Data Format Description

Software Systems Flight Data Bases

Format Revision 9.0
Updated November, 1990

Revision A. March 1991

Data Format Description for Software Systems Flight Data Bases

1. Introduction

This document describes the concepts and file formats of a simple, binary visual system data base. This data base format can be created and edited using the "mgflt" version of the Software Systems MultiGen program, and is called the MultiGen flight data format, or simply the *flight* format. Flight has been designed to support low end visual systems, and be easily expandable. The data base format is generally designed to meet update requirements, and is therefore not necessarily optimized for real time use; however, it can easily be compiled into a different format, if desired, to accommodate a specific real time software's requirements.

2. Concepts Supported in Flight

The flight data base format is designed to support both simple and relatively sophisticated real time software applications. The full implementation of flight supports variable levels of detail, instancing (both within a file and to external files), replication, animation sequences, bounding boxes for real time culling, shadows, advanced scene lighting features, lights and light strings, transparency, texture mapping, and several other features.

A simple real time software package that interprets a flight data base can implement a subset of the data base specification, and use data bases that contain that subset. Such an application would scan for the color table, polygons, and vertices, and ignore the groups, objects and other more sophisticated features described here.

3. Data Base Hierarchy

The flight data base hierarchy allows the visual data base to be organized in logical groupings, and to facilitate real time functions such as level of detail switching and instancing. The flight data base is organized in a tree type structure. Each node of the tree can point down and across (see Figure 1).

Header: There is one header record per file. It is always the first record in the file and represents the top of the data base hierarchy and tree structure. There is never an across pointer in a header, and it always points down to a group.

Group: A group is a logical subset of a data base. MultiGen allows groups to be manipulated (translated, rotated, scaled, etc.) as a single entity. Groups can point down and across to other groups, level of detail beads, or to objects.

