR)
- 5.1.7 - Texture map information block (TMAP)
- 5.1.8 - Texture map user interface information block (TMEI)
- 5.1.9 - Texture map shading attributes block (TATR)
- 5.1.10 - Secondary texture map information block (BMAP) (obsolete)
- 5.1.11 - Secondary texture map user interface information block (BMEI)
- 5.1.12 - Secondary texture map shading attributes block (BATR) (obsolete)
- 5.1.13 - Reflectance map information block (RMAP)
- 5.1.14 - Reflectance map user interface information block (RMEI)
- 5.2 - Coordinate list block (CORD)
- 5.3 - DCoordinate list block (DCOR)
- 5.4 - Color vertex list block (CVRT)
- 5.5 - Normal vertex list block (NVRT)
- 5.6 - Texture vertex list block (TVRT)
- 5.7 - Bump alignment vertex list block (BVRT)
- 5.8 - Element list block (ELEM)

6 - Light data form block (LITE) 24
- 6.1 - Light Header form block (LHDR)
- 6.1.1 Light info header (LINF)
- 6.1.2 - Light Attributes block (LATR)
- 6.1.3 - Light Linkage Attributes block (LLNK)
- 6.1.4 - Light Linkage Constraints (CNST) (optional)
- 6.1.5 - Light Shadow Info (LSHD) (optional)
- 6.1.6 - Light Glow Info (LGLW)
- 6.1.7 - Light Fog Info (LFOG) (optional)

Electric Image Animation System

2 - Introduction

The purpose of FACT is to provide a single file format for the storage and exchange of 3D models for use in rendering and animation. The format was designed to be both forward and backward compatible, memory efficient, capable of storing most forms of 3D data, and relatively simple to support for both import and export. The FACT format is a native format of the Electric Image Animation System, and is used by other programs such as the Electric Image Modeler and Amorphium from Play. FACT is a contraction of facet.

This document describes version 2.0 of the FACT file format (.fac extension).
The FACT file format was created by Mark Granger and Markus Houy. This document was first created by Michael Uhlik, and plagiarizes a document by Mark Granger.

For more information about the FACT format or Electric Image contact:

tom@eias3d.com
igors@eias3d.com

3 - General Information

3.1 - File suffixes and types

On the Apple Macintosh operating system, a FACT file has a type of 'FACT' and a creator, which is 'EIAM'. On the Microsoft Windows NT operating system, a FACT file ends in a suffix '.fact'.

3.2 - Endian data

All multibyte values in the FACT file format are written as big endian values, so that the most significant byte is written first. Platforms such as Intel processor based machines must reverse the bytes of multibyte values when reading and writing. This is true for both integer and IEEE float values (but not single character or string values, of course).

3.3 - Byte Alignment

In order to facilitate reading on machines with optimized access to aligned words, all FACT file format IFF blocks have an even size.

3.4 - IFF and FORM

The FACT file format follows the conventions of the Interchange File Format, or IFF. IFF files are composed of serialized blocks of data. Each block has a type, a size, and usually some number of bytes of data. The type is the first four bytes of the block, and is usually interpreted as a stream of four ASCII characters. The size directly follows the type, and is a four byte integer. After the size follow "size" bytes of data. For example, a block containing a text string could be constructed like this:

bytes	data	interpretation
4	'TEXT'	4 bytes of data interpreted as ASCII characters, for the type of the IFF block
4	0x0000000c	the size of the data in the block - the number 12, written as a 32 bit unsigned integer
12	'Hello World<CR>'	12 bytes of data. In general, any kind of data could be here; this particular block is designed to be ASCII text.

In the FACT file format, IFF blocks which contain other IFF blocks have a type prefix 'FORM'. 'FORM' blocks have the type 'FORM', a size, and then another four-byte type, followed by one or more other blocks. For example, a block containing two text strings could be constructed like this:

bytes	data	interpretation
4	'FORM'	beginning of a FORM block
4	0x00000020	The size of the data in the block - the number 32, written as a 32 bit unsigned integer
4	'TEST'	the form type identifier
4	'TEXT'	the beginning of a TEXT block
4	0x00000006	the size of this TEXT block
6	'Hello\0'	six bytes of data for the 1st TEXT block
4	'TEXT'	the beginning of another TEXT block
4	0x00000006	the size of this TEXT block
6	'World'	another 6 bytes of string data

When reading IFF blocks, you should pay attention to the size. If a block has a shorter size than you expect, you must stop reading at the end of the block and supply default values for the information you are missing or otherwise handle the situation. If a block has a longer size than you expect, you must skip to the end of it before starting to read the next block. This allows some inter-version compatibility.

3.5 - Repeated structures

Some complex (as in containing more than one component) data types are written in several IFF blocks of the FACT file format. To increase readability, these types will be defined here and the insertion of their name in an IFF block will be understood to mean that the data listed here appears in the stream.

color - 4 unsigned char (8 byte) color values for alpha, red, green, and blue. Each value can range from 0 to 255.

bytes	data
1	alpha value
1	red value
1	green value
1	blue value

vector - a triplet of 8-byte IEEE double precision floats

bytes	data
8	x value of coordinate
8	y value of coordinate
8	z value of coordinate

matrix - a 4 dimensional matrix of 8-byte IEEE double precision floats

```
bytes          data
--------------------
8              matrix element 00     (0th row, 0th column)
8              matrix element 01     (0th row, 1st column)
8              matrix element 02     (0th row, 2nd column)
8              matrix element 03     (0th row, 3rd column)

8              matrix element 10     (1st row, 0th column)
8              matrix element 11     (1st row, 1st column)
8              matrix element 12     (1st row, 2nd column)
8              matrix element 13     (1st row, 3rd column)

8              matrix element 20     (2nd row, 0th column)
8              matrix element 21     (2nd row, 1st column)
8              matrix element 22     (2nd row, 2nd column)
8              matrix element 23     (2nd row, 3rd column)

8              matrix element 30     (3rd row, 0th column)
8              matrix element 31     (3rd row, 1st column)
8              matrix element 32     (3rd row, 2nd column)
8              matrix element 33     (3rd row, 3rd column)
```

