Software Systems

Flight Format™ Specification

Revision 13.0
June, 1993

TABLE OF CONTENTS

LIST OF TABLES

Data Format Description for Software Systems Flight Data Bases

1 Introduction

1.1 About the Flight Format Description

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 Software Systems' **MultiGen**, and is called the **MultiGen Flight** data format, or simply the **Flight** format.

1.2 Document Conventions

Paragraphs which contain a discussion of material new to the current software release are marked with a revision bar, such as the one that appears to the right.

---

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, degrees of freedom, 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, material properties, 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.

Version 13 of **Flight** supports two methods of vertex coordinate storage, integer and double precision float. Many record types have two representations, one storing integer coordinates the other double precision. Databases are either entirely integer or entirely double precision however. Integer and double precision type records should not be mixed in one database file.

Note: SOFTWARE SYSTEMS WILL NO LONGER SUPPORT INTEGER FORMAT VERTEX COORDINATES AS OF VERSION 14.0 OF FLIGHT. Version 14 of MultiGen (and version 13 of ModelGen) will automatically convert each integer data base read to floating point format.

Integer record formats supported under version 13.0 are listed at the end of this document.

---

3 Data Base Hierarchy

The **Flight** data base hierarchy allows the visual data base to be organized in logical groupings, and is designed to facilitate real time functions such as level of detail switching and instancing. The **Flight** data base is organized in a tree structure. Each node (or bead) of the tree can point down and/or 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. The header always points down to a group.

Group: A group bead is used to organize 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.

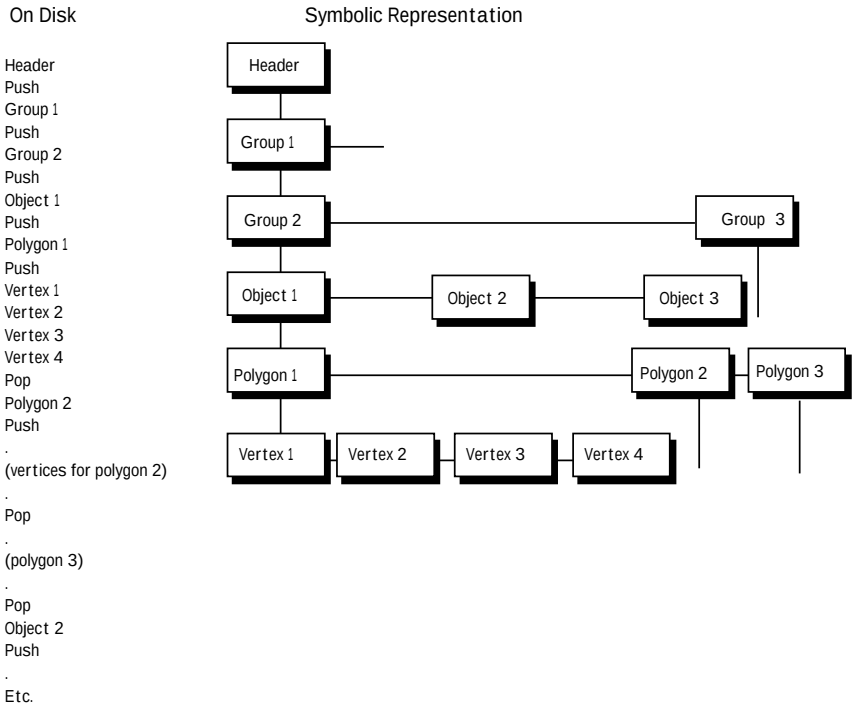

Figure 1. Example of Data Base Hierarchy

Level of Detail: A level of detail (LOD) bead is similar to a group, but it serves as a switch to turn the display of everything below it on or off based on range (the switch in/switch out distance and center location).

Degree of Freedom: A degree of freedom (DOF) bead is similar to a group with several transformations. It is used to specify the articulation of parts in the database and to set limits on the motion of those parts.

Object: An object bead contains a logical collection of polygons. An object can point across to another object, group or LOD and down to a polygon.

Polygon: A polygon bead contains a set of vertices that describe a closed polygon in a counter clockwise direction. Polygons have color, texture, materials, transparency etc. associated with them.

Nested Polygon: A *nested* polygon (or sub-face), is a bead describing a face that lies within, and is drawn on top of, another "super" polygon. Nested faces can themselves be nested. This construct is used to determine z buffer priority.

Vertex: A vertex contains a coordinate x, y, and z. Some vertices also contain vertex normals and texture mapping information. Coordinates may be stored in either integer or double precision format (but note that VERSION 14 OF **FLIGHT** WILL NOT SUPPORT INTEGER COORDINATE STORAGE). Double precision coordinates are stored in a vertex table near the beginning of the file, and are accessed through relative offset pointers after the Polygon record. Integer coordinates are not pooled and are stored in distinct records after each Polygon record.