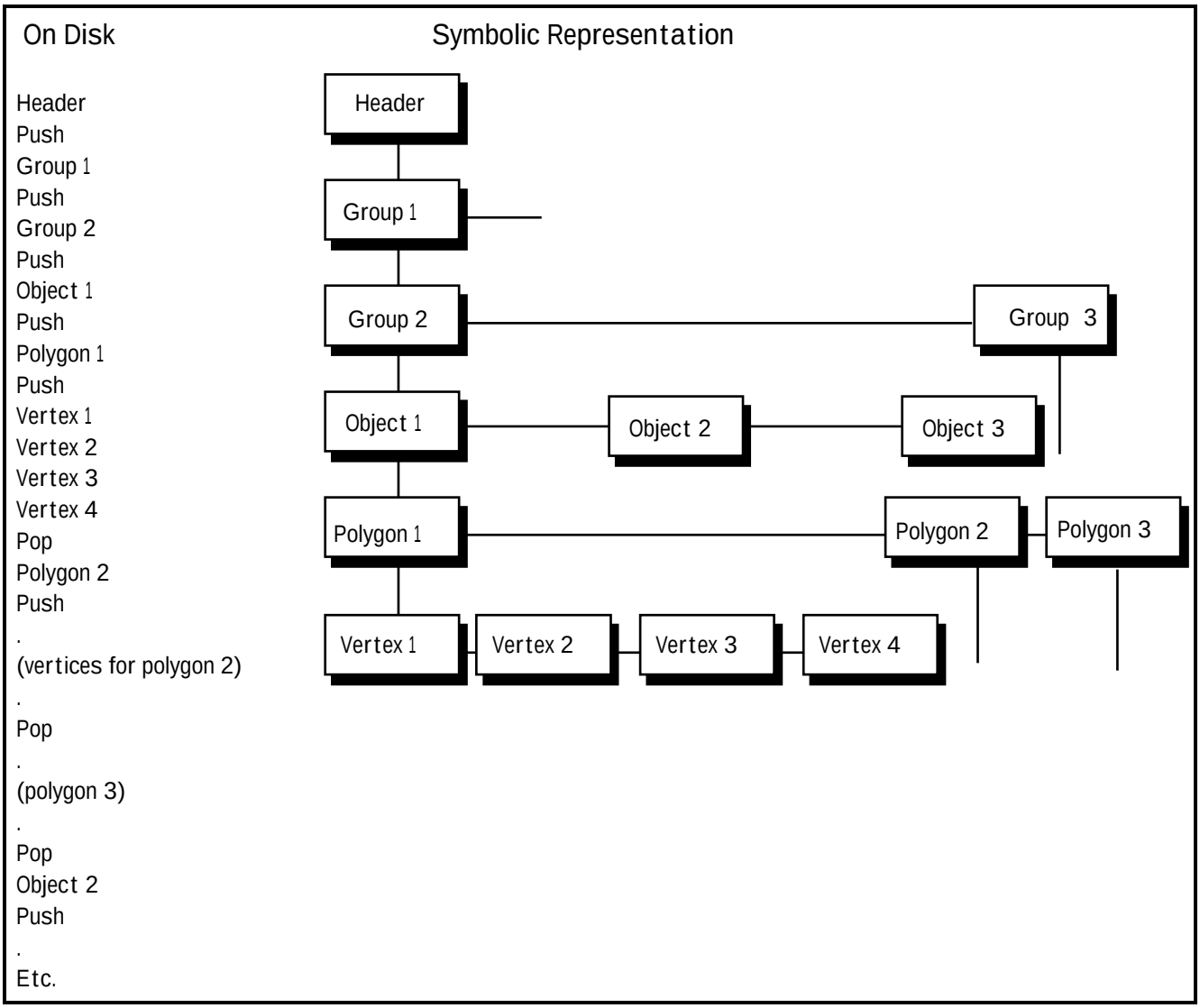

Figure 1. Example of Data Base Hierarchy

Level of Detail: A level of detail (LOD) bead is similar to a group, but it serves as a switch to turn the groups and objects below it on or off based on range. It has a switch in and switch out distance associated with it.

Object: An object is a logical collection of polygons. An object can point across to another object and down to a polygon.

Polygon: A polygon is a collection of vertices that describes a closed polygon in a counterclockwise direction. Polygons have a color code associated with them.

Nested Polygon: A *nested* polygon (sometimes called a subface), is a face that lies within, and is draw on top of, another "super" polygon. Nested faces can themselves be nested recursively.

Vertex: A vertex contains a coordinate x, y, and z. Sometimes vertices contain vertex normals and texture mapping information.

4. Data Base Files

When MultiGen writes a flight data base to disk, it converts the tree structure to a linear stream of op codes and data. The first part of each record is a header that contains an op code, record length, and, in some cases, an 8 byte ASCII id. A *push* op code is used to specify that the next bead is a down pointer from the last bead described. A *pop* op code returns to the level above. If neither a push or pop op code is found between nodes of the tree, an across pointer is implied. Thus, a polygon op code will be followed by a push op code, then all the vertices that describe the polygon, then a pop op code, which might be followed by the next polygon op code of the same object. Refer to Figure 1.

Flight data base files have the extension *.flt* by convention.

5. Instancing

Instancing is the ability to describe a group or object one time, and then display it one or more times with various transformations. The flight format supports instancing of objects and groups with rotate, translate, and scale operations.

In the flight format, a group or object definition that can be instanced is called an instance definition. An instance definition op code is followed by an ID and a stand alone data base tree. An instance is invoked from a group by following its op code by a transformation matrix op code, and the op codes for each translate, rotate, and scale operation (these are for MultiGen's use and can be ignored by the real time program), followed by an instance reference op code and an instance ID. Instance definitions can themselves contain instance definitions and references. Refer to Figure 2.

The flight format also allows entire data base files to be instanced. This is known as external instancing.