Xmatrix - a 4 dimensional matrix of 12-byte Extended floats

```
bytes          data
--------------------
12             matrix element 00     (0th row, 0th column)
12             matrix element 01     (0th row, 1st column)
12             matrix element 02     (0th row, 2nd column)
12             matrix element 03     (0th row, 3rd column)

12             matrix element 10     (1st row, 0th column)
12             matrix element 11     (1st row, 1st column)
12             matrix element 12     (1st row, 2nd column)
12             matrix element 13     (1st row, 3rd column)

12             matrix element 20     (2nd row, 0th column)
12             matrix element 21     (2nd row, 1st column)
12             matrix element 22     (2nd row, 2nd column)
12             matrix element 23     (2nd row, 3rd column)

12             matrix element 30     (3rd row, 0th column)
12             matrix element 31     (3rd row, 1st column)
12             matrix element 32     (3rd row, 2nd column)
12             matrix element 33     (3rd row, 3rd column)
```

4 - Top Level structure

Each entire FACT file is contained in one FORM block with form type ‘3DFL’ (3D data FiLe). A 3DFL form block contains a ‘FHDR’ (File Header) form block followed by one
or more ‘GRUP’ (GRoUP) form blocks or ‘LITE’ (light) form blocks. These blocks represent the 3D objects and lights in the file. As it appears on disk:

3DFL form block

```
bytes          data
--------------------
4              'FORM'
4              integer - size of 3DFL data
4              '3DFL'

4              'FORM'
4              integer - size of FHDR data (x+4)
4              'FHDR'
x              data in FHDR block; see below

4              'FORM'
4              integer - size of 1st group data (y+4)
4              'GRUP'
y              1st group data

4              'FORM'
4              integer - size of 2nd group data (z+4)
4              'GRUP'
z              2nd group data

4              'FORM',
4              integer - size of 1st light data (a+4)
4              'LITE'
a              1st light data
```

And so on for each GRUP or LITE form block in the file.

4.1 - File Header

The file header block contains only one block, a 'FINF' (File INFormation) block. The FINF block contains information relating to the whole file.

FHDR (file header) form block

```
bytes          data
--------------------
4              'FORM'
4              integer - size of FHDR block (x+4)
4              'FHDR'
x              FINF block
```

FINF block

```
bytes          data
--------------------
4              'FINF'
4              integer - size of FINF block (52 in this version)
4              integer - Total # of coordinates in all groups in the file
4              integer - Total # of polygons in all groups the file
4              integer - Total # of groups in the file
4              float - minimum x value of coordinates (minimum x entent)
4              float - minimum y value of coordinates (minimum y entent)
```

```
4               float - minimum z value of coordinates (minimum z entent)
4               float - maximum x value of coordinates (maximum x entent)
4               float - maximum y value of coordinates (maximum y entent)
4               float - maximum z value of coordinates (maximum z entent)
4               flags - file flags; unused, set to zero
4               float - x anchor point for this file
4               float - y anchor point for this file
4               float - z anchor point for this file
```

5 - Group data form block (GRUP)

A group is usually visualized as a 3D model. It may contain lists of vertices, polygons, material information, and other information. There may be many groups in a 3DFL form block.
Many IFF blocks may be present in a GRUP form block. These are:

Group header form block (GHDR)
Coordinate list block (CORD)
DCoordinate list block (DCOR)
Color vertex list block (CVRT)
Normal vertex list block (NVRT)
Texture vertex list block (TVRT)
Bump alignment vertex list block (BVRT)
Element list block (ELEM)

This is the recommended order. None of these blocks are required unless they are specifically referenced by some other block. A CORD or DCOR block is required, for example, if it is referenced by an ELEM block. The same rule applies for the order in which the blocks must appear. For instance, an ELEM block must be preceded by a CORD or DCOR block. Each block type should appear no more than once in a GRUP form block.

The serial format for a GRUP block is:

```
bytes           data
--------------------
4               'FORM'
4               size of the GRUP block data (x+y+....)
4               'GRUP'
x               GRUP sub-block
y               GRUP sub-block,
```

And so on. The possible GRUP sub-blocks are listed above and explained below.

5.1 - GHDR- Group header form block

The Group Header form block contains information that applies to the group as a whole - for example, material information. It may contain any combination of the following blocks:

Group information block (GINF)
Group hierarchy and linkage information block (GLNK)
Group starting position block (GSTR)
Group motion blur motion block (GBLR)
Group shading attributes block (GATR)
Texture map information block (TMAP)

Texture map user interface information block (TMEI)
Texture map shading attributes block (TATR)
Secondary texture map information block (BMAP) (obsolete)
Secondary texture map user interface information block (BMEI)
Secondary texture map shading attributes block (obsolete) (BATR)
Reflectance map information block (RMAP)
Reflectance map user interface information block (RMEI)

This is the recommended order for these blocks. As in the 'GRUP' (group data) form block, none of these blocks is specifically required. Also, there may be dependancies between these blocks and their order should be adjusted accordingly when writing, so that the dependant block is written after the block it depends on.

The serial format for a GHDR block is:

```
bytes          data
--------------------
4              'FORM'
4              integer - size of the GHDR block data (x+y+....)
4              'GHDR'
x              GHDR sub-block
y              GHDR sub-block,
```

And so on. The possible GHDR sub-blocks are listed above and explained below.