4 Data Base Files

When MultiGen writes a **Flight** data base to disk, it converts the tree structure to a linear stream of records. The first part of each record is a header which specifies the record opcode (e.g., its type), record length, and, in some cases, an 8 byte ASCII ID. A record containing the push opcode (or 'push record') is used to represent each down pointer. A record containing a pop opcode (or 'pop record') returns to the previous level of hierarchy.

If a record's opcode is neither push nor pop, a sibling pointer is implied. Thus, a record with a polygon opcode will be followed by a push record, then the vertex information describing the polygon, then a pop record. This, in turn, will be followed by the polygon record for the next polygon in the same object, or by a pop record to return to object level. 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 operations such as rotate, translate, and scale, and put.

In the **Flight** format, a group or object definition that can be instanced is called an instance definition. An instance definition contains a record with an instance definition opcode, followed by an ID and a stand alone data base tree. An instance is invoked from a group by following the group record with a record containing a transformation matrix, and then records 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 opcode 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 referencing.

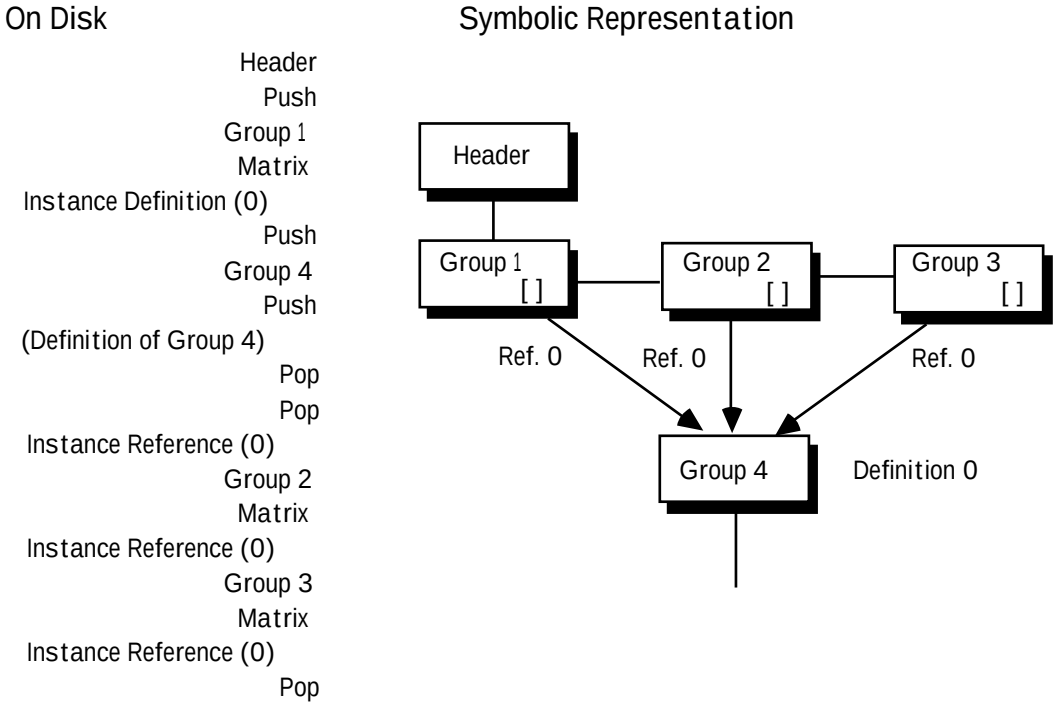

Figure 2. Instancing: Group 4 is Displayed Three Times

6 Replication

Replication is the ability to repeat the drawing of a group or object several times, applying 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 by one or more transformation opcode records and a replication opcode record.

7 Bounding Boxes

Bounding box records can be used by the real time software to determine if a particular group is in view. The (optional) bounding box opcode record is placed immediately after the group record, and includes the extents created by instancing and replication.

8 Flight Record Format

8.1 Header Record

The header record is found at the beginning of the data base file. The most important fields for the real time software are those which specify database units:

The **Vertex storage type** field indicates whether the database uses integer or floating point coordinates. Coordinates must all be of the same type within a given file. Note: VERSION 14.0 OF **FLIGHT** WILL NO LONGER SUPPORT INTEGER DATA BASE COORDINATES.

The **Vertex coordinate units** field specifies whether the units are meters, feet, inches, etc.

The **Unit multiplier/Divisor** should be 1 in a floating point database.

In an integer data base, the **Unit multiplier** supplies a scale for the **vertex coordinate units**. A positive number multiplies the vertex coordinate units, while a negative number is interpreted as a divide. For example, if the **Vertex storage type** = 4 and the **Unit multiplier** = 10, the vertex units are 10 feet. If the multiplier is -10, the vertex units are .1 feet.

Latitude and longitude values are stored in the data base header if it was created using the MultiGen Terrain Option. They are scaled integers which can be converted to floating point by the C language equation,

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

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

Delta x and y values are used to "place" the database when several separate databases are used to represent an area, each of which has a local origin of zero.

