Chapter 2

On to OpenGL and 3D Computer Graphics

This is the first technical chapter and done right will get students "into" CG. To this end keep in mind that, in addition to the introduction to OpenGL and 3D CG, there's another vital goal to this chapter: to dispel the notion that CG is a field only for people who would cheerfully face a second-order partial differential equation in MMA combat and who write recursive subroutines on the bus home. *It's not*! The average CS student is perfectly capable of an excellent understanding of CG and of creating fine 3D apps provided she is willing to *think a bit in 3D*.

Simply following the chapter linearly from start to finish at a pace that takes most of the class along – you may have to slow down and spend extra time on certain topics but it's worth it for this chapter – should be all that's needed to get the job done.

1. Section 2.1: Introduces the workhorse program `square.cpp`. Students see their first complete OpenGL program and begin trying to relate its syntax to the black square it draws. (Note the use of classical OpenGL. If you have not done so already, you should read the section in the preface about our pedagogical approach to shader-based OpenGL.) This leads to the very important next section.

2. Section 2.2: Introduces the viewing box, world coordinates and explains how a 3D scene is projected to make a 2D image via a synthetic camera process.

   Emphasize the simplicity of the process: objects are drawn in an imaginary viewing box in world space (in case of orthographic

projection) and projected to the box's front, which is like a film; the film is then printed on the OpenGL window part of the screen, which is like paper. *That's it*! There are exactly two spaces involved, world and screen, and one goes from one to the other via a projection.

Some of our peer textbooks, in addition to world and screen coordinates, invoke local coordinates, device coordinates, normalized coordinates and even viewing and projection coordinates. None of these additional systems are necessary to our mind. We go all the way from this chapter to projection transformations and rational primitives toward the end of the book without needing anything more than world and screen systems (it's the KISS principle).

As a general rule, perform all the book experiments in class. For Section 2.2, in particular, ask the students to help you do most of the exercises in class as well.

3. Section 2.3: A benign section on screen coordinates in the OpenGL window. The only point to emphasize is the flip in y-axis from world to screen system.

4. Section 2.4: A chance to do some fun experiments with clipping. Point out that the implementation of clipping might not be as simple as the "clipping knife" of Figure 2.15 would suggest, because the two new vertices of the resulting quad must be computed. This is done inside the pipeline and there's no need to worry about it now, but it's worth noting.

5. Section 2.5: Explains that OpenGL is a state machine and that properties of an object are specified by the states (e.g., color) at its vertices. These vertex properties are spread into the interior of an object by interpolation. This section also explains that OpenGL draws in code order: objects are created by drawing commands as the code is processed linearly from start to finish.

   You might want to expand on interpolation a little (or not at all) depending on the math inclinations of your class. Keep in mind that Chapter 7 down the road is all about convexity and interpolation.

6. Section 2.6: Introduces all the OpenGL drawing primitives. Emphasize that 2D primitives like polygons should be plane and convex, otherwise rendering is unpredictable; Experiment 2.17 illustrates this.

   Observe that polygons and rectangles have been discarded from the latest OpenGL, but, of course, each can be made with triangle strips.

7. Section 2.7: This is where students see for the first time a curve being approximated by straight segments. Get in place the general design principle that all curved objects have to be approximated by

straight and flat ones. Mention the trade-off between the quality of the approximation (i.e., the fineness of the mesh) and complexity,

8. Section 2.8: A first look at 3D with the simple Experiment 2.20. Keep the description of the z-buffer as simple as possible: primitives are processed one by one (in code order), and each pixel making up a primitive has its z-value checked with the current value in its slot in the z-buffer before it is drawn, ...

   The experiments using `helix.cpp` nicely set up perspective projection. Emphasize the similarity to orthographic projection – the difference being only in rays traveling to a point instead of in parallel. A logical consequence is that the shape of the viewing box follows that of parallel rays while that of the frustum convergent ones. Shoot-and-print is similar too for both kinds of projection.

9. Section 2.9: Drawing projects to choose assignments from.

10. Section 2.10: First assemblage of a curved *surface*, a hemisphere, from triangle primitives. Emphasize the do-it-yourself nature of determining the constituent triangles from the equations for the surface. WYSIWYG systems like Maya and Studio MAX do simplify the design process but there's something to be said for first learning to make things "by hand" before applying sophisticated tools.

11. Section 2.11: Goes line by line over the syntax of `square.cpp` – all routine, except that the comparison of core and compatibility versions of OpenGL, in relation to the context-setting statements in the main routine, might be worth dwelling on a bit.

Additional points:

(a) The *Experimenter* software should help run the experiments.

(b) If you come up with interesting modifications of the book experiments or new ones, as almost certainly you will, then you might want to include them in your copy of *Experimenter* for future use. The introduction to *Experimenter* says how to do this.

(c) Remember the mantra: experiment-discuss-repeat. However, if you like to follow the linear structure of a set of lecture slides, then the book figures – available sequentially, one PowerPoint presentation per chapter, at the book's website – might be a starting point to preparing your own.

(d) This is a 3-lecture chapter but do not worry if you go over. It is worth doing well. If the students are making eye contact and not hiding under the table at the end of this chapter, then you've just made a terrific start.

Solution to Exercise 2.4

Change glOrtho(0.0, 100.0, 0.0, 100.0, -1.0, 1.0) to glOrtho(0.0, 200.0, 0.0, 100.0, -1.0, 1.0).

Solution to Exercise 2.5

If only the z coordinate of a point is changed, then perpendicular projection takes it to the same point on the viewing face as before.

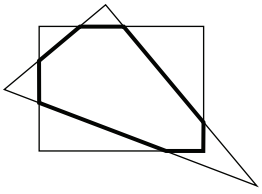

Figure 2.1: Clipping a triangle to a septagon.

Solution to Exercise 2.7

Figure 2.1 shows clipping to 7 sides.

Solution to Exercise 2.11

The color will be "mid-way" between $(1, 0, 0)$ and $(0, 1, 0)$, which is $(\frac{1+0}{2}, \frac{0+1}{2}, \frac{0+0}{2}) = (0.5, 0.5, 0)$, a yellow.

Solution to Exercise 2.24

The corners of the front face are $(-5.0, -5.0, -5.0)$, $(5.0, -5.0, -5.0)$, $(-5.0, 5.0, -5.0)$ and $(5.0, 5.0, -5.0)$. The x and y values of the vertices of the back face are scaled from those on the front by a factor of 20 ($= \mathit{far}/\mathit{near} = 100/5$). These vertices are, therefore, $(-100.0, -100.0, -100.0)$, $(100.0, -100.0, -100.0)$, $(-100.0, 100.0, -100.0)$ and $(100.0, 100.0, -100.0)$.