5.1.1 - Group information block (GINF)

The Group Information block contains information common to the group as a whole, such as the name of the group and the transformation of its local coordinate matrix.

```
bytes          data
--------------------
4              'GINF'
4              integer - size of the GINF block (  here)
4              integer - total number of coordinates in the GRUP block
4              integer - total number of polygons in the GRUP block
4              integer - reserved, set to zero when writing
4              float - minimum x value of group extents (minimum x entent)
4              float - minimum y value of group extents (minimum y entent)
4              float - minimum z value of group extents (minimum z entent)
4              float - maximum x value of group extents (maximum x entent)
4              float - maximum y value of group extents (maximum y entent)
4              float - maximum z value of group extents (maximum z entent)
4              flags - see "Group Information Flags" below
32             string - group name.  32 byte null-terminated string.
4              integer - date of last group modification
2              integer - group ID number (short int)
192            xmatrix - Coordinate absolute (flat) transformation matrix
192            xmatrix - Normal absolute (flat) transformation matrix
192            xmatrix - Coordinate relative (hierarchical) transformation matrix
192            xmatrix - Normal relative (hierarchical) transformation matrix
4              integer - number of child groups which follow, not including their child groups
4              integer - number of child groups which follow, including their child groups
4              integer - The number of child groups to cycle if cycle child groups bit is set
               (0 to cycle all)
```

Group Information Flags:

name	bit	explanation
Invisible Flag	31	Invisible group
Break Facet Flag	30	Break nonflat polygons
Correct Normal Flag	29	Correct normals (either all clockwise or all counter-clockwise vertices)
Detail First Flag	28	Detail polygons first in list (reverse detail priority)
Cycle Flag	27	Cycle child groups
Flatten Complex Flag	26	Flatten complex polygons
Outward Normal Flag	25	The group normals all point outwards
Solid Flag	24	Reserved
Texture Vertices Flag	23	The group contains texture vertex records
Normal Vertices Flag	22	The group contains normal vertex records
Color Vertices Flag	21	The group contains color vertex records
Bump Vertices Flag	20	The group contains bump vertex records
Fog Flag	19	Disable camera fog for the group when this bit is set
Cull Back Facets Flag	18	Cull back facing facets
Reverse Back Facets Flag	17	Reverse the surface normal before culling back facing facets
Color Illumination Flag	16	Use the color vertices as illumination values
Shade Lines Flag	15	Lightsource shade point and line facets
Sample Level 2	2	The group's sampling level is stored in the lowest three bits of the flags:
Sample Level 1	1	0 = 1x1; 1 = 2x2; 2 = 4x4; 3 = 8x8; 4 = 16x16; 5 = 32x32; 6 = 64x64
Sample Level 0	0	7 = automatic (the same as the frame's maximum sampling level)

5.1.2. - Group hierarchy and linkage information block (GLNK)

This block contains information about the transformation of the group's local coordinate system. Its information may be used in addition to the hierarchical data structure but is not required. The order of transformations applied by EIAS is given here:

1. Offset to link point (-x,-y,-z) using localOffset - This is the rotation point around which the group rotates.
2. Change to the local coordinate system using localRotation - Apply inverse rotation (-z,-x,-y).
3. Do the geometry transformation using key framed motion in animation program.
4. Change back to the global coordinate system using localRotation - Apply rotation (y,x,z).
5. Undo offset to link point (x,y,z) using localOffset - Put group back to its original point.
6. Change to link coordinate system using parentRotation (-z,-x,-y) - This is the initial rotation of the link.
7. Offset (x,y,z) to the position on the parent using parentOffset - This is the point where the link is attached to the parent.

bytes	data
4	'GLNK'
4	size of group hierarchy and linkage information block (350 here)
2	integer - linking type (see Hierarchical Linking Types below) (short int)
24	vector - position minimum limit
24	vector - position maximum limit
24	vector - rotation minimum limit
24	vector - rotation maximum limit
24	vector - local offset - local coordinate system's offset to the world

```
24              vector - parent link location - starting offset
24              vector - parent link rotation - starting rotation
24              vector - local rotation - local coordinate system's rotation to the world
192             Xmatrix - Pre-Rotation matrix: this will supersede Parent Rotation if defined.
                Use Parent Rotation only if Pre-Rotation is all zeros to define Pre-Rotation.
4               integer - limit flag - used to define limits of hierarchical motion.  Bit 31 =
                zero if old style hierarchy, bit 31 = 1 if new style hierarchy.
24              vector - pre-scale
```

Hierarchical Linking Types for GLNK block

The linking type field may contain one of two types of values, depending on whether bit 15 of the field is 1 or 0. If it is 0, the field contains a integer value with the following interpretations:

```
value   name                  interpretation
-------------------------------------------
0       free link             x,y,z rotation and x,y,z translations
1       ball planar link      x,y,z rotation and x,z translations only
2       cylinder planar link  y,z rotation and x,z translations only
3       planar link           y rotation and x,z translations only
4       socket link           x,y,z rotations only
5       cylinder link         z rotations and z translations only
6       universal link        y,z rotations only
7       slide link            z translation only
8       pin link              z rotation only
9       lock link             No transformations
```

If bit 15 is set to 1, the field is interpreted as a bit field with the following significant bits:

```
bit #   name                  interpretation
-------------------------------------------
15      custom link bit       set to 1 if custom movement limitation are used
0       lock x pos            set to 1 if x translation is locked
1       lock y pos            set to 1 if y translation is locked
2       lock z pos            set to 1 if z translation is locked
3       lock x pos            set to 1 if x rotation is locked
4       lock y pos            set to 1 if y rotation is locked
5       lock z pos            set to 1 if z rotation is locked
```

5.1.3 - Group starting position block (GSTR)

This block contains the starting transformation for groups - this may be used to setup groups in space without calculating new points.

```
bytes           data
--------------------
24              vector - offset: starting offset
24              vector - center: starting center of rotation
24              vector - starting rotation: starting rotation/reference
24              vector - scale: starting scale
```

5.1.4 - Group motion blur block (GBLR)

The motion blur block contains information used in rendering a motion blur effect for the group.

```
bytes          data
--------------------
4              flags - see Group Motion Blur Flags below
4              float - blur intensity
4              integer - blur layer
192            Xmatrix - blur matrix - coordinate absolute matrix from the previous frame
```

Group Motion Blur Flags

```
bit #   name                  interpretation
-------------------------------------------
31      Enable Blur           true to enable motion blur for a group
30      Blur Intensity        true to calculate the intensity for point/line motion blur
28      Facet Blur            true for facet motion blur, false for point-line motion blur
```

5.1.5 - Group outline block (GOUT)

