

Analysis of Flights of Fantasy

Jed Margolin

{Company's} Mr. {Person's} Prior Art References

Flights of Fantasy, Christopher Lampton (The Waite Group 1993). Discusses computer graphics algorithms for the purpose of generating perspective scenery for flight simulators. Includes source code examples.

Flights of Fantasy – Programming 3-D Video Games in C++ by Christopher Lampton (ISBN: 1878739182). [{Ref. 7a}](#) Mr. {Person's} characterization of the book does not do the book justice. This is an excellent book that teaches practical computer graphics including 3D graphics. Only a small part of it is about generating perspective scenery. Scenery is taught in **Chapter 13: Fractal Mountains and Other Types of Scenery** (Pages 439 – 455). That's 17 pages out of a 556 page book. From page 440:

So far, though, we've populated our world with very few objects-a cube here, a pyramid there. How do we build up something that looks like the scenery in the real world?

Landscaping the Imagination

Building scenery for a flight simulator is a tedious task, but the results can be worthwhile. So far, we've given you no tools in this book for this task other than ASCII text files that can hold object descriptions and that can be edited with an ASCII text editor. Ambitious readers, however, may want to write their own scenery- and object-design utility that uses the mouse, the joystick, or the keyboard to drag polygons around and to build buildings, mountains, and other pieces of the landscape. You should be aware, though, that such programs are difficult to write, with much of the difficulty lying in the user interface itself.

Without such a utility, you have two alternatives for developing scenery for your flight simulators: edit the ASCII object descriptors by hand or write custom programs for generating specific types of scenery. In this chapter, we'll give you some tips about both methods.

Lampton then teaches the use of fractals for generating perspective scenery. If {Company} generated fractal terrain instead of using the DTED the FAA would not have certified it. And the system would also have been useless for synthetic vision.

The Lampton Flight Simulator is a Simulator. It does not contain the elements needed to fulfill FAA's definition of synthetic vision. As previously discussed, according to FAA:

Synthetic vision means a computer-generated image of the external scene topography from the perspective of the flight deck that is derived from aircraft attitude, high-precision navigation solution, and database of terrain, obstacles and relevant cultural features.

The Lampton Flight Simulator does not use the DTED and is not used with a real, physical aircraft.

However, there is information relevant to the DTED. In **Chapter Eight Polygon-Fill Graphics** it teaches the same method of storing polygons as is explained in my article **Polygon Databases - Is a Digital Terrain Elevation Database (DTED) really a Polygon Database?** [{Ref. 7b}](#)

Lampton teaches the use of a Point List (**vertex type *vertex; // Array of vertices in object**) and a Face List (**polygon_type *polygon; // List of polygons in object**) because (page 269):

It would be wasteful to store two separate lists of vertices, so we've defined one of these lists (the one in the polygon structure) as a list of pointers that point at the vertex descriptors in the list maintained by the object structure. This concept is illustrated in Figure 8-4.

Figure 8-4 makes it clearer than the program snippet. The following is from Lampton CHAPTER EIGHT POLYGON-FILL GRAPHICS (page 269):

The Object-Type Structure

Now we need a structure that will describe an object made up of polygons, similar to the *shape_type* structure that we used in Chapter 7 for storing wireframe shapes. Here's the structure definition:

```
struct    object_type {
    int    number_of_vertices;    // Number of vertices in object
    int    number_of_polygons;    // Number of polygons in object
    int    x,y,z;                // World coordinates of
                                // object's Local origin
    polygon_type *polygon;        // List of polygons in object
    vertex_type *vertex;          // Array of vertices in object
    int    backface_removal;      // Do we want backface removal?
};
```

You'll notice that much of this structure is redundant. We've seen some of these same fields in the *polygon_type* structure. Why do we need a list of vertices in both the polygon structure and the object structure? The reason is that sometimes we'll want to treat a *polygon* as a list of vertices; at other times we'll want to treat an object as a list of vertices. It would be wasteful to store two separate lists of vertices, so we've defined one of these lists (the one in the polygon structure) as a list of pointers that point at the vertex descriptors in the list maintained by the object structure. This concept is illustrated in Figure 8-4.

The `backface_removal` field might look mysterious to you. We'll discuss this field momentarily.