6. Replication

Replication is the ability to repeat the drawing of a group or object several times and apply a transformation each time. For example, a string of lights could be drawn by replicating a single light several times with a translation. In the flight format, replication is accomplished by following the group or object by one or more transformation op codes and a replication op code.

7. Bounding Boxes

When a bounding box op code is encountered in the flight data base format, it can be used by the real time software to determine if a particular group is in view. When used, the bounding box op code will immediately follow the group op code, and will include the extents created by instancing and replication.

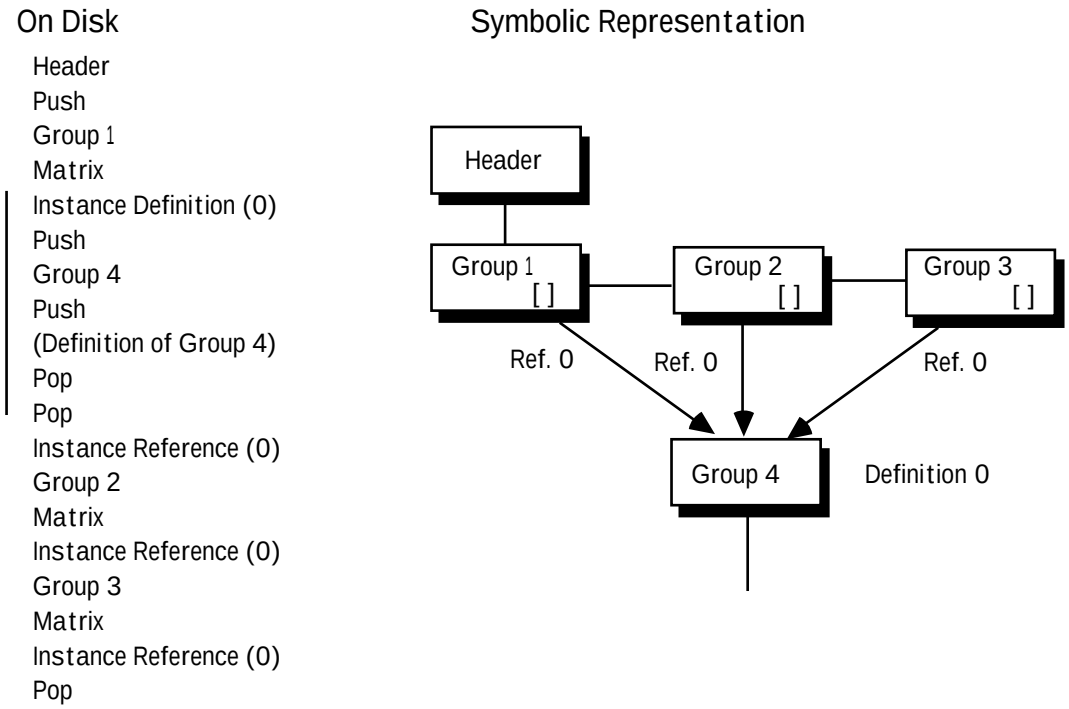

Figure 2. Instancing: Group 4 is Displayed Three Times

8. Detailed Explanation of Records

8.1. Header Record

The header record is found at the beginning of the data base file. The most important information for the real time software tells the value of LSB of integer coordinates. The *Units* field specifies whether the units are meters, feet, inches, etc. The Unit Multiplier/Divisor specifies how many of the units are represented by each LSB. A positive value represents a multiplier; thus a 4 in the units field and a 10 in the unit multiplier field would mean the vertex units are 10 feet. To avoid using floating point numbers, a negative multiplier is interpreted as meaning divide; thus a 4 in the units field and a -10 in the unit multiplier field would mean the vertex units are .1 feet.

The latitude and longitudes in the data base header exist if the data base was created from the MultiGen Terrain Option. They are stored as scaled integers, and can be converted to floating point by the equation,

$$l_{float} = l_{int} / (float) (1<< 30)*360.0$$

Positive latitudes reference the northern hemisphere, and positive longitudes reference the western hemisphere.