The Group outline block contains information about rendering an outline around a group.

```
bytes          data
--------------------
4              'GOUT'
4              size of GOUT block (28 here)
4              flags - see Group Outline Flags below
4              color - polygon outline color
4              float - polygon outline thickness
4              color - edge outline color
4              float - edge outline thickness
4              color - silhouette outline color
4              float - silhouette outline thickness
```

Group Outline Flags

```
bit #   name                  interpretation
-------------------------------------------
31      outline polygon       true to enable polygon outlines
30      outline edge          true to enable edge outlines
29      outline silhouette    true to enable silhouette outlines
```

5.1.6 - Group shading attributes block (GATR)

The group shading attributes block contains information about the material used for a group - for instance, data about its color, transparency, and luminance.

```
bytes          data
--------------------
4              'GATR'
4              size of GATR block ( here)
2              integer - shade flags - see Group Shading Flags below
4              color - reference - surface color; overrides element color when override
```

```
                color flag is set
4               color - ambient - ambient reflection color
4               color - specular - specular reflection color
4               float - spread - specular highlight spread exponent
4               color - transparency - transparency components, with alpha interpreted as
                the gloss factor (255=no gloss; 0=100% gloss)
4               float - refraction - refraction index
4               color - reflection - reflection color factor for material.  Alpha is not used.
4               color - luminance - self illumination components, with alpha interpreted as
                the shade factor
4               float - line weight - the thickness of points and/or lines in the group
4               float - edge transparency dropoff - the degree to which the transparency
                changes from the center to the edge (0.0=none)
4               color - edge transparency - the transparency color at the edge of an object
4               color - diffuse - diffuse reflection color
4               float - terminator - terminator dropoff factor
4               float - highlight - highlight dropoff factor
```

* the following values were added for EIAS v2.8:

```
4               float - edge diffuse value - diffuse value at the edge of an object
4               float - edge specular value - specular value at the edge of an object
4               float - edge reflection value - reflection value at the edge of an object
4               float - edge ambient value - ambient value at the edge of an object
4               float - edge luminance value - luminance value at the edge of an object
2               integer - shade flags 2 - see Group Shading Flags 2, below
4               color - transmission - diffuse shading transmission color.  Alpha is
                not used.
4               float - edge transmission value - the diffuse shading transmission value at
                the edge of an object
4               color - glow - glow color, alpha is the shaded glow factor
4               float - edge diffuse dropoff - the degree to which the diffuse changes
                from the center to the edge (0.0 = none)
4               float - edge specular dropoff - the degree to which the specular changes
                from the center to the edge (0.0 = none)
4               float - edge reflection dropoff - the degree to which the reflection changes
                from the center to the edge (0.0 = none)
4               float - edge ambient dropoff - the degree to which the ambient changes
                from the center to the edge (0.0 = none)
4               float - edge luminance dropoff - the degree to which the luminance
                changes from the center to the edge (0.0 = none)
4               float - edge transmission dropoff - the degree to which the transmission
                changes from the center to the edge (0.0 = none)
4               float - edge transparency value - the transparency value at the edge of an object
```

Group Shading Flags

```
bit #   name                    interpretation
-------------------------------------------
15      color blend             Blend average vertex colors across facets
14      glossy                  Add white to brightly lit surfaces (obsolete, set to false)
13      smooth                  Enable smooth interpolation of facet normals for the group
12      super sample            Set the group's sampling level to 2x2 (obsolete - now set in the GINF block)
11      shade mode 3            Bits 8-11 are the shading mode (See Shading Modes 10 shade mode 2 below)
9       shade mode 1
8       shade mode 0
7       override color          Override surface color with GATR reference color
```

6	transparency	Disable filter filter transparency for the group
5	transparency alpha	Use facet and color vertex alpha channel as transparency
3	shadow mask	Set to cause the group to generate a shadow mask
2	shadow cast	Set to prevent the group from casting shadows
1	shadow receive	Set to prevent the group from receiving shadows
0	shadow precision	Set to use a high precision shadow buffer algorithm for the group

Shading Modes

value	name	interpretation
0	flat	Compute a constant shading color for each facet
1	gouraud	Interpolate shading colors across facets
2	phong	Compute a shading color for each pixel
15	wire	Only render points, lines and polygon edges

Group Shading Flags 2

bit #	name	interpretation
5	add transparency	Use additive transparency mode (vs subtractive mode)
4	new transparency	Use the new transparency shading function
3	bias transparency	Bias the transparency color with the diffuse color
2	bias specular	Bias the specular color with the diffuse color
1	bias reflection	Bias the reflection color with the diffuse color
0	shade refraction	Change the reflection and transparency attributes based on the refraction index

5.1.7 - Texture map information block (TMAP)

The texture map information block contains information about the texture map applied to a group.

bytes	data
4	'TMAP'
4	size of TMAP data (712 here)
2	flags - texture flags - see Texture Mapping Flags below
2	integer - texture type - Alpha Mode (see below)* 256 + RGB Mode (see below), or Shader Option Flags and Channel Modes (see below)
32	string - texture type name - reserved, set to null string
2	integer - mapping type - see Texture Mapping Types below
32	string - mapping type name - reserved, set to null string
192	Xmatrix - The 4 by 4 transformation matrix for the original group coordinates
32	string - the file name of the texture image file
2	integer - the top of the rectangle within the image file that will be used for texture mapping
2	integer - the left coordinate of the rectangle
2	integer - the bottom coordinate of the rectangle
2	integer - the right coordinate of the rectangle
24	vector - The offset of the mapped point within the texture
24	vector - The scale of the mapped point within the texture
4	integer - the directory reference ID of the texture map
32	string - the volume name of the texture map

```
8               float - the scale factor applied to the bump normal
192             Xmatrix - the rotational matrix for the bump
4               integer - texture frame - the frame of the associated animation file to use
4               float - texture time - the time as a real number, for custom textures
4               float - the density factor for the texture; >1 blurs, <1 sharpens
2               integer - the maximum density allowed in the map (0=disable)
24              vector - back offset - reserved, must be equal to offset
24              vector - back scale - reserved, must be equal to scale
24              vector - left offset - reserved, must be equal to offset
24              vector - left scale - reserved, must be equal to scale
24              vector - right offset - reserved, must be equal to offset
24              vector - right scale - reserved, must be equal to scale
24              vector - top offset - reserved, must be equal to offset
24              vector - top scale - reserved, must be equal to scale
24              vector - bottom offset - reserved, must be equal to offset
24              vector - bottom scale - reserved, must be equal to scale
```