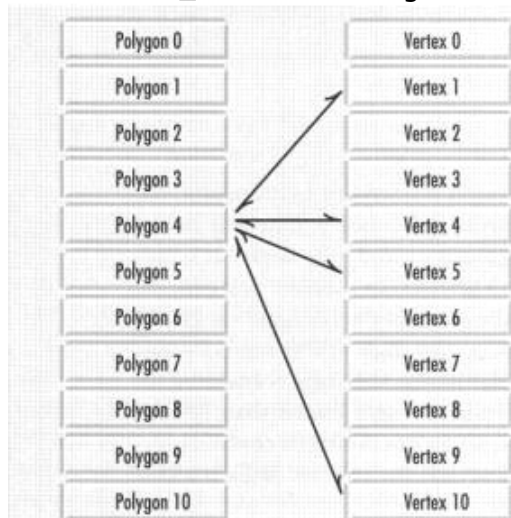


Figure 8-4
The pointers in the polygon list, pointing to the vertices in the vertex list

Lampton also discusses the basics of Polygon Smoothing (page 535):

CHAPTER SIXTEEN THREE-DIMENSIONAL FUTURE

Polygon Smoothing

No amount of light sourcing can turn a polygon-fill object into a completely realistic representation of an actual object. The problem is that real objects, unlike polygon-fill objects, have curves. With some exceptions (such as crystalline structures), nature isn't made of polygons. There are techniques, however, that can turn a surface made out of polygons into a smoothly curved surface, at least on the computer screen. The most popular of these techniques are *Gouraud* and *Phong* shading.

Gouraud shading is the simpler of the two, though it produces slightly less realistic results than Phong shading. Instead of filling polygons with a solid color based on the angle of that surface relative to a light source, Gouraud shading interpolates the color of each pixel on each scan line inside the polygon, based not only on the light-sourced color of the polygon but on the colors of adjacent polygons as well and the distance of the pixel from the edges of the polygon. Pixels near the center of the polygon are given the color that all of the pixels in the polygon would be given using the simple light sourcing techniques described earlier, but pixels toward the edge of the polygon have colors closer to those of adjacent polygons. Phong shading uses much the same technique, though the calculations used to determine the colors of the pixels are a bit more complex.

Compared to ordinary polygon-fill graphics, Gouraud and Phong graphics require a great many computations and are therefore quite slow. At present, it's not likely that a straightforward implementation of either technique could be used in a real-time flight simulator. However, at least one popular three-dimensional game - *Links*, the golf simulation from Access Software - uses

533

FLIGHTS OF FANTASY

rendering techniques that look suspiciously close to either Phong or Gouraud shading. This is made possible by the fact that *Links* is not a real-time simulation. The player watches each scene drawn on the display over a period of seconds. Such slow rendering would be unacceptable in a flight simulator, where the out-the-window view is constantly changing and at least a dozen frames a second are required to make the animation seem smooth. But in *Links*, the view of the golf course only changes when the ball is hit and therefore remains static for long periods of time. And the high quality of the graphics is well worth the amount of time required to produce them.

The results produced by these polygon-smoothing techniques are a dramatic improvement over the polygon-fill techniques we've used in this book. Will it be possible in the future to use them in a flight simulator? Almost certainly. In fact, simplified versions of these techniques have already found their way into several recent flight simulators.

The following is what is in the Lampton book. The first is the Table of Contents. The second is a more detailed list of the contents.

Flights of Fantasy – Programming 3-D Video Games in C++, Christopher Lampton, The Waite Group, 1993

Table of Contents

Chapter 1: A Graphics Primer	1
Chapter 2: The Mathematics of Animation	25
Chapter 3: Painting in 256 Colors	61
Chapter 4: Making It Move	107
Chapter 5: Talking to the Computer	131
Chapter 6: All Wired Up	173
Chapter 7: From Two Dimensions to Three	225
Chapter 8: Polygon-Fill Graphics	263
Chapter 9: Faster and Faster	305
Chapter 10: Hidden Surface Removal	333
Chapter 11: Polygon Clipping	367
Chapter 12: The View System	403
Chapter 13: Fractal Mountains and Other Types of Scenery	437
Chapter 14: Sound Programming	457
Chapter 15: The Flight Simulator	477
Chapter 16: The Three-Dimensional Future	529
Appendix A : Flying the Flights of Fantasy Flight Simulator	541
Appendix B: Books on Three-Dimensional Animation	547

