Primitive Objects

Lecture 5
Sections 2.4, 2.5, 4.2

Robb T. Koether

Hampden-Sydney College

Mon, Sep 7, 2009
Outline

1. Color
2. Primitive Objects
3. Examples
4. Other Shapes
5. Assignment
The GPU maintains a large number of state variables, one of which is color.

The function `glColor*()` is used to set the current color.

Each vertex (point) will assume various attributes, including color, according to the state of the GPU at the time the vertex is created.

Until the color state is changed, it is used for each object drawn.
The function `glColor*()` takes 3 or 4 parameters.

- Red level
- Green level
- Blue level
- Optional alpha value (for transparency)

We will probably not use the alpha value.
OpenGL does not overload functions.

Why not?

Therefore, a separate function name must be chosen for each choice of parameter list.
The * in the function name `glColor*()` represents any of a list of options:

- `glColor3b()` - 3 bytes (unsigned chars).
- `glColor3bv()` - array of 3 bytes.
- `glColor3d()` - 3 doubles.
- `glColor3dv()` - array of 3 doubles.
- `glColor3f()` - 3 floats.
- `glColor3fv()` - array of 3 floats.
- `glColor3i()` - 3 ints.
- `glColor3iv()` - array of three ints.
OpenGL Function Notation

- `glColor3s()` - 3 shorts.
- `glColor3sv()` - array of 3 shorts.
- `glColor3ub()` - 3 unsigned bytes.
- `glColor3ubv()` - array of 3 unsigned bytes.
- `glColor3ui()` - 3 unsigned ints.
- `glColor3uiv()` - array of 3 unsigned ints.
- `glColor3us()` - 3 unsigned shorts.
- `glColor3usv()` - array of 3 unsigned shorts.
- `glColor4b()` - 4 bytes.
OpenGL Function Notation

- `glColor4bv()` - array of 4 bytes.
- `glColor4d()` - 4 doubles.
- `glColor4dv()` - array of 4 doubles.
- `glColor4f()` - 4 floats.
- `glColor4fv()` - array of 4 floats.
- `glColor4i()` - 4 ints.
- `glColor4iv()` - array of 4 ints.
- `glColor4s()` - 4 shorts.
- `glColor4sv()` - array of 4 shorts.
OpenGL Function Notation

- `glColor4ub()` - 4 unsigned bytes.
- `glColor4ubv()` - array of 4 unsigned bytes.
- `glColor4ui()` - 4 unsigned `ints`.
- `glColor4uiv()` - array of 4 unsigned `ints`.
- `glColor4us()` - 4 unsigned `shorts`.
- `glColor4usv()` - array of 4 unsigned `shorts`.
The glColor*() Function

- To draw a red object, make the function call
  \[ \text{glColor3f}(1.0, 0.0, 0.0); \]
  before drawing the object.
- To draw yellow objects, use
  \[ \text{glColor3f}(1.0, 1.0, 0.0); \]
OpenGL will draw a number of primitive objects.

- Points
- Lines
- Triangles
- Quadrilaterals
- Polygons
It also draws some more specialized primitive objects

- Line loops
- Line strips
- Triangle strips
- Quadrilateral strips
- Triangle fans
First, choose the type of object to be drawn.

Use `glBegin()` to specify the type of object and to begin the drawing.

Use `glEnd()` to end the drawing.

Between these two function calls, use `glVertex*()` to list the vertices of the objects.
Drawing Primitive Objects

Example (Draw a Red