The Delta X and Y values to Place Database are used when several separate data bases are used to represent an area and each data base has a local origin of zero.

Record Type	Data Type	Length (Bytes)	Description
Header	Int	2	Op Code = 1
	Int	2	Length of the record
	Char	8	ID field (Not currently used)
	Int	4	Format revision level = 6
	Int	4	This data base revision level
	Char	32	Date and time of last revision
	Int	2	Next group ID number
	Int	2	Next LOD ID number
	Int	2	Next obj ID number
	Int	2	Next polygon ID number
	Int	2	Unit multiplier/divisor Positive for unit multiply Zero for no multiply/divide Negative for unit divide e.g. -100 = divide by 100
	Int	1	Units of vertex coordinates 0 = Meters 1 = Kilometers 4 = Feet 5 = Inches 8 = Naut. miles
	Int	1	Spare
	Bool	4	Flags (left to right) 0 = Save vertex normals 1-31 Spare
	Int	4	Southwest Data Base Corner Lat.
	Int	4	Southwest Data Base Corner Long.
	Int	4	Northeast Data Base Corner Lat.
	Int	4	Northeast Data Base Corner Long.
	Int	4	Latitude of Data Base Origin (0,0)
	Int	4	Longitude of Data Base Origin (0,0)
	Int	4	Projection Type 0 = Flat Earth 1 = Trapezoidal 2 = Round Earth
	Int	4	Southwest Data Base Coordinate X
	Int	4	Southwest Data Base Coordinate Y
	Int	4	Delta X to Place Database
	Int	4	Delta Y to Place Database
	Int	4 *52	Spare

8.2. Group Record

The group flags can be queried by the real time software if required. The *animation* flags specify that the beads directly below the group are an animation sequence; each bead is a frame of the sequence. The special effects IDs are normally zero, but can be set as required for special purpose meanings by the particular application program interpreting the data. The group *relative priority* specifies a fixed ordering of the object relative to the other groups at this level; since MultiGen sorts based on this field before outputting the data base, it can be ignored by the real time software.

Record Type	Data Type	Length (Bytes)	Description
Group	Int	2	Op Code = 2
	Int	2	Length of the record
	Char	8	7 char ASCII ID; 0 terminates
	Int	2	Group relative priority
	Int	2	Spare for fullword alignment
	Bool	4	Flags (left to right) 0 = Terrain 1 = Forward animation 2 = Cycling animation 3 = Bounding box follows 4-31 Spare
	Int	2	Special effects ID 1 - RT defines
	Int	2	Special effects ID 2 - RT defines
	Int	4*14	Spare

8.3. Level of Detail Record

The slant range distance is calculated by the real time software by using the distance from the eyepoint to the LOD center found in the bead; this center takes instancing and replication into account. When the *Use previous slant range* flag is set it means that the slant range is the same as the previous level of detail at the same level. This can be used to save the real time software the calculation of redundant slant ranges when determining if a level of detail should be displayed.

Record Type	Data Type	Length (Bytes)	Description
Level of Detail	Int	2	Op Code = 3
	Int	2	Length of the record
	Char	8	7 char ASCII ID; 0 terminates
	Int	4	Switch in distance
	Int	4	Switch out distance
	Int	2	Special effects ID 1 - RT defines
	Int	2	Special effects ID 2 - RT defines
	Bool	4	Flags (left to right) 0 = Use previous slant range 1-31 Spare

Int	12	Center coordinate of LOD block
Int	4*11	Spare

8.4. Object Record

The time of day object flags can be used to stop display of certain objects depending on the current time of day. The illumination flag, when set, means the object is self illuminating, and therefore not subject to the normal lighting effects. The shadow flag is used to indicate that the object represents the shadow of the rest of the group; when part of a moving model (e.g. an aircraft), this shadow can be displayed on the terrain or runway by the real time software with appropriate distortions to provide a realistic effect. The object *relative priority*, like the group relative priority, specifies a fixed ordering of the object relative to the others in its group; since MultiGen sorts based on this field, it can be ignored by the real time software.

