OpenGL Notes [a]

Stu Pomerantz
smp@psc.edu
http://www.psc.edu/~smp
May 23, 2007

[a]Most material is adapted from: OpenGL ARB, et. al, "The OpenGL Programming Guide", Third Ed., Reading: Addison-Wesley, 1999

Specifying Geometric Primitives

- Primitives are: points, lines, triangles, quads (quadrilaterals)
- Primitives are specified by their **vertices**
- Usually many primitives are specified at once.
  - For instance, the surface of a car may be composed of many triangles
- From an efficiency standpoint the, say, 5000 triangles that make up a surface should almost aways be specified at once when all are likely to undergo the same geometric transformation (such as rotation).
- There are other kinds of primitives, notably splines, which won't be discussed today.

Specifying Geometric Primitives

- To specify a vertex for a primitive, call glVertex*()
  - For a 2D floating point vertex, `glVertex2f(x,y)`
  - For a 3D floating point vertex, `glVertex3f(x,y,z)`
- glVertex*() calls are *always* enclosed in `glBegin(constant)` and `glEnd()` calls.
  - The value of `constant` specifies the kind of primitives you are making
  - Since each primitive requires a specific number of vertices you can unambiguously specify many primitives inside glBegin() and glEnd()

Specifying Geometric Primitives

- Specify 6 points:

```
glBegin(GL_POINTS) ;
  glVertex2f(0,0) ;
  glVertex2f(1,0) ;
  glVertex2f(1,1) ;
  glVertex2f(2,2) ;
  glVertex2f(3,2) ;
  glVertex2f(3,3) ;
glEnd() ;
```

Specifying Geometric Primitives

- Specify 3 lines:

```
glBegin(GL_LINES) ;
  glVertex2f(0,0) ; // line 0
  glVertex2f(1,0) ;
  glVertex2f(1,1) ; // line 1
  glVertex2f(2,2) ;
  glVertex2f(3,2) ; // line 2
  glVertex2f(3,3) ;
glEnd() ;
```

Specifying Geometric Primitives

- Specify 2 triangles:

```
glBegin(GL_TRIANGLES) ;
  glVertex2f(0,0) ; // triangle 0
  glVertex2f(1,0) ;
  glVertex2f(1,1) ;
  glVertex2f(2,2) ; // triangle 1
  glVertex2f(3,2) ;
  glVertex2f(3,3) ;
glEnd() ;
```

Specifying Geometric Primitives

- Specify 5 connected line segments:

```
glBegin(GL_LINE_STRIP) ;
  glVertex2f(0,0) ;
  glVertex2f(1,0) ;
  glVertex2f(1,1) ;
  glVertex2f(2,2) ;
  glVertex2f(3,2) ;
  glVertex2f(3,3) ;
glEnd() ;
```

- (0,0) is the first point of the first line and (3,3) is the last point of the last line.

Specifying Geometric Primitives

- Specify a *convex* polygon:

```
glBegin(GL_POLYGON) ;
  glVertex2f(0,-1) ; // home plate for a baseball game
  glVertex2f(1,0) ;
  glVertex2f(1,1) ;
  glVertex2f(-1,1) ;
  glVertex2f(-1,0) ;
glEnd() ;
```

- Loosely speaking, convex means no 'dents' or 'dimples'.

Specifying Geometric Primitives

- Only a few OpenGL calls are valid inside glBegin() and glEnd()
  - can specify colors with glColor()
  - can specify texture coordinates (for texture mapping).

```
// specify a red line
glBegin(GL_LINES) ;
  glColor3f(0,1,0) ;
  glVertex2f(0,0) ;
  glVertex3f(1,1) ;
glEnd() ;
```

Specifying Geometric Primitives

```
// specify a line with a red vertex and a blue vertex
glBegin(GL_LINES) ;
  glColor3f(0,1,0) ;
  glVertex2f(0,0) ;

  glColor3f(0,0,1) ;
  glVertex3f(1,1) ;
glEnd() ;
```

- In the right OpenGL state, the colors on the line will be linearly interpolated from red to blue.

Introduction to OpenGL Buffers

- The end result of a display() is an image on the screen.
- OpenGL stores this image in its *color buffer*
- To make a simple animation:
  1. clear the color buffer to the background color
  2. display primitives (make image in color buffer)
  3. update
  4. goto 1
- Can think of the color buffer as the frame buffer.
- OpenGL has other buffers which will be introduced later.

Introduction to OpenGL Buffers

- To specify the background color, call `glClearColor()`
- glClearColor() takes 4 floats, ranging from 0 to 1, that specify an RGBA color.
- We won't use the alpha value in glClearColor() in this class.

```
// the background color is red.
glClearColor(1,0,0,1) ;
```

Introduction to OpenGL Buffers

- To actually clear the color buffer to the background color specified by glClearColor() call
  `glClear(GL_COLOR_BUFFER_BIT)`
- Note that `GL_COLOR_BUFFER_BIT` is specified since glClear() can be used to clear more buffers than just the color buffer by or-ing constants representing other buffers together then the glClear() call.

The Viewport

- `glViewport(x,y,width,height)` specifies the *lower left* corner and width and height of window that OpenGL is going to draw into.
- These numbers are specified with respect to window that 'contains' the OpenGL context.
- The viewport need not be the entire window, as it is with most demo code.

An Oversimplified Model of OpenGL

geometric primitive

↓

transformation

↓

frame buffer

- Primitive specification has been discussed briefly. It will be revisited.
- Transformation is the next critical step to understand.
- The frame buffer will be revisited also.