Texture Mapping Flags

```
bit #   name                    interpretation
-------------------------------------------
15      textureXModBit          Set to enable multiple copies of the texture in the x direction
14      textureYModBit          Set to enable multiple copies of the texture in the y direction
13      textureNegativeZBit     Set to allow texture mapping for coordinates with negative z values
12      textureMipBit           Set to filter the map with MIP mapping
11      textureSmoothBit        Set to enable interpolation smoothing
10      textureXFlipFlopBit     Set to flip-flop multiple copies of the texture in the x direction
9       textureYFlipFlopBit     Set to flip-flop multiple copies of the texture in the y direction
8       textureCFilterBit       Set to enable the custom shading attributes filter
7       textureInvertRGBBit     Set to invert the map's RGB channels
6       textureSharpBit         Set to use linear interpolation of textures for faster sharper results
5       textureSumBit           Set to filter the texture with summation mapping (overrides MipBit
4       textureXClampBit        Set to clamp to the color of the nearest texture edge in the x direction
3       textureYClampBit        Set to clamp to the color of the nearest texture edge in the y direction
2       textureShaderModeBit    Set to use textureMode as shader option flags and channel modes (see below)
1       textureInvertAlphaBit   Set to invert the map's alpha channel
0       textureDisableBit       Set to disable the texture map
```

Alpha Modes

The alpha modes are directions for how to use the alpha channel of the texture.

```
value   name                    interpretation
-------------------------------------------
1       none                    Do not use the alpha channel
2       mix                     Mix the surface color with the ARGB texture color
3       replace                 Replace the surface color with the ARGB texture color
4       bump                    Modify the surface normal by the change in the texture alpha
5       surface                 Factor the surface color by the texture alpha
6       ambient                 Factor the ambient color by the texture alpha
7       specular                Factor the specular color by the texture alpha
8       reflection              Factor the reflection color by the texture alpha
9       transparency            Factor the transparency alpha by the texture alpha
10      luminance               Factor the luminance color by the texture alpha
11      edge transparency       Factor the edge transparency alpha by the texture alpha
12      diffuse                 Factor the diffuse color by the texture alpha
```

RGB modes

The RGB modes are directions for how to use the RGB channels of the texture.

value	name	interpretation
1	none	do not use the rgb channel
2	surface	change the surface color
3	ambient	change the ambient color
4	specular	change the specular color
5	reflection	change the reflection color
6	transparency	change the transparency color
7	luminance	change the luminance color
8	edge transparency	change the edge transparency color
9	diffuse	change the diffuse color

Shader Option Flags

The shader option flags occupy only the top 3 bits of the texture type field in the texture map information block. The bottom 8 bits contain the channel mode (as an integer).

bit #	name	interpretation
15	multiply	Multiply the selected channel by MAX(r,g,b)
14	mask	Use the alpha channel as a mask
13	replace RGB	Replace the RGB channels with the alpha channel

Channel Modes

value	name	interpretation
0	none	Do not use the shader color
1	bump	Bump value
2	displacement	(Obsolete, use Bump instead)
3	clip	Clip value
4	opacity	Opacity value
5	gloss	Gloss value
6	shade	Shade value
7	alpha	Alpha value
8	bloom	Bloom value
9	glow	Glow RGB color
10	reference	Reference RGB color
11	diffuse	Diffuse RGB color
12	ambient	Ambient RGB color
13	specular	Specular RGB color
14	reflection	Reflection RGB color
15	transparency	Transparency RGB color
16	luminance	Luminance RGB color
17	transmission	Transmission RGB color
18	edge diffuse	Edge diffuse value
19	edge ambient	Edge ambient value
20	edge specular	Edge specular value
21	edge reflection	Edge reflection value
22	edge transparency	Edge transparency value
23	edge luminance	Edge luminance value
24	edge transmission	Edge transmission value

Texture Mapping Types

value	name	interpretation
0	none	mapping disabled
1	flat	map a flat texture (same as above)
2	cylindrical	map the texture around to a cylinder
3	spherical	map the texture around to a sphere
4	cubic	choose the best side of a cube based upon the surface normal
5	planar	map to the current image pixel's coordinate
6	pyramid	map the texture with a pyramid projection

5.1.8 - Texture map user interface information block (TMEI)

The Texture map user interface information block was added to tell the interface of Electric Image Animation system how to draw the texture projection primitive. It adds additional fields for the texture image file location. EI Camera uses only the TMAP, RMAP, and BMAP blocks for rendering.

bytes	data
4	'TMEI'
4	size of TMEI data (varies depending on mapping type)
4	flags - additional texture map flags - must default to 0x80000000.
2	integer - top of full texture rectangle
2	integer - left of full texture rectangle
2	integer - bottom of full texture rectangle
2	integer - right of full texture rectangle
24	vector - Texture Origin (usually group's midpoint)
24	vector - Texture Offset (from texture origin; rotation center of group)
24	vector - Rotation
2	integer - volume index of texture
4	integer - directory ID of texture

* the rest of the block can vary depending on the mapping type selected. See below for more explanation.

For a flat mapping, the format of the rest of the block is:

8	float - Image Scale in X
8	float - Image Scale in Y
8	float - Image Scale in Z

For a cylindrical mapping:

8	float - Base Radius - default radius of enclosing sphere for group
8	float - Degree of texture wrap, default 360.0deg (full map around perimeter)
8	float - Height Scale - default 1.0 (texture height * height factor)
8	float - Real Height

For a spherical mapping:

8	float - Sphere Radius - default radius of enclosing sphere of group
8	float - Wrap Angle - Degree of longitude texture wrap: default 360deg (full map around sphere)
8	float - Band Angle - Degree of latitude texture wrap: default 180deg (full map pole to pole)
8	float - Band Angle Start - Degree of latitude start: default 180deg (start from "north" pole)

Electric Image Animation System

Some additional information about texture mapping in EI and the TMAP and TMEI blocks:

Image Rectangles

Electric Image uses the image size (in pixel) to determine the scale factors for the texture projections. The field "fullRect" contains the enclosing of the image. For example, if the image is 640 by 480 pixels, the full texture rectangle in the TMEI block is

left = 0,
top = 0,
right = 640,
bottom = 480.