Record Type	**Data Type**	**Length (Bytes)**	**Description**
Object	Int	2	Op Code = 4
	Int	2	Length of the record
	Char	8	7 char ASCII ID; 0 terminates
	Bool	4	Flags (left to right) 0 = Don't display in daylight 1 = Don't display at dusk 2 = Don't display at night 3 = Don't illuminate 4 = Flat shaded 5 = Group's shadow object 6 = Terrain 7-31 Spare
	Int	2	Object relative priority
	Int	2	Transparency factor = 0 for solid = 0xffff for totally clear
	Int	2	Special effects ID 1 - RT defines
	Int	2	Special effects ID 2 - RT defines
	Int	4*16	Spare

8.5. Polygon Record

Color codes in both polygons and vertices are 5 bits of color followed by 7 bits of intensity. The color record that follows the header defines the brightest RBG components of each color code. The other intensities can be calculated by linearly interpolating these components. Although a maximum of 128 intensities are defined in the format, the software interpreting the flight format can use less by ignoring the least significant bits of the intensities.

Record Type	Data Type	Length (Bytes)	Description
Polygon	Int	2	Op Code = 5
	Int	2	Length of the record
	Char	8	7 char ASCII ID; 0 terminates
	Int	4	Reserved for IR code
	Int	2	Polygon relative priority
	Int	1	How to draw the polygon = 0 Draw solid backfaced = 1 Draw solid no backface = 2 Draw wireframe and not close = 3 Draw closed wireframe = 4 Surround with wireframe in alternate color = 8 Omni-directional light = 9 Uni-directional light = 10 Bi-directional light
	Int	1	Spare
	Int	2	Primary color/intensity code
	Int	2	Secondary color code, if any
	Int	1	Not used
	Int	1	LSB on if texture template = 3 for axis type rotate = 5 for point rotate
	Int	2	Not used
	Int	2	*Texture pattern No., 0 or -1 if none
	Int	6	Spare

*Note: Because files created before this format revision have the texture pattern field set to 0, a 0 may mean no texture or pattern number 0. If the vertex for u, v fields exists, a 0 means pattern 0.

8.6. Vertex Records

Record Type	Data Type	Length (Bytes)	Description
Abs. Vertex	Int	2	Op Code = 7
	Int	2	Length of the record
	Int	4	X coordinate
	Int	4	Y coordinate
	Int	4	Z coordinate
	Float	8	*Optional texture (u, v)

Record Type	Data Type	Length (Bytes)	Description

Shaded Vertex	Int	2	Op Code = 8
	Int	2	Length of the record
	Int	4	X coordinate
	Int	4	Y coordinate
	Int	4	Z coordinate
	Int	2	Not used
	Int	2	Vertex color
	Float	8	*Optional texture (u, v)

Record Type	Data Type	Length (Bytes)	Description
Normal Vertex	Int	2	Op Code = 9
	Int	2	Length of the record
	Int	4	X coordinate
	Int	4	Y coordinate
	Int	4	Z coordinate
	Int	2	Not used
	Int	2	Vertex color
	Int	12	Vertex normal, scaled * 2**30
	Float	8	Optional texture (u, v)*

*To determine if texture (u,v) exists for a vertex, check the length of the record.

8.7. Control Records

Record Type	Data Type	Length (Bytes)	Description
Push Level	Int	2	Op Code = 10
	Int	2	Length of the record = 4
Pop Level	Int	2	Op Code = 11
	Int	2	Length of the record = 4
Push Subface	Int	2	Op Code = 19
	Int	2	Length of the record = 4
Pop Subface	Int	2	Op Code = 20
	Int	2	Length of the record = 4

8.8. Comment Records

Comment records contain text that can follow the header, group, level of detail, object, or polygon records.