Header Record Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 1
Int	2	Length of the record
Char	8	ID field (Not currently used)
Int	4	Format revision level
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) Always equal to 1 for floating point databases.
Int	1	Vertex coordinate units 0 = Meters 1 = Kilometers 4 = Feet 5 = Inches 8 = Naut. miles
Int	1	if TRUE set texwhite on new polygons
Bool	4	Flags (bits, from 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
Int	4	Longitude of Data Base Origin
Int	4	Projection Type 0 = Flat Earth 1 = Trapezoidal 2 = Round Earth 3 = Lambert 4 = UTM
Int	4	Not Used
Int	4	Not Used
Int	4	Not Used
Int	4	Not Used
Int	4	Lambert Upper Lattitude
Int	4	Lambert Lower Lattitude
Int	4	Not Used
Int	2	Next degree of freedom ID number
Int	2	Vertex Storage Type 0 = Integer 1 = Double Precision Float
Int	4	Database Origin 100 = Flight 200 = DIG I/DIG II 300 = Evans and Sutherland CT5A/CT6 400 = PSP DIG 600 = General Electric CIV/CV 700 = Evans and Sutherland GDF
Double	8	Southwest Data Base Coordinate x
Double	8	Southwest Data Base Coordinate y
Double	8	Delta x to Place Database
Double	8	Delta y to Place Database

8.2 Group Record

Group flags are available to the real time software as follows: The *animation* flags specify that the beads directly below the group are an animation sequence, each bead being one frame of the sequence. The special effects IDs are normally zero, but can be set to support an application program's interpretation of the data. The group's *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 saving the data base, it can be ignored by the real time software.

Group Record Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 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 (bits, from left to right) 0 = Terrain 1 = Forward animation 2 = Cycling animation 3 = Bounding box follows 4 = Freeze Bounding Box 5= Default parent 6-31 Spare
Int	2	Special effects ID 1 - defined by real time
Int	2	Special effects ID 2 - defined by real time
Int	2	Significance Flags
Int	2	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.

Level of Detail Double Precision Record Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 73
Int	2	Length of the record
Char	8	7 char ASCII ID; 0 terminates
Int	4	Spare
Double	8	Switch in distance
Double	8	Switch out distance
Int	2	Special effects ID 1 - defined by real time
Int	2	Special effects ID 2 - defined by real time
Bool	4	Flags (bits, from left to right)

		0 = Use previous slant range 1 = SPT flag: set to 0 for replacement LOD, 1 for additive LOD 2 = Freeze center (don't recalculate) 3-31 Spare
Double	8	Center coordinate x of LOD block
Double	8	Center coordinate y of LOD block
Double	8	Center coordinate z of LOD block

8.4 Degree of Freedom Record

The fields of the degree of freedom record combine to specify a local coordinate system and the range allowed for translation, rotation, and scale with respect to that coordinate system.

The degree of freedom record can be viewed as a list of applied transformations consisting of the following elements:

[PTTTRRRSSSP]

It is important to understand the order in which these transformations are applied to the geometry. A pre-multiplication is assumed by MultiGen, so the transformation linked list must be read *backwards* to describe its affect on the geometry contained below the DOF. Here, a degree of freedom is interpreted as a Put followed by three Scales, three Rotates, three Translates and a final Put. Taking the transformations in reverse order, they represent:

1. A Put (3 point to 3 point transformation). This Put, brings the local coordinate system to the world origin with its x-axis aligned along the world x-axis and with the local y-axis in the world x-y plane. Testing against the DOF's constraints is performed in this standard position and then the final Put repositions the local coordinate system in its original position. The first Put is therefore the inverse of the last.

2. Scale in x

3. Scale in y.

4. Scale in z.

5. Rotation about z (twist)

6. Rotation about y (inclination)

7. Rotation about x (azimuth)

8. Translation in x.

9. Translation in y.

10. Translation in z.

11. A final Put. This Put moves the DOF local coordinate system back to its original position in the scene. (See 1).

The degree of freedom record specifies the *minimum*, *maximum*, and *current* values for each transformation. Only the *current* value affects the actual transformation applied to the geometry. The *increment* value is included to allow the setting of discrete allowable values within the range of legal values represented by the DOF.