Index: 549

Contents Preface viii Chapter 1: A Graphics Primer 3 The Descendants of Flight Simulator 4 Graphics and Games 5 Text vs. Graphics 6 The Programmer's View 7 Inside the VGA 8 IBM Graphics Improve 9 Video Modes Galore 10 Resolution and Color Depth 10 The Memory Connection 12 Bits, Bytes, and Binary 14 Memory Addresses 15 Pointing at Memory 16 Near and Far 11 How Bitmaps Work 18 A Two-Color Bitmap 19 Mode 13h Memory 20 The Color Palette 20 Programming the VGA 21 Chapter 2: The Mathematics of Animation 27 Cartesian Coordinates 28 The Cartesian Plane 29 Geometry Meets Algebra 31 Coordinates on the Computer 33 From Coordinates to Pixel Offsets 34 Moving the Origin 35 Addressing Multiple Pixels 36	Accessing the Video Bios 72 Restoring the Video Mode 74 More about Bitmaps 75 Finding the Bitmap 76 Drawing Pictures in Memory 77 Clearing the Screen 78 Lots and Lots of Colors 80 Setting the Palette 81 The Default Palette 83 Storing Bitmaps 85 Compressing Bitmaps 86 Graphics File Formats 88 OOP PCX File Loader 88 Inside a PCX File 89 The PCX structure 90 PCX Limitations 91 The PCX Loader 92 Reading the Bitmap and Palette 93 A PCX Viewer 96 Compiling and Running the Program 97 Making It Small Again 98 The Compression Function 99 Decompressing the Data 104 Building the Program 105 Chapter 4: Making It Move 109 Motion Pictures, Computer-Style 110 Bitmaps in Motion 112 A Structure for Sprites 113 Building a Sprite Class 114
--	---