Comments	Int	2	Op Code = 31

	Int	2	Length of the record
	Char	(variable)	Text description of data base

8.9 Color Table

Color Table	Int	2	Op Code = 32
	Int	2	Length of the record
	Int	6	Brightest RGB of color 0, intensity 127
	Int	6	Brightest RGB of color 1, intensity 127
	etc.		
	Int	6	Brightest RGB of color 27
	Spare	8*6	Space for colors 28-32
	Int	6	Fixed intensity color 0 (4096)
	Int	6	Fixed intensity color 1 (4097)
	etc.		

Note that the first part of the color record contains the *brightest* RGB of colors 0-27, intensity 127. Intensities 0-126 for each of these colors are calculated by linearly interpolating between intensity 0, which is black for all colors (RGB 0, 0, 0), and the values provided for intensity 127. Space is provided for colors 28-32, but they are not currently used by MultiGen. The second part of the color table contains the RGBs of 56 fixed intensity colors which do not require any interpolation. The color/intensity field of the polygon or vertex attributes referencing these colors will contain a value of 4096 for first fixed intensity color, 4097 for the second fixed intensity color, etc.

8.10. Transformations

Trans. Matrix	Int	2	Op Code = 49
	Int	2	Length of the record
	Float	16*4	4X4 single prec. matrix

Note: Op codes 40-48 follow the transformation matrix, and specify the individual tranformations that make up the matrix. These op codes are for MultiGen use only, and should be ignored by the real time software reading the file.

8.11. Geometry

Vector	Int	2	Op Code = 50
	Int	2	Length of the record
	Int	4	i component, scaled 2**30
	Int	4	j component
	Int	4	k component
Bounding Box	Int	2	Op Code = 51
	Int	2	Length of the record
	Int	12	X, Y, Z of lowest corner
	Int	12	X, Y, Z of highest corner

8.12. Replication and Instancing

Replicate	Int	2	Op Code = 60
	Int	2	Length of the record
	Int	2	Number of replications
	Int	2	Spare for fullword alignment
Instance Ref.	Int	2	Op Code = 61 (Rev 3 code = 16)
	Int	2	Length of the record
	Int	2	Spare
	Int	2	Instance definition number
Instance Def.	Int	2	Op Code = 62 (Rev 3 code = 17)
	Int	2	Length of the record
	Int	2	Spare
	Int	2	Instance definition number
External Ref.	Int	2	Op Code = 63
	Int	2	Length of the record
	Char	80	79 char ASCII path; 0 terminates

8.13. Texture Pattern File Reference

One record exists for each different texture pattern referenced in the database.

Texture Ref.	Int	2	Op Code = 64
	Int	2	Length of the record
	Char	80	Filename of texture pattern
	Int	4	Pattern index
	Int	4	x location in texture palette
	Int	4	y location in texture palette

Add 1 to the pattern index and the polygon pattern reference number on Silicon Graphics machines because the texture pattern ids start at 1.

A palette and pattern system can be used to reference the texture patterns. A MultiGen texture palette is made up of 64 patterns, currently 256 texels on a side. The pattern index for the first palette is 0 - 63, for the second palette 64 - 127, etc. Note that if less than 64 patterns exist on a palette, several pattern indices will be unused. The x and y palette locations can be used to store offset locations in the palette for display.