Degree of Freedom Double Precision Record Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 14
Int	2	Length of the record
Char	8	7 char ASCII ID; 0 terminates
Double	8	Origin of the DOF's local coordinate system; x coordinate
Double	8	Origin of the DOF's local coordinate system; y coordinate
Double	8	Origin of the DOF's local coordinate system; z coordinate
Double	8	Point on the x-axis of the DOF's local coordinate system; x coordinate
Double	8	Point on the x-axis of the DOF's local coordinate system; y coordinate
Double	8	Point on the x-axis of the DOF's local coordinate system; z coordinate
Double	8	Point in xy plane of the DOF's local coordinate system; x coordinate.
Double	8	Point in xy plane of the DOF's local coordinate system; y coordinate.
Double	8	Point in xy plane of the DOF's local coordinate system; z coordinate.
Double	8	Minimum z value with respect to the local coordinate system.
Double	8	Current z value with respect to the local coordinate system.
Double	8	Maximum z value with respect to the local coordinate system.
Double	8	Increment in z.
Double	8	Minimum y value with respect to the local coordinate system.
Double	8	Current y value with respect to the local coordinate system.
Double	8	Maximum y value with respect to the local coordinate system.
Double	8	Increment in y.
Double	8	Minimum x value with respect to the local coordinate system.
Double	8	Current x value with respect to the local coordinate system.
Double	8	Maximum x value with respect to the local coordinate system.
Double	8	Increment in x.
Double	8	Minimum azimuth (rotation about the x-axis).
Double	8	Current azimuth
Double	8	Maximum azimuth.
Double	8	Increment in azimuth
Double	8	Minimum increment (rotation about the y-axis).
Double	8	Current increment
Double	8	Maximum increment.
Double	8	Increment in increment
Double	8	Minimum twist (rotation about the z-axis).
Double	8	Current twist
Double	8	Maximum twist.

Double	8	Increment in twist
Double	8	Minimum z scale (about local origin).
Double	8	Current z scale (about local origin).
Double	8	Maximum z scale (about local origin).
Double	8	Increment for scale in z.
Double	8	Minimum y scale (about local origin).
Double	8	Current y scale (about local origin).
Double	8	Maximum y scale (about local origin).
Double	8	Increment for scale in y.
Double	8	Minimum x scale (about local origin).
Double	8	Current x scale (about local origin).
Double	8	Maximum x scale (about local origin).
Double	8	Increment for scale in x.

8.5 Object Record

The time of day object flags can be used to inhibit display of certain objects depending on the current time of day. The illumination flag, when set, means the object is self illuminating and is not subject to normal lighting effects. The shadow flag is used to indicate that the object represents the shadow of the rest of the group. When used as part of a moving model (e.g. an aircraft), the real time software can apply appropriate distortions to create a realistic shadow on the terrain or runway. The object's *relative priority* specifies a fixed ordering of the object relative to the others in its group. Since MultiGen sorts on relative priority, it can be ignored by the real time software.

Object Record Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 4
Int	2	Length of the record
Char	8	7 char ASCII ID; 0 terminates
Bool	4	Flags (bits from 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 - defined by real time
Int	2	Special effects ID 2 - defined by real time

Int	2	Significance
Int	2	Spare

8.6 Polygon Record

Color codes are made up of 5 bits of color followed by 7 bits of intensity in both polygons and vertices. The color record which follows the header defines the brightest RGB components of each color code. The other intensities can be calculated by linearly interpolating these components. Although **Flight** format allows as many as 128 intensities to be defined, the software interpreting the **Flight** format can use fewer by ignoring the least significant bits of the intensities.

If a polygon contains a non-negative material code, its apparent color will be a combination of the face color and the material color as described in the Material Record section below.

If a polygon contains a non-negative material with an alpha component, and the transparency field is set, the total transparency is the product of the material alpha and the face transparency.

Polygon Record Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 5
Int	2	Length of the record
Char	8	7 char ASCII ID; 0 terminates
Int	4	IR Color 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 closed = 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	Texwhite = if TRUE, draw textured polygon white (see note 1 below)
Int	2	Primary color/intensity code
Int	2	Secondary color code, if any
Int	1	Not used
Int	1	Set template transparency =0 None =1 Fixed = 3 Axis type rotate = 5 Point rotate
Int	2	Detail texture pattern no. -1 if none
Int	2	Texture pattern no. -1 if none (see note 2 below)
Int	2	Material code [0-63]. -1 if none.
Int	2	Surface material code (for DFAD)
Int	2	Feature ID (for DFAD)

Int	4	IR Material codes
Int	2	Transparency = 0 for solid = 0xffff for totally clear
Int	2	Spare

Notes: (1) If the texwhite field is set, polygon color will be ignored if and only if the face has been textured. (2) A 0 in the texture pattern field may indicate either that the face is not textured (if created before version 10 of **Flight**) or that texture pattern 0 has been applied (in version 10.0 and after). In the latter case, texture u,v fields will be included in vertex records (see below).

8.7 Vertex Table

Double Precision vertex records are stored in a vertex pool for the entire database. This pool is located near the beginning of the **Flight** file, ahead of all of the polygon records.

The vertex table header record signifies the start of the vertex table. It contains a one word entry specifying the total length of the vertex table, which is equal to the length of the header record plus the length of the following vertex records. The individual vertex records follow this header, each starting with its opcode. The length field in the vertex table header record makes it possible to skip over the vertex records until the data is actually needed.

