OpenGL Notes [a]

Stu Pomerantz
smp@psc.edu
http://www.psc.edu/∼ smp
September 1, 2004

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

OpenGL Geometry

- The fundamental geometric primative in OpenGL is the point. Points are also called **vertices**. A single point is a **vertex**.
- Points are used to build more complex geometry.
- There are many ways to specify points in OpenGL. This most basic is using `glVertex*()` ; For example:
  - 2D: `glVertex2f(1.0, 2.0) ;`
  - 2D: `glVertex2i(4, -1) ;`
  - 3D: `glVertex3f(20,30,40) ;`
  - 3D: `GLfloat p[3]= {1,2,3} ; glVertex3fv(p) ;`

OpenGL Geometry

- Since vertices are used to specify different kinds of geometric objects, OpenGL must be told what kind of object(s) the vertices are making.
- To tell OpenGL the kind of object(s) your vertices make up, enclose `glVertex*()` calls within `glBegin(`*constant*`)` and `glEnd()` commands.
- `glBegin(`*constant*`)` accepts these constants:
  - `GL_POINTS`, `GL_LINES`, `GL_LINE_LOOP`, `GL_LINE_STRIP`
  - `GL_TRIANGLES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN`
  - `GL_QUADS`, `GL_QUAD_STRIP`
  - `GL_POLYGON`

OpenGL Geometry Examples

This code:

```
glBegin(GL_POINTS) ;
  glVertex2i(0,0);
  glVertex2i(1,0) ;
  glVertex2i(1,1) ;
  glVertex2i(0,1) ;
glEnd() ;
```

draws 4 points.

OpenGL Geometry Examples

This code:

```
glBegin(GL_LINES) ;
  glVertex2i(0,0);
  glVertex2i(1,0) ;
  glVertex2i(1,1) ;
  glVertex2i(0,1) ;
glEnd() ;
```

draws 2 lines. The first, from (0,0) to (1,0) and the second from (1,1) to (0,1).

OpenGL Geometry Examples

This code:

```
glBegin(GL_LINE_LOOP) ;
  glVertex2i(0,0);
  glVertex2i(1,0) ;
  glVertex2i(1,1) ;
  glVertex2i(0,1) ;
glEnd() ;
```

draws a square.

OpenGL Geometry Examples

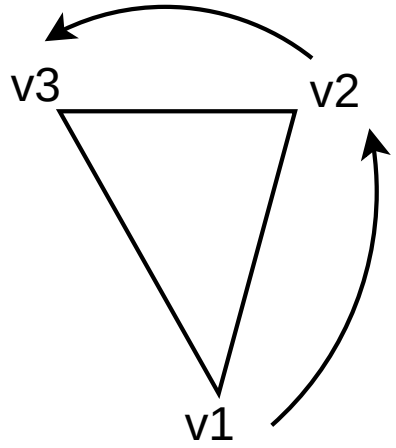
This code:

```
glBegin(GL_QUADS) ;
  glVertex2i(0,0);
  glVertex2i(1,0) ;
  glVertex2i(1,1) ;
  glVertex2i(0,1) ;
glEnd() ;
```

draws a filled square, or quad.

OpenGL Geometry

- The *order* that vertices are specified is important.
- Each `glBegin(`*constant*`)` expects a certain ordering of vertices in order to produce the expected result.
- Polygon (e.g. triangle) vertices are usually specified *counter clockwise*.



OpenGL Geometry

- This direction is called the polygon *winding*.
- The expected winding is counter clockwise, but that can be changed.
- Winding is used for *back face culling* discussed later.

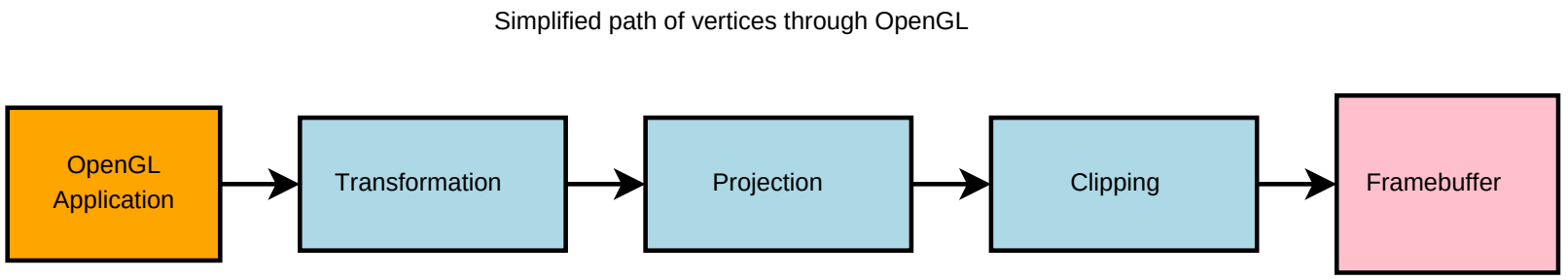
OpenGL Viewport

- The viewport is the rectangular area of the window in which your OpenGL drawing takes place. OpenGL is told about this area using `glViewport(lowerx, lowery, width, height);`
- *lowerx* & *lowery* are the lower left corner of the viewport relative to the lower left corner of the window containing the viewport.
- *width* & *height* are just the width and height of the viewport.
- Usually this only needs to be called if a *reshape* event occurs (e.g. the window size changes).
- Usually put in `glutReshapeFunc();`

OpenGL ClearColor

- The background color of the viewport is specified using `glClearColor() ;`
- For example, to set the clear color to blue: `glClearColor(0.0, 0.0, 1.0, 1.0) ;`
- Notice that the color values are always floating point, and an alpha value (the fourth component) is always specified.
- To reset the color buffer of the viewport to the clear color (say, just before drawing a new frame) you would use: `glClear(GL_COLOR_BUFFER_BIT) ;` at the top of your `display()` function.

The OpenGL Pipeline



- **Transformation** Vertices are rotated and translated into place.
- **Projection** The projection of vertices onto the image plane is calculated.
- **Clipping** Vertices outside the image plane are clipped away.

How each of these steps works has not yet been presented.
Many important details are omitted in this diagram.

Black Box Code

- Unfortunately, the author of your text introduces numerous OpenGL commands without first explaining the mathematics and algorithms behind how they work.
- These functions must essentially be treated as black boxes until later chapters where they are explained.

Black Box Code

```
glMatrixMode(GL_PROJECTION) ;
glLoadIdentity() ;
glOrtho(left, right, bottom, top, near, far) ;
glMatrixMode(GL_MODELVIEW) ;
```

- Tell OpenGL we want to set the projection transformation
- load the identity matrix into the projection transformation
- set up an orthographic projection whose boundaries are *left*, *right*, *bottom* & *top*. Clipping planes are at *near* and *far*.
- Tell OpenGL to go back to its normal mode.

Black Box Code

```
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH ) ;
glEnable(GL_DEPTH_TEST) ;
```

- Tell GLUT to setup up a single buffered, RGB window which has a depth buffer.
- Tell OpenGL to enable depth testing
- depth buffering is used for hidden surface removal. This algorithm will be discussed later.

Black Box Code

```
glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT) ;
```

- Tell OpenGL to clear the color buffer and the depth buffer
- This would typically be done at the top of the `display() ;` function.