<p>Into the Third Dimension 37</p> <p>Shapes, Solids, and Coordinates 40</p> <ul style="list-style-type: none"> Three-Dimensional Vertices 42 Graphing Equations 43 Solving the Equation 44 Fractals 46 <p>Transformations 46</p> <ul style="list-style-type: none"> Translating and Scaling 47 Rotating on X, Y, and Z 49 Doing What Comes Virtually 49 Degrees vs. Radians 51 Rotation Formulas 52 <p>Matrix Algebra 54</p> <ul style="list-style-type: none"> Building Matrices 54 Multiplying Matrices 55 Three Kinds of Rotation Matrices 57 <p>Floating-Point Math 58</p> <p>Chapter 3: Painting in 256 Colors 63</p> <p>Color Depth vs. Resolution 64</p> <p>Getting in the Mode 64</p> <p>Doing It in Assembly Language 65</p> <ul style="list-style-type: none"> Assembly Language Pointers 67 Getting It into a Register 67 Assembly Language Odds and Ends 69 Passing Parameters to Assembly Language Procedures 70 	<ul style="list-style-type: none"> A Simple Sprite Drawing Routine 116 Transparent Pixels 117 <ul style="list-style-type: none"> Speeding Things Up 118 Putting Sprites 118 <ul style="list-style-type: none"> Erasing Sprites 119 Grabbing Sprites 120 The Walkman Cometh 122 <ul style="list-style-type: none"> Buffering the Screen 122 Constructing a Sprite 124 Setting Up the Walking Sequence 124 Looping Through the Sequence 125 Walkman Struts His Stuff 126 Building The Program 129 <p>Chapter 5: Talking to the Computer 133</p> <p>Programming The joystick 134</p> <ul style="list-style-type: none"> Analog vs. Digital 135 Any Old Port 137 The Mathematics of Truth 138 Masking Binary Digits 140 Other Bitwise Operations 141 Decoding the Gameport Byte 141 The Button-Reading Procedure 143 A Bit of Information 144 Timing the Status Bit 144 Counting Down 145 The Joystick-Reading Procedure 145
<p>Calibrating the Joystick 147</p> <p>Not an Input Device Was Stirring...Except for the Mouse 147</p> <ul style="list-style-type: none"> Listening to the Mouse's Squeak 148 Button-Down Mice 150 Mickeying with the Mouse 151 <p>The Mouse Position Function 152</p> <ul style="list-style-type: none"> All Keyed Up 153 The BIOS Keyboard Routines 153 <p>Managing Events 156</p> <p>The Event Manager Functions 158</p> <ul style="list-style-type: none"> The Master of Events 159 Inside the Event Manager 160 <p>Walkman Returns 164</p> <ul style="list-style-type: none"> Entering a New Dimension 172 Building the Program 172 <p>Chapter 6: All Wired Up 175</p> <p>Snapshots of the Universe 176</p> <ul style="list-style-type: none"> Bitmap Scaling 176 Rendering Techniques 177 <p>Wireframe and Polygon-Fill 178</p> <ul style="list-style-type: none"> Vertex Descriptors 180 <p>A Two-Dimensional Wireframe Package 182</p> <ul style="list-style-type: none"> Bresenham's Algorithm 186 <p>A Line-Drawing Routine In C++ 189</p> <ul style="list-style-type: none"> Testing the Line-Drawing Function 191 Drawing Random Lines 192 A Line-Drawing Function in Assembler 194 <p>Drawing Shapes 196</p> <ul style="list-style-type: none"> Creating Shape Data 199 A Shape-Drawing Program in C++ 200 <p>Transforming Shapes 203</p>	<p>Chapter 8: Polygon-Fill Graphics 265</p> <p>The Polygon Descriptor 266</p> <ul style="list-style-type: none"> The Vertex Type Structure 267 The Polygon_Type Structure 268 The Object_Type Structure 269 <p>New Structures 270</p> <p>Polygon Drawing 270</p> <ul style="list-style-type: none"> The Polygon-Drawing Algorithm 273 The Polygon-Drawing Function 275 Getting to the Top 276 From Start to Finish 277 Drawing the Polygon 278 Bresenham Does His Thing 279 The Four Subsections 279 Counting Off the Pixels 281 Looping Until Y Increments 281 Drawing the Line 283 Cleaning Up 284 The Complete Function 284 <p>Manipulating Polygons 293</p> <ul style="list-style-type: none"> The Transform() Function 293 The Project() Function 294 Backface Removal 295 The Draw_Object() Function 296 Limitations of Backface Removal 297 <p>The Polygon-Fill Display Program 297</p> <ul style="list-style-type: none"> ASCII Object Descriptors 297 Reading the Data File 299 The Polygon Demonstration Program 302 <p>Chapter 9: Faster and Faster 307</p> <p>Knowing Where to Optimize 308</p>

<ul style="list-style-type: none"> Local Coordinates 203 Translating 205 Scaling 208 Rotating 212 Doing It with Matrices 217 Chapter 7: From Two Dimensions to Three 227 The Z Coordinate 228 Cubic Universes 229 From World Coordinates to Screen Coordinates 231 Storing the Local X, Y, and Z Coordinates 233 Creating a Three-Dimensional Shape 234 Projection and Perspective 236 The Project() Function 239 Transformation in Three Dimensions 241 The Global Transformation Matrix 243 The Scale() Function 246 The Translate() Function 247 The Rotate() Function 248 The Transform() Function 250 The Draw-Shape() Function 251 <ul style="list-style-type: none"> Drawing a Cube 252 Animating the Cube 254 <ul style="list-style-type: none"> The Cube-Rotating Program 255 Drawing a Pyramid 257 Drawing a Letter W 258 	<ul style="list-style-type: none"> Nested Loops 308 <ul style="list-style-type: none"> The Innermost Loop 309 Profiling Your Code 310 <ul style="list-style-type: none"> Executing Tprof 311 Reducing to Zero 312 Profiling for Real 313 Integer Fixed-Point Arithmetic 314 <ul style="list-style-type: none"> Working with Fixed-Point Arithmetic 315 Adjusting the Decimal Position 316 Is It Worth It? 318 Using Look-Up Tables to Avoid Calculations 319 Unrolling the Loop 324 <ul style="list-style-type: none"> The Mechanics of the Loop 325 Putting It All to Work 326 Chapter 10: Hidden Surface Removal 335 The Problem 336 The Painter's Algorithm 337 <ul style="list-style-type: none"> Getting the Order Right 339 The Five Tests 343 Mutual Overlap 346 Time Considerations 347 The Z-Buffer Algorithm 348 Back to Depth Sorting 350 Drawing the Polygon List 366
---	---

