Drawing Primitives
Woo, Neider et Al., Chapter 2

Clearing

Pixels are stored in bitplanes

`glClearColor (red, green, blue, alpha)`
Sets the color used to (red, green, blue, alpha)

`glClear(GL_COLOR_BUFFER_BIT)`
Performs the clear operation on one or more buffers at the same time

Colors in RGBA

<table>
<thead>
<tr>
<th><code>glColor3f</code></th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0.0, 0.0, 0.0)</td>
<td>Black</td>
</tr>
<tr>
<td>(1.0, 0.0, 0.0)</td>
<td>Red</td>
</tr>
<tr>
<td>(1.0, 1.0, 0.0)</td>
<td>Yellow</td>
</tr>
<tr>
<td>(0.0, 1.0, 0.0)</td>
<td>Green</td>
</tr>
<tr>
<td>(0.0, 1.0, 1.0)</td>
<td>Cyan</td>
</tr>
<tr>
<td>(0.0, 0.0, 1.0)</td>
<td>Blue</td>
</tr>
<tr>
<td>(1.0, 0.0, 1.0)</td>
<td>Magenta</td>
</tr>
<tr>
<td>(1.0, 1.0, 1.0)</td>
<td>White</td>
</tr>
<tr>
<td>(0.5, 0.5, 0.5)</td>
<td>Gray</td>
</tr>
</tbody>
</table>

Forcing Completion of Drawing

Multiple commands may be gathered in a buffer before execution (efficiency)

`glFlush ()`: Forces execution before the gathering is complete

`glFinish ()`: Forces all commands to be completed. The function does not return until then

Reshape Callback

- Called when the user changes the size of the window
- For simple 2D drawings:

```c
void reshape(int w, int h)
{
    glViewport (0, 0, (GLsizei) w, (GLsizei) h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho2D(0.0, (GLdouble) w, 0.0, (GLdouble) h);
}
```

Drawing Primitives

`glBegin (mode);`
- commands...
`glEnd;`

Mode can be for example:
- `GL_POINTS`: Individual points
- `GL_LINES`: Pairs of vertices used to draw segments
- `GL_TRIANGLES`: Triple of vertices used to draw triangles
- `GL_QUADS`: Quadruples of vertices used to draw quadrilaterals
- `GL_POLYGON`: Boundary of a polygon
...
**Drawing Primitives**

Commands between glBegin() and glEnd():
- glVertex*: Specifies vertex coordinates
- glColor*: Specifies color

Example:
```c
glBegin(GL_POLYGON);
glColor3f (1.0, 0.0, 0.0);
glVertex3f (0.25, 0.25, 0.0);
glVertex3f (0.75, 0.25, 0.0);
gColor3f (0.0, 0.0, 1.0);
gVertex3f (0.75, 0.75, 0.0);
gVertex3f (0.25, 0.75, 0.0);
glEnd();
```

**Lines and Points Features**

- **glPointSize** (size): Specifies the size of the point (default is 1.0)

- **glLineWidthSize** (width): Specifies the width of the line (default is 1.0)

  When antialiasing is disabled, values are rounded to the nearest integer

- **glLineStipple** (factor, pattern)

  **glEnable(GL_LINE_STIPPLE)** : Used to draw dashed and dotted lines. Pattern is stretched by factor

Example:
```c
#define PI 3.14159
GLint CirclePoints = 100;
gBegin(GL_LINE_LOOP);
  for(i=0; i<CirclePoints; ++i) {
    angle = 2 * PI * i / CirclePoints;
    glVertex2f (cos(angle), sin(angle));
  }
gEnd();
```

Note: Obviously, not the best way to draw a circle...