Vertices may be shared, and are accessed through the vertex list record that follows each polygon record. The length of each vertex list record is determined by the number of vertices in the polygon; for each vertex, there is a one word field pointing to its vertex record in the vertex table. Since this offset includes the length of the vertex header record, the value of the first pointer is 8.

Floating Point Format Vertex Table Header Record

Data Type	Length (Bytes)	Description
Int	2	Opcode = 67
Int	2	Length of the record
Int	4	Length of this record plus length of the vertex pool.

(followed by) **Floating Point Format Vertex Records**

Record Type	Data Type	Length (Bytes)	Description
Shaded	Int	2	Opcode = 68
Vertex	Int	2	Length of the record
	Int	2	Vertex Color -1 if Not Shaded
	Bool	2	Flags (bits, from left to right) 0 = Hard edge flag 1 = Don't touch normal when shading 2-15 Spare
	Double	8	x coordinate
	Double	8	y coordinate
	Double	8	z coordinate

.....

Shaded	Int	2	Opcode = 69
Vertex/	Int	2	Length of the record
Vertex	Int	2	Vertex Color -1 if Not Shaded
Normal	Bool	2	Flags (bits, from left to right) 0 = Hard edge flag 1 = Don't touch normal when shading 2-15 Spare
	Double	8	x coordinate
	Double	8	y coordinate
	Double	8	z coordinate
	Float	12	Vertex normal
	Int	4	Not Used
.....			
Shaded	Int	2	Opcode = 71
Vertex/	Int	2	Length of the record
Textured	Int	2	Vertex Color -1 if Not Shaded
	Bool	2	Flags (bits, from left to right) 0 = Hard edge flag 1 = Don't touch normal when shading 2-15 Spare
	Double	8	x coordinate
	Double	8	y coordinate
	Double	8	z coordinate
	Float	8	Texture(u,v)
.....			
Shaded	Int	2	Opcode = 70
Vertex/	Int	2	Length of the record
Vertex	Int	2	Vertex Color -1 if Not Shaded
Normal/	Bool	2	Flags (bits, from left to right)
Textured			0 = Hard edge flag 1 = Don't touch normal when shading 2-15 Spare
	Double	8	x coordinate
	Double	8	y coordinate
	Double	8	z coordinate
	Float	12	Vertex normal
	Float	8	Texture(u,v)
	Int	4	Not Used

Vertex List: Floating Point Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 72
Int	2	Length of the record
Int	4	Byte offset to this vertex record in vertex table;
.		Number of vertices in this list is determined by: (Length of this record - 4)/4.

8.8 Control Records

Control Record Format

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

8.9 Comment Records

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

Comment Record Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 31
Int	2	Length of the record
Char	(variable)	Text description of data base

8.10 Color Table

RGB is made up of two bytes of red, two bytes of green, and two bytes of blue.

Color Table Record Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 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	4*6	Space for colors 28-32
Int	6	Fixed intensity color 0 (4096)
Int	6	Fixed intensity color 1 (4097)
etc.		

The color record must follow the header record and precede the first push.

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 the first fixed intensity color, 4097 for the second fixed intensity color, etc.

8.11 Material Table

The material table contains descriptions of 64 material types. The material table is not written with the data base unless a face has been assigned a non-negative material code. The appearance of a face in MultiGen is a combination of the face color and the material code. The material record must follow the header record and precede the first push. The face color is factored into the material properties as follows:

Ambient

The displayed material's ambient component is the product of the ambient component of the material and the face color:

Displayed ambient (red) = Material ambient (red)* face color(red)
Displayed ambient (green) = Material ambient (green)* face color (green)
Displayed ambient (blue) = Material ambient (blue)* face color(blue)

For example, suppose the material has an ambient component of {1.0, .5, .5} and the face color is {100, 100, 100}. The displayed material will have as its ambient color {100, 50, 50}.

Diffuse:

As with the ambient component, the diffuse component is the product of the diffuse component of the material and the face color:

Displayed diffuse (red) = Material diffuse (red)* face color(red)
Displayed diffuse (green) = Material diffuse (green)* face color(green)
Displayed diffuse (blue) = Material diffuse (blue)* face color(blue)

Specular:

Unlike ambient and diffuse components, the displayed specular component is taken directly from the material:

Displayed specular (red) = Material specular (red)
Displayed specular (green) = Material specular (green)
Displayed specular (blue) = Material specular (blue)

Emissive:

The displayed emissive component is taken directly from the material:

Displayed emissive (red) = Material emissive (red)
Displayed emissive (green) = Material emissive (green)
Displayed emissive (blue) = Material emissive (blue)

Shininess:

MultiGen drawing takes the shininess directly from the material. Specular highlights are tighter, with higher shininess values.