The rectangle field in TMAP contains the cropping rectangle of the image. This rectangle specifies the actual rectangular region of the image to be used and is a subset of the full rectangle in TMEI. The default size of the TMAP rect is the TMEI full rectangle (use the whole image).

Specifying the texture image file location

The TMAP structure defines the name, volume name, and directory ID of the texture image file. For historical purposes, the TMEI duplicates the directory ID and adds the volume index to help find the texture image file. The texture directory ID in TMEI contains the same directory ID as the directory reference ID in TMAP. Additionally, the volume index field contains the index of the volume in which to find the texture image file. The current version of EI uses the TMEI to setup its internal file reference and, therefore, must be set.

Projection geometry

The interface draws projection primitives to help the user to align the textures to the underlying goemetry. There are three projection types which need the additional geometry information to draw the primitives: flat, cylindrical, and spherical.

Flat

This projection works like a projector throwing an image onto the geometry. The image scale fields in the TMEI block provide a way of scaling the image to fit the geometry. Each pixel of the image corresponds to one unit in floating point space. In other words, 1 pixel = 1.0. The scale factor allows us to modify that relationship to make the image appear larger or smaller on the geometry. For default scale, you may choose a side of the extent box enclosing the geometry and compute the scale by taking the extent box size and dividing it by the image size.

For example, if you have a cube with 10.0 units each side and a texture image of size 640 by 480, then the scaling factor of the texture map, if the map is projected on the front of the cube, is

```
imageScaleX = 10.0 / 640 = 0.0156;
imageScaleY = 10.0 / 480 = 0.0208;
```

The "Image Scale in Z" parameter specifies which direction the image is being projected. In the interface, there is an arrow perpenticular to the texture plane primitive. It is used to indicate which direction the projection is going to be done if the user has unselected "Negative Z" in the interface. The number in "Image Scale in Z" is not as important as its sign. However, since the arrow is drawn using the "Image Scale in Z", it is best to copy

either “Image Scale in X” or “Image Scale in Y” into “Image Scale in Z”.

If the geometry is flat (2 dimensional), care should be taken, so that the map optimally projects onto the face of the flat geometry and not on its side.

Cylinder

This projection works by enclosing the geometry with a cylinder. The field “Base Radius” defines the radius of the cylinder. It is computed so that it encloses the extent box of the geometry. This depends on which axis the cylinder is specified. The field “Real Height” defines the height of the cylinder. It is computed depending on the axis to which the cylinder is aligned. It is up the developer which axis to use to compute the “Base Radius” and “Real Height”. In the interface, the align tool allows the user to indicate which axis the cylinder best fits. The field “Height Scale” defaults to 1.0 to indicate that the complete “Real Height” is used to define the scale of the Y dimension of the texture image. The field “Wrap Angle” defaults to 360.0 which indicates that the image is wrapped completely around the geometry (360 degrees) in the X dimension.

Sphere

This projection works by enclosing the geometry with a sphere. The fields “Wrap Angle”, “Band Angle”, and “Band Angle Start” are used to specify how to map the texture image onto goemetry. The field “Sphere Radius” is used to draw the sphere primitive in the interface. The field “Sphere Radius” contains the radius of the enclosing sphere primitive and is computed from the midpoint to the furthest corner of the enclosing extent box of the goemetry. The field “Wrap Angle” defaults to 360.0 which indicates that the image is wrapped completely around the geometry (360 degrees) in the X dimension. The field “Band Angle” defaults to 180.0 which wraps the Y dimension of the image from pole to pole (180 degrees). A lower value compresses image around the equator. The field “Band Angle Start” defaults to 180.0 which starts the Y dimension wrapping at the pole.

General Transformation

All projections use the following transformations. The field “Texture Origin” in the TMEI block indicates the origin of the texture within the geometry. This is the starting point of the transformation and it is in the local coordinate space of the geometry. Usually, it is the mid point of the extent box of the geometry, but may be specified somewhere else for a better fit. The field “Texture Offset” specfies the offset of the texture from its origin. This allows the user to move the texture somewhere other than its default position. The default value is (0.0, 0.0, 0.0). The field “Rotation” allows the texture to rotated around its rotation point. The rotation point of a flat texture is the center of the flat primitive. The rotation point of a cylindrical texture is the center of the cylinder primitive. The rotation point of a spherical texture is the center of the sphere primitive. The default value is (0.0, 0.0, 0.0).

In any case, use the Electric Image texture interface to check what kind of numbers are being generated when you apply the align tool.

5.1.9 - Texture map shading attributes block (TATR)

The optional Texture map shading attributes block contains the shading attributes for the texture map. The format for the TATR block is identical to the format of the GATR block. The TATR block is only used if the texture attributes bit is set in the TMAP texture flags, the texture alpha mode is Mix, and the TATR block actually exists. The shade flags field in the GATR block is used as a set of enable flags, one for each value in the structure. For future compatibility, all unused flags will be zero.

5.1.10 - Secondary texture map information block (BMAP) (obsolete)

The secondary texture map information block is identical to the group attribute block (GATR), except for the IFF type. It is now obsolete.

5.1.11 - Secondary texture map user interface information block (BMEI)

The secondary texture map user interface information block is identical to the texture map user interface information block (TMEI), except for the IFF type.

5.1.12 - Secondary texture map shading attributes block (obsolete) (BATR)

The secondary texture map shading attributes block is identical to the group attribute block (GATR), except for the IFF type. It is now obsolete.

5.1.13 - Reflectance map information block (RMAP)

The reflectance map information block is identical to the texture map information block, except for the IFF type.

5.1.14 - Reflectance map user interface information block (RMEI)

The reflectance map user interface information block is identical to the texture map user interface information block (TMEI), except for the IFF type.

