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; // Array of vertices in object) and a Face List (*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:

```c
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.
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 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
Into the Third Dimension 37
Shapes, Solids, and Coordinates 40
Three-Dimensional Vertices 42
Graphing Equations 43
Solving the Equation 44
Fractals 46
Transformations 46
Translating and Scaling 47
Rotating on X, Y, and Z 49
Doing What Comes Virtually 49
Degrees vs. Radians 51
Rotation Formulas 52
Matrix Algebra 54
Building Matrices 54
Multiplying Matrices 55
Three Kinds of Rotation Matrices 57
Floating-Point Math 58

Chapter 3: Painting in 256 Colors 63
Color Depth vs. Resolution 64
Getting in the Mode 64
Doing It in Assembly Language 65
Assembly Language Pointers 67
Getting It into a Register 67
Assembly Language Odds and Ends 69
Passing Parameters to Assembly Language Procedures 70
Calibrating the Joystick 147
Not an Input Device Was Stirring...Except for the Mouse 147
Listening to the Mouse's Squeak 148
Button-Down Mice 150
Mickeying with the Mouse 151
The Mouse Position Function 152
All Keyed Up 153
The BIOS Keyboard Routines 153
Managing Events 156
The Event Manager Functions 158
The Master of Events 159
Inside the Event Manager 160
Walkman Returns 164
Entering a New Dimension 172
Building the Program 172

Chapter 6: All Wired Up 175
Snapshots of the Universe 176
Bitmap Scaling 176
Rendering Techniques 177
Wireframe and Polygon-Fill 178
Vertex Descriptors 180
A Two-Dimensional Wireframe Package 182
Bresenham's Algorithm 186
A Line-Drawing Routine In C++ 189
Testing the Line-Drawing Function 191
Drawing Random Lines 192
A Line-Drawing Function in Assembler 194
Drawing Shapes 196
Creating Shape Data 199
A Shape-Drawing Program in C++ 200
Transforming Shapes 203

Chapter 8: Polygon-Fill Graphics 265
The Polygon Descriptor 266
The Vertex Type Structure 267
The Polygon_Type Structure 268
The Object_Type Structure 269
New Structures 270
Polygon Drawing 270
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
Manipulating Polygons 293
The Transform() Function 293
The Project() Function 294
Backface Removal 295
The Draw_Object() Function 296
Limitations of Backface Removal 297
The Polygon-Fill Display Program 297
ASCII Object Descriptors 297
Reading the Data File 299
The Polygon Demonstration Program 302

Chapter 9: Faster and Faster 307
Knowing Where to Optimize 308
### Chapter 7: From Two Dimensions to Three

- The Z Coordinate
- Cubic Universes
- From World Coordinates to Screen Coordinates
- Storing the Local X, Y, and Z Coordinates
- Creating a Three-Dimensional Shape
- Projection and Perspective
- The Project() Function
- Transformation in Three Dimensions
- The Global Transformation Matrix
- The Scale() Function
- The Translate() Function
- The Rotate() Function
- The Transform() Function
- The Draw-Shape( Function
- Drawing a Cube
- The Cube-Rotating Program
- Drawing a Pyramid
- Drawing a Letter W
- Fractal Mountain Ranges
- The Mountain-Drawing Program
- The Lineseg( ) Function
- The Mountain Program
- Three-Dimensional Fractals

### Chapter 8: Nested Loops

- The Innermost Loop
- Profiling Your Code
- Executing Tprof
- Reducing to Zero
- Profiling for Real
- Integer Fixed-Point Arithmetic
- Working with Fixed-Point Arithmetic
- Adjusting the Decimal Position
- Is It Worth It?
- Using Look-Up Tables to Avoid Calculations
- Unrolling the Loop
- The Mechanics of the Loop
- Putting It All to Work

### Chapter 9: Transformation in Three Dimensions

- The Z-Coordinate
- Cubic Universes
- From World Coordinates to Screen Coordinates
- Storing the Local X, Y, and Z Coordinates
- Creating a Three-Dimensional Shape
- Projection and Perspective
- The Project() Function
- Transformation in Three Dimensions
- The Global Transformation Matrix
- The Scale() Function
- The Translate() Function
- The Rotate() Function
- The Transform() Function
- The Draw-Shape( Function
- Drawing a Cube
- The Cube-Rotating Program
- Drawing a Pyramid
- Drawing a Letter W
- Fractal Mountain Ranges
- The Mountain-Drawing Program
- The Lineseg( ) Function
- The Mountain Program
- Three-Dimensional Fractals

### Chapter 10: Hidden Surface Removal

- The Problem
- The Painter's Algorithm
- Getting the Order Right
- The Five Tests
- Mutual Overlap
- Time Considerations
- The Z-Buffer Algorithm
- Back to Depth Sorting
- Drawing the Polygon List

### Chapter 11: Polygon Clipping

- The View Volume
- Clipping Against the View Volume
- The Sutherland-Hodgman Algorithm
- Four Types of Edge
- The Point-Slope Equation
- The Clipped Polygon Structure
- The Clipping Function(s)
- The Front of the View Volume
- Type One Edge
- Type Two Edge
- Type Three Edged
- Type Four Edge
- The Z_clip() Function
- The Rest of the View Volume
- Finishing the Clipping

### Chapter 12: The View System

- Moving Around Inside the Computer
- The Stationary Viewer vs. the Moving Viewer
- Making the Viewer an Object in the World
- Rotating the Viewer
- Aligning the Universe
- The Alignview() Function
- Relative Motion
- Aligning Rotation
- The Transform() Function
- The View Class
- The Setview() Function
- The Setworld( ) Function

### Chapter 13: Sound Programming

- What Is Sound?
- The Parts of a Wave
- Complex Sound Waves
- Recording Sound
- Digital Sound
- The Sound Blaster DAC
- Digital Sound Effects
- The FM Synthesizer
- Inside the FM Chip
- The Sound Blaster Interface
- Initializing the FM Chip
- Setting Up a Sound
- Setting Up the Envelopes
- Turning On the Engine

### Chapter 14: Sound Programming

- Fractal Mountain Ranges
- The Mountain-Drawing Program
- The Lineseg( ) Function
- The Mountain Program
- Three-Dimensional Fractals

### Chapter 15: The Flight Simulator

- Interfacing to the View System
- Animating the View
- Flight Simulators
- The Flight Model
- The Flight of an Airplane
- Thrust
- Lift
- Controlled Flight
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

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

Appendix A: Flying the Flights of Fantasy Flight Simulator 541
Appendix B: Books on Three-Dimensional Animation 547

Index: 549

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/pilotsrefs/PolygonDatabases2.pdf](http://www.jmargolin.com/patents2/pilotsrefs/PolygonDatabases2.pdf)