Alpha:

An alpha of 1.0 is fully opaque, while 0.0 is fully transparent. When drawing polygons (faces), MultiGen combines the transparency value of the polygon record with the alpha value of the material record. The final alpha applied to a polygon as it is drawn by MultiGen is a floating point number between 0.0 (transparent) and 1.0 (opaque), and is computed as follows:

final alpha = material alpha * (1.0 - (polygon transparency / 0xffff))

Material Table Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 66
Int	2	Length of the record
Float	4	Ambient red component of material 0.*
Float	4	Ambient green component of material 0.*
Float	4	Ambient blue component of material 0.*
Float	4	Diffuse red component of material 0*.
Float	4	Diffuse green component of material 0*.
Float	4	Diffuse blue component of material 0.*
Float	4	Specular red component of material 0.*
Float	4	Specular green component of material 0.*
Float	4	Specular blue component of material 0.*
Float	4	Emissive red component of material 0.*
Float	4	Emissive green component of material 0.*
Float	4	Emissive blue component of material 0.*
Float	4	Shininess. (A single precision floating point value [0.0-128.0]).
Float	4	Alpha. (A single precision floating point value [0.0-1.0], where 1.0 is opaque).

Bool	4	Flags 0 = Materials used 1-31 Spare
Int	4*31	Spares for material 0.
Float etc.	4	Ambient red component of material 1.*

*Single precision floating point values, [0.0, 1.0]

8.12 Transformations

Transformation Matrix Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 49
Int	2	Length of the record
Float	16*4	4x4 Single Precision Matrix

Note: Opcodes 40-48 and 76-82 follow the transformation matrix, and specify the individual transformations that make up the make the matrix. These opcodes are for MultiGen use only, and should be ignored by the real time software reading the file.

8.13 Geometry

Vector Formats

Record Type	Data Type	Length (Bytes)	Description
Vector	Int	2	Opcode = 50
	Int	2	Length of the record
	Float	4	i component, 32 bit float
	Float	4	j component
	Float	4	k component
Bounding Box	Int	2	Opcode = 74
Floating	Int	2	Length of the record
	Double	8	x coordinate of lowest corner
	Double	8	y coordinate of lowest corner
	Double	8	z coordinate of lowest corner
	Double	8	x coordinate of highest corner
	Double	8	y coordinate of highest corner
	Double	8	z coordinate of highest corner

8.14 Replication and Instancing

Replication and Instancing Formats

Record Type	Data Type	Length (Bytes)	Description
Replicate	Int	2	Opcode = 60
	Int	2	Length of the record
	Int	2	Number of replications
	Int	2	Spare for fullword alignment
Instance Ref.	Int	2	Opcode = 61 (Rev 3 code = 16)
	Int	2	Length of the record
	Int	2	Spare
	Int	2	Instance definition number
Instance Def.	Int	2	Opcode = 62 (Rev 3 code = 17)
	Int	2	Length of the record
	Int	2	Spare
	Int	2	Instance definition number
External Ref.	Int	2	Opcode = 63
	Int	2	Length of the record
	Char	200	199 char ASCII Path; 0 terminates

8.15 Texture Pattern File Reference

There is one record for each texture pattern referenced in the database. These records must follow the header record and precede the first push.

Texture Pattern File Reference Format

Data Type	Length (Bytes)	Description
Int	2	Opcode = 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 256 patterns, currently 512 texels on a side. The pattern index for the first palette is 0 - 255, for the second palette 256 - 511, etc. Note that if less than 256 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.

8.16 Eyepoint Positions

Eyepoint Position Double Precision Format

Record Type	Data Type	Length (Bytes)	Description
Eyepoints	Int	2	Opcode = 83
	Int	2	Length of the record
Last Position 0	Double	3*8	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 Fly Through
	Float	4	Pitch of Fly Through
	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		
Eyepoint 2	Same as Last Position		
Eyepoint 3	Same as Last Position		
Eyepoint 4	Same as Last Position		
Eyepoint 5	Same as Last Position		
Eyepoint 6	Same as Last Position		
Eyepoint 7	Same as Last Position		
Eyepoint 8	Same as Last Position		
Eyepoint 9	Same as Last Position		

9 Texture Files

9.1 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
SGI intensity modulation with alpha
SGI RGB
SGI RGB with alpha

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.

9.2 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:

Texture Attribute File Format

Data Type	Length (Bytes)	Description
Int	4	Number of texels in u direction
Int	4	Number of texels 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 type -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: 0 - TX_POINT 1 - TX_BILINEAR 2 - TX_MIPMAP (Obsolete) 3 - TX_MIPMAP_POINT 4 - TX_MIPMAP_LINEAR 5 - TX_MIPMAP_BILINEAR 6 - TX_MIPMAP_TRILINEAR 7 - None 8 - TX_BICUBIC 9 - TX_BILINEAR_GEQUAL 10- TX_BILINEAR_LEQUAL 11 - TX_BICUBIC_GEQUAL 12 - TX_BICUBIC_LEQUAL