8.14. Eyepoint Positions

Eyepoints	Int	2	Op Code = 65
	Int	2	Length of the record

Last Position 0	Int	3*4	X, Y, Z of rotation center
	Float	3*4	Yaw, pitch, roll angles
	Float	16*4	4X4 single prec. rotation matrix
	Float	4	Field of view
	Float	4	Scale
	Float	2*4	Near and far clipping plane
	Float	16*4	4X4 single prec. fly-through matrix
	Float	3*4	X, Y, Z of eyepoint in database
	Float	4	Yaw of flythrough
	Float	4	Pitch of flythrough
	Float	3*4	i, j, k vector for eyepoint direction
	Int	4	Flag (True if no fly-through)
	Int	4	Flag (True if ortho drawing mode)
	Int	4	Flag (True if this is a valid eyepoint)
	Int	11*4	Spare
Eyepoint 1	Same as last position (256 bytes)		
Eyepoint 2	Same as last position (256 bytes)		
Eyepoint 3	Same as last position (256 bytes)		
Eyepoint 4	Same as last position (256 bytes)		
Eyepoint 5	Same as last position (256 bytes)		
Eyepoint 6	Same as last position (256 bytes)		
Eyepoint 7	Same as last position (256 bytes)		
Eyepoint 8	Same as last position (256 bytes)		
Eyepoint 9	Same as last position (256 bytes)		

Note: Total length of record is 256 bytes * 10 eyepoints plus header.

9.0 Texture Pattern Files

Flight format does not have its own texture pattern format but rather uses existing texture formats and refers to patterns by filename (see section 8.13). The following file formats are currently supported:

AT & T image 8 format (8 bit color lookup)
AT & T image 8 template format
SGI intensity modulation (**i**)
SGI intensity modulation with alpha (**ia**)
SGI RGB(**rgb**)
SGI RGB with alpha (**rgba**)

The format of the file can be determined either from the file name extension, from magic numbers within the file, or from the texture attribute file as described below.

9.01 The SGI Texture Format

Header (512 bytes)

		i	**ia**	**rgb**	**rgba**
magic number	0x01da	same	same	same	
type		0001	same	same	same
dimension		unused			
xsize		number of texels in the u direction (horizontal)			
ysize		number of texels in the v direction (vertical)			
zsize		number of components per texel			
wastebytes		unused			
char name [80]		filename			
etc...					

Image Data (xsize * ysize * zsize bytes)

512 bytes offset from start of file

1st component xsize * ysize bytes	intensity	intensity	red	red
2nd component xsize * ysize bytes	N/A	alpha	green	green
3rd component xsize * ysize bytes	N/A	N/A	blue	blue
4th component	N/A	N/A	N/A	alpha

Definitions

magic number - (2 bytes) always 0x01da for SGI image file format. This is an arbitrary number.

type - (2 bytes) always 0x0001 for non compressed files and 0x0101 for run length encoded files. Run length encoded files are not supported by MultiGen.

xsize - (2 bytes) number of texels in the u direction (horizontal).

ysize - (2 bytes) number of texels in the u direction (horizontal).

zsize - (2 bytes) number of texels in the u direction (horizontal).

9.1 Texture Attribute Files

A corresponding attribute file is created for each texture pattern, with the name of the attribute file the same as the texture file followed by the extension ".attr". These attribute files are used by MultiGen, and may not be necessary for the real time software using the data base. They are in the following format:

Int	4	Number of pixels in u direction
Int	4	Number of pixels in v direction
Int	4	Real world size u direction
Int	4	Real world size v direction
Int	4	X component of up vector
Int	4	Y component of up vector
Int	4	File format -1 Not used 0 AT & T image 8 pattern 1 AT & T image 8 template 2 SGI intensity modulation 3 SGI intensity w/ alpha 4 SGI RGB 5 SGI RGB w/ alpha
Int	4	Minification filter type
Int	4	Magnification filter type
Int	4	Repetition type
Int	4	Repetition type in u direction
Int	4	Repetition type in v direction
Int	4	Modify flag
Int	4	x pivot point for rotating textures
Int	4	y pivot point for rotating textures
Int	17*4	17 spare words for expansion

The attribute file is used to determine how to parse the texture pattern file and to determine how the texture hardware and software environment is to be set for that pattern.