<ul style="list-style-type: none"> Chapter 11: Polygon Clipping 369 The View Volume 370 <ul style="list-style-type: none"> Clipping Against the View Volume 325 The Sutherland-Hodgman Algorithm 376 Four Types of Edge 371 <ul style="list-style-type: none"> The Point-Slope Equation 300 The Clipped Polygon Structure 382 The Clipping Function(s) 385 <ul style="list-style-type: none"> The Front of the View Volume 306 Type One Edge 387 Type Two Edge 387 Type Three Edged 387 Type Four Edge 388 The Z_clip() Function 389 The Rest of the View Volume 390 Finishing the Clipping 392 Chapter 12: The View System 405 Moving Around Inside the Computer 406 <ul style="list-style-type: none"> The Stationary Viewer vs. the Moving Viewer 406 Making the Viewer an Object in the World 407 Rotating the Viewer 409 Aligning the Universe 410 The Alignview() Function 410 Relative Motion 412 Aligning Rotation 414 The Transform() Function 415 The View Class 417 The Setview() Function 418 The Setworld() Function 419 	<ul style="list-style-type: none"> Fractal Mountain Ranges 448 <ul style="list-style-type: none"> The Mountain-Drawing Program 449 The Lineseg() Function 451 The Mountain Program 452 Three-Dimensional Fractals 454 Chapter 14: Sound Programming 459 What Is Sound? 460 <ul style="list-style-type: none"> The Parts of a Wave 460 Complex Sound Waves 461 Recording Sound 463 <ul style="list-style-type: none"> Digital Sound 464 The Sound Blaster DAC 466 <ul style="list-style-type: none"> Digital Sound Effects 467 The FM Synthesizer 468 <ul style="list-style-type: none"> Inside the FM Chip 469 The Sound Blaster Interface 469 <ul style="list-style-type: none"> Initializing the FM Chip 469 Setting Up a Sound 470 Setting Up the Envelopes 472 Turning On the Engine 474 Chapter 15: The Flight Simulator 479 Interfacing to the View System 480 <ul style="list-style-type: none"> Animating the View 482 Flight Simulators 483 The Flight Model 404 The Flight of an Airplane 485 <ul style="list-style-type: none"> Thrust 487 Lift 489 Controlled Flight 492
--	---

<p>The Update() Function 420 The Draw_Horizon Function 421 The Horizon Problem 421 How Horizons Work 422 The Function Code 424 Remapping the Angles 425 Rotating the Horizon 426 Creating the Sky and Ground Polygons 427 The Display() Function 432 A New Backface Removal Function 434</p> <p>Chapter 13: Fractal Mountains and Other Types of Scenery 439 Landscaping the Imagination 440 Using Object Files 440 How an Object File Is Organized 441 Using a Program to Generate Object Files 443 Right-Handed or Left-Handed 444 Fractal Realism 444 Self-Similarity 445 Making Use of Fractals 446 Recursive Graphics 447</p>	<p>Control Surfaces 492 The State Vector 494 From Input to Flight Model 495 The Getcontrols() Function 495</p> <p>Chapter 16: The Three-Dimensional future 531 Light Sourcing 532 Polygon Smoothing 533 Ray Tracing 534 Implementing A Ray Tracer 535 Bitmapped Animation 536 Do-It-Yourself Bitmaps 539 The Ultimate Flight Simulator 540</p> <p>Appendix A: Flying the Flights of Fantasy Flight Simulator 541 Appendix B: Books on Three-Dimensional Animation 547</p> <p>Index: 549</p>
---	--

References

Ref. 7a - **Flights of Fantasy – Programming 3-D Video Games in C++** by Christopher Lampton (ISBN: 1878739182).

Ref. 7b - **Polygon Databases - Is a Digital Terrain Elevation Database (DTED) really a Polygon Database?**, Jed Margolin, <http://www.jmargolin.com/patents2/pilotrefs/PolygonDatabases2.pdf>