Int	4	Magnification filter type: 0 - TX_POINT 1 - TX_BILINEAR 2 - None 3 - TX_BICUBIC 4 - TX_ADD_DETAIL 5 - TX_MODULATE_DETAIL 6 - TX_BILINEAR_GEQUAL 7 - TX_BILINEAR_LEQUAL 8 - TX_BICUBIC_GEQUAL 9 - TX_BICUBIC_LEQUAL
Int	4	Repetition type: 0 - TX_REPEAT 1 - TX_CLAMP 2 - (Obsolete)
Int	4	Repetition type in u direction (See Above)
Int	4	Repetition type in v direction (See Above)
Int	4	Modify flag (for internal use)
Int	4	x Pivot point for rotating textures
Int	4	y Pivot point for rotating textures
Int	4	Environment type: 0 - TV_MODULATE 1 - TV_BLEND 2 - TV_DECAL 3 - TV_COLOR
Int	4	TRUE if intensity pattern to be loaded in alpha with white in color.
Int	8 * 4	8 words of spare.
Double	8	Real world sizeu for Floating pt. databases.
Double	8	Real world sizev for Floating pt. databases.
Int	4	Code for origin of imported texture.
Int	4	Kernel version number.
Int	4	Internal Format type: 0 - default 1 - TX_I_12A_4 2 - TX_IA_8 3 - TX_RGB_5 4 - TX_RGBA_4 5 - TX_IA_12 6 - TX_RGBA_8 7 - TX_RGBA_12 8 - TX_I_16 (shadow mode only) 8 - TX_RGB_12
Int	4	External Format type: 0 - default 1 - TX_PACK_8 2 - TX_PACK_16

Int	4	Boolean TRUE if using following 8 floats for MIPMAP kernel.
Float	8 * 4	8 Floats for kernel of separable symmetric filter.
Int	4	Boolean if TRUE send:
Float	4	LOD0 for TX_CONTROL_POINT
Float	4	SCALE0 for TX_CONTROL_POINT
Float	4	LOD1 for TX_CONTROL_POINT
Float	4	SCALE1 for TX_CONTROL_POINT
Float	4	LOD2 for TX_CONTROL_POINT
Float	4	SCALE2 for TX_CONTROL_POINT
Float	4	LOD3 for TX_CONTROL_POINT
Float	4	SCALE3 for TX_CONTROL_POINT
Float	4	LOD4 for TX_CONTROL_POINT
Float	4	SCALE4 for TX_CONTROL_POINT
Float	4	LOD5 for TX_CONTROL_POINT
Float	4	SCALE5 for TX_CONTROL_POINT
Float	4	LOD6 for TX_CONTROL_POINT
Float	4	SCALE6 for TX_CONTROL_POINT
Float	4	LOD7 for TX_CONTROL_POINT
Float	4	SCALE7 for TX_CONTROL_POINT
Float	4	clamp
Int	4	magfilteralpha: 0 = TX_POINT 1 = TX_BILINEAR 2 = None 3 = TX_BICUBIC 4 = TX_SHARPEN 5 = TX_ADD_DETAIL 6 = TX_MODULATE_DETAIL 7 = TX_BILINEAR_GEQUAL 8 = TX_BILINEAR_LEQUAL 9 = TX_BICUBIC_GEQUAL 10 = TX_BIBICUBIC_LEQUAL
Int	4	magfiltercolor: 0 = TX_POINT 1 = TX_BILINEAR 2 = None 3 = TX_BICUBIC 4 = TX_SHARPEN 5 = TX_ADD_DETAIL 6 = TX_MODULATE_DETAIL 7 = TX_BILINEAR_GEQUAL 8 = TX_BILINEAR_LEQUAL 9 = TX_BICUBIC_GEQUAL 10 = TX_BIBICUBIC_LEQUAL
Float	22*4	spare
Int	4	Boolean TRUE if using next 5 integers for Detail Texture.
Int	4	J argument for TX_DETAIL.

Int	4	K argument for TX_DETAIL
Int	4	M argument for TX_DETAIL
Int	4	N argument for TX_DETAIL
Int	4	Scramble argument for TX_DETAIL
Int	4	Boolean TRUE if using next for floats for TX_TILE.
Float	4	Lower left u value for TX_TILE.
Float	4	Lower left v value for TX_TILE
Float	4	Upper right u value for TX_TILE
Float	4	Upper right v value for TX_TILE
Int	160 * 4	spare
Char	512 * 1	Comments.

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.

10 Integer Record Formats

SOFTWARE SYSTEMS WILL NO LONGER SUPPORT INTEGER FORMAT VERTEX COORDINATES AS OF VERSION 14.0 OF FLIGHT. Version 14 of MultiGen (and version 13 of ModelGen) will automatically convert each integer data base read to floating point format.