5.2 - Coordinate list block (CORD)

A Coordinate list block can be used to store vertex positions for polygons in a group. It is simply a list of single precision floating point ordered triples, one for each point, specifying x, y, and z.

```
bytes          data
--------------------
4              'CORD'
4              size of CORD data - equal to number of positions x 12.
4              float - x value of coordinate 1
4              float - y of coordinate 1
4              float - z of coordinate 1
4              float - x of coordinate 2
4              float - y of coordinate 2
4              float - z of coordinate 2
```
etc...

5.3 - DCoordinate list block (DCOR)

A DCoordinate list block (double precision coordinate list block) is the double precision version of the Coordinate list block. It is a list of double precision floating point ordered triples, one for each point, specifying x, y, and z.

```
bytes          data
--------------------
4              'DCOR'
4              size of DCOR data - equal to number of positions x 24.
8              float - x value of coordinate 1
8              float - y of coordinate 1
8              float - z of coordinate 1
8              float - x of coordinate 2
8              float - y of coordinate 2
8              float - z of coordinate 2
```
etc...

5.4 - Color vertex list block (CVRT)

The color vertex list block stores vertex colors for the polygons in a group. There should be the same number of color vertices as coordinates (or DCoordinates). The format is a list of 'color' structures.

```
bytes          data
--------------------
4              'CVRT'
4              size of CVRT data - equal to number of vertices x 4.
4              color - color vertex 1
4              color - color vertex 2
```
etc...

5.5 - Normal vertex list block (NVRT)

The normal vertex list block stores vertex normals for the polygons in a group. There should be the same number of vertex normals as there are coordinates (or DCoordinates). The format is a list of single precision floating point ordered triples, one for each normal, specifying x, y, and z.

```
bytes          data
--------------------
4              'NVRT'
4              size of NVRT data - equal to number of vertices x 12.
4              float - x value of normal vertex 1
4              float - y of normal vertex 1
4              float - z of normal vertex 1
4              float - x of normal vertex 2
4              float - y of normal vertex 2
4              float - z of normal vertex 2
```
etc...

5.6 - Texture vertex list block (TVRT)

The texture vertex list block stores texture positions for each vertex in a group. There should be the same number of texture coordinates as there are coordinates (or DCoordinates). Its format is similar to the other coordinate list structures. It uses single precision floating point ordered triples.

```
bytes          data
--------------------
4              'TVRT'
4              size of TVRT data - equal to number of vertices x 12.
4              float - x value of texture vertex 1
4              float - y of texture vertex 1
4              float - z of texture vertex 1
4              float - x of texture vertex 2
4              float - y of texture vertex 2
4              float - z of texture vertex 2
```
etc...

5.7 - Bump alignment vertex list block (BVRT)

The bump alignment vertex list block contains bump alignment vectors for each vertex. There should be the same number of bump alignment vectors as there are coordinates (or DCoordinates). Its format is similar to the other coordinate list structures. It uses single precision floating point ordered triples.

```
bytes          data
--------------------
4              'BVRT'
4              size of BVRT data - equal to number of vertices x 12.
4              float - x value of bump alignment vector 1
4              float - y of bump alignment vector 1
4              float - z of bump alignment vector 1
4              float - x of bump alignment vector 2
4              float - y of bump alignment vector 2
4              float - z of bump alignment vector 2
```
etc...

5.8 - Element list block (ELEM)

The Element list blocks contains the elements of the group - the 3D shapes or structures which represent the group. Usually, these shapes are polygons which use the coordinates listed in the CORD or DCOR blocks as their vertices. The Element list block is structured as a list of "Element" sub-blocks.

```
bytes          data
--------------------
4              'ELEM'
4              size of ELEM block data
x              element sub-block 1
y              element sub-block 2
```
etc....

Element Sub-Block

```
bytes          data
--------------------
1              flags - Element Flags
1              integer - Element Type
```

* the format of the rest of the sub-block depends on the Element Type field. See below for more information.

(if Element Type is QuadPoly)

bytes	data
4	color - element color - surface color and matting alpha
4*x	integer - 4 coordinate indices, where x is the number of bytes used for each index. See below for more details.

(if Element Type is MultiPoly)

bytes	data
4	integer - Element Size - the total size in bytes of the MultiPoly portion of the sub-block. See below for more details.
4	color - element color - surface color and matting alpha
4	integer - Element Skip - the number of QuadPoly records into which the MultiPoly was tesselated
n*x	n coordinate indices, where x is the number of bytes used for each index. See below for more details.

(if Element Type is MiscBlock)

bytes	data
4	integer - Element Size - the total size in bytes of the MiscBlock portion of the sub-block. See below for more details.
x	x bytes of some kind of data

Element Type

value	name	interpretation
0	QuadPoly	The rest of the Element sub-block follows the QuadPoly format
1	MultiPoly	The rest of the Element sub-block follows the MultiPoly format
2-255	MiscBlock	The rest of the Element sub-block is in some other format; see below

The Element sub-block can define any kind of three dimensional component, from simple convex polygons with fewer than four edges to complex polygons and parametric surfaces. An Element structure may also be used for special effects such as modification of shading attributes within a single group. The data within an Element is defined by the Element Type field. If the Type is 0, the Element contains a simple polygon as defined by the QuadPoly structure described below. If the Type is 1, the Element contains a complex polygon as defined by the MultiPoly structure described below.

The elementType values between 2 and 255 are not yet defined and should be ignored by skipping Element Size bytes. Because Element Size is a long, the possibilities of expansion within this structure are almost unlimited. If the number of Element Types are ever exhausted, types which contain sub-types could be defined. One problem which is typical of most complex data formats is that few applications can accept all of the types of data. Because of this, a method is provided within the Element structure to provide for decomposition of complicated polygons and surfaces into simple polygons without having the problem of duplicated surfaces. Each complex element (not a QuadPoly) which can be decomposed into simple QuadPoly elements will contain an Element Skip field. If the application reading the file can accept the complex element, it will read that element and then ignore the simple elements that follow it as defined by the Element Skip field. If the application cannot accept the complex element, it will be ignored and the sub elements into which it was decomposed will be read normally and will be used in place of the complex element.

It is important to note that it is NOT required that an application which creates a FACT file must decompose the complex elements it writes into simple elements. Since Electric Image will define and publish additions to the element structure, applications may depend upon Transporter to decompose any defined complex elements into simple elements. This should minimize the amount of work required to convert from any 3D model format into the FACT format.

The Element structure was designed to store polygons in an optimal amount of memory. The Element structure is flexible enough to handle objects containing more than 65535 coordinates when necessary and yet it can also be used to store polygons in model of 256 or fewer coordinates without wasting any memory. This is performed by using variable sized vertex indexes. The size of an index is determined by the number of coordinates in the group which precede the Element block. 1..255 coordinates are indexed by bytes, 256..65535 are indexed by shorts, 65536..16777215 are indexed by 24 bit integers and 16777216 or more are indexed by 32 bit long integers. When reading a list of vertexes from a complex element, the polygon should be closed when the Element Size is reached or a zero index is read. Any remaining bytes should be skipped (so that the number of bytes read is exactly equal to Element Size).

QuadPoly is a sub structure of the Element structure which is used to define a simple polygon. When the Type of an Element structure is zero, it contains a QuadPoly structure. The points field of this structure contain exactly four variable size coordinate indexes. A QuadPoly can define a point, line, triangle or convex quadrangle. Any unused vertices must contain zeros. Only QuadPoly records will be rendered. Complex polygons, defined by the MultiPoly structure below, are used only to display outlines and to define the normals of their sub polygons when rendered.

6 - Light data form block (LITE)

The light data form block appears at the same level as the Group (GRUP) form block. Its serial format is:

```
bytes          data
--------------------
4              'FORM'
4              integer - size of data in the Light form block
4              'LITE'
x              Light Header form block
```

6.1 - Light Header form block (LHDR)

The Light header form block contains information about a single light. It is currently the only block in the Light form block.

```
bytes          data
--------------------
4              'FORM'
4              integer - size of data in the Light Header form block
4              'LHDR'
x              Light Info Header block
y              Light Attributes block
z              Light Linkage Attributes block
a              Light Linkage Constraints (optional)
b              Light Shadow Info (optional)
c              Light Glow Info (optional)
d              Light Fog Info (optional)
```

6.1.1 Light info header (LINF)

The Light Info header contains some general information the light.

```
bytes          data
--------------------
4              'LINF'
4              integer - size of data in the Light Info Header block (112 here)
4              flags - Light Flags (see below)
4              flags - Project Flags (see below)
32             string - name of light, as a null-terminated string
4              integer - date of last modification (unused)
2              integer - light ID #
2              integer - Light Type (see below)
24             vector - light position
24             vector - light reference (where it's pointing)
4              integer - total children (number of children to follow)
4              integer - total offspring (number of offspring, children and children's children)
8              float - roll angle of light
```

Light Flags

```
bit #   name                  interpretation
-------------------------------------------
0       glow                  enable glow
1       fog                   enable fog
2       shadow                enable shadows
3       specular              enable specular reflection
4       blend                 enable blend (what blend?)
5       fog interpolate       enable fog interpolation
6       glow interpolate      enable glow interpolation
7       fog streak            enable fog streaking
8       glow streak           enable glow streaking
9       illumination          enable illumination
10      light projection      enable light projection
11      glow volume           enable glue volume
12      fog volume            enable fog volume
13      glow invert           invert glow
14      fog invert            invert fog
15      dropoff               enable dropoff
31      visibility            light is visible
```

Project Flags

```
bit #   name                  interpretation
-------------------------------------------
0       dropoff         show dropoff
1       angle                 show angle
2       position path         show position path
3       reference path        show reference path
4       glow                  show glow
5       fog                   show fog
6       flare                 show flare
7       size                  show size
8       offset inner cone     inner cone is offset
9       flare dropoff         show flare dropoff
10      volume dropoff show volume dropoff
```

Electric Image Animation System

Light Types

value	name	interpretation
0	none	
1	at camera	
2	infinite	
3	local	
4	ambient	
5	spot	
6	tube	
-1	generic vector	

6.1.2 - Light Attributes block (LATR)

The Light Attributes block contains some information about the photometric properties of the light.

bytes	data
4	'LATR'
4	size of data in the LATR block (52 here)
4	color - light color
8	float - factor
8	float - dropoff distance; 0.0 turns it off
8	float - intensity; 0.0 is off
8	float - inner cone angle (degrees)
8	float - outer cone angle (degrees)
8	float - size - set to 1.0

6.1.3 - Light Linkage Attributes block (LLNK)

The Light Linkage Attributes block is identical to the Group Linkage Attributes block except for the IFF type.

6.1.4 - Light Linkage Constraints (CNST) (optional)

6.1.5 - Light Shadow Info (LSHD) (optional)

```
bytes          data
--------------------
4              'LSHD'
4              size of data in the LSHD block (48 here)
2              integer - shadow type - 0 is buffer (? what are the other types?)
4              flags - buffer flags 1
4              flags - buffer flags 2
2              integer - buffer x
2              integer - buffer y
4              float - buffer zoom
4              float - buffer minimum bias
4              float - buffer delta bias
4              float - res factor
4              float - res min
2              integer - samples
4              integer - min samples
4              float - darkness
4              color - buffer color
```

6.1.6 - Light Glow Info (LGLW)

```
bytes          data
--------------------
4              'LGLW'
4              size of data in the LGLW block (40 here)
4              color - outside color
4              color - inside color
8              float - factor - interpolation factor, usually 1.0
8              float - inner cone, in degrees
8              float - outer cone, in degrees
8              float - intensity, 0.0 turns it off
```

6.1.7 - Light Fog Info (LFOG) (optional)

```
bytes          data
--------------------
4              'LFOG'
4              size of data in the LFOG block (40 here)
4              color - outside color
4              color - inside color
8              float - interpolation factor, usually 1.0
8              float - inner cone, in degrees
8              float - outer cone, in degrees
8              float - intensity: 0.0 turns it off
```