Level of Detail Integer Record Format (OBSOLETE)

Data Type	Length (Bytes)	Description
Int	2	Opcode = 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 - defined by real time
Int	2	Special effects ID 2 - defined by real time
Bool	4	Flags (bits, from left to right) 0 = Use previous slant range 1 = SPT flag: set to 0 for replacement LOD, 1 for additive LOD 2 = Freeze center (don't recalculate) 3-31 Spare
Int	12	Center coordinate of LOD block
Int	4*14	Spare

Degree of Freedom Integer Record Format (OBSOLETE)

Data Type	Length (Bytes)	Description
Int	2	Opcode = 13
Int	2	Length of the record
Char	8	7 char ASCII ID; 0 terminates
Int	4	Origin of the DOF's local coordinate system;l x coordinate
Int	4	Origin of the DOF's local coordinate system; y coordinate
Int	4	Origin of the DOF's local coordinate system; z coordinate
Int	4	Point on the x-axis of the DOF's local coordinate system; x coordinate

Int	4	Point on the x-axis of the DOF's local coordinate system; y coordinate
Int	4	Point on the x-axis of the DOF's local coordinate system; z coordinate
Int	4	Point in xy plane of the DOF's local coordinate system; x coordinate.
Int	4	Point in xy plane of the DOF's local coordinate system; y coordinate.
Int	4	Point in xy plane of the DOF's local coordinate system; z coordinate.
Int	4	Minimum z value with respect to the local coordinate system.
Int	4	Current z value with respect to the local coordinate system.
Int	4	Maximum z value with respect to the local coordinate system.
Int	4	Increment in z.
Int	4	Minimum y value with respect to the local coordinate system.
Int	4	Current y value with respect to the local coordinate system.
Int	4	Maximum y value with respect to the local coordinate system.
Int	4	Increment in y.
Int	4	Minimum x value with respect to the local coordinate system.
Int	4	Current x value with respect to the local coordinate system.
Int	4	Maximum x value with respect to the local coordinate system.
Int	4	Increment in x.
Int	4	Minimum azimuth (rotation about the x-axis).
Int	4	Current azimuth
Int	4	Maximum azimuth .
Int	4	Increment in azimuth
Int	4	Minimum increment (rotation about the y-axis).
Int	4	Current increment
Int	4	Maximum increment .
Int	4	Increment in increment
Int	4	Minimum twist (rotation about the z-axis).
Int	4	Current twist
Int	4	Maximum twist.
Int	4	Increment in twist .
Int	4	Minimum z scale (about local origin).
Int	4	Current z scale (about local origin).
Int	4	Maximum z scale (about local origin).
Int	4	Increment for scale in z.
Int	4	Minimum y scale (about local origin).
Int	4	Current y scale (about local origin).
Int	4	Maximum y scale (about local origin).
Int	4	Increment for scale in y.
Int	4	Minimum x scale (about local origin).
Int	4	Current x scale (about local origin).
Int	4	Maximum x scale (about local origin).
Int	4	Increment for scale in x.

Vertex Record Integer Format (OBSOLETE)

Record Type	Data Type	Length (Bytes)	Description
Absolute	Int	2	Opcode = 7
Vertex	Int	2	Length of the record
	Int	4	x coordinate
	Int	4	y coordinate
	Int	4	z coordinate
	Float	8	*Optional texture (u, v)
.....			
Shaded	Int	2	Opcode = 8
Vertex	Int	2	Length of the record
	Int	4	x coordinate
	Int	4	y coordinate
	Int	4	z coordinate
	Int	1	Hard edge flag
	Int	1	Don't touch normal when shading flag.
	Int	2	Vertex color
	Float	8	*Optional texture (u, v)
.....			
Normal	Int	2	Opcode = 9
Vertex	Int	2	Length of the record
	Int	4	x coordinate
	Int	4	y coordinate
	Int	4	z coordinate
	Int	1	Hard edge flag
	Int	1	Don't touch normal when shading flag.
	Int	2	Vertex color
	Int	12	Vertex normal, scaled * 2**30
	Float	8	*Optional texture (u, v)

In an integer database, vertex records are stored after the polygon record to which they belong. Each record contains an opcode, followed by the coordinates, and other optional fields. Check the length of each vertex record to determine if the optional texture u,v field is included.

Bounding Box Integer Formats (OBSOLETE)

Bounding Box	Int	2	Opcode = 51
Int	Int	2	Length of the record
	Int	12	x, y, z of lowest corner
	Int	12	x, y, z of highest corner

Eyepoint Position Integer Formats (OBSOLETE)

Record Type	Data Type	Length (Bytes)	Description
Eyepoints	Int	2	Opcode = 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 Fly Through
	Float	4	Pitch of Fly Through
	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		
Eyepoint 2	Same as Last Position		
Eyepoint 3	Same as Last Position		
Eyepoint 4	Same as Last Position		
Eyepoint 5	Same as Last Position		
Eyepoint 6	Same as Last Position		
Eyepoint 7	Same as Last Position		
Eyepoint 8	Same as Last Position		
Eyepoint 9	Same as Last Position		

11 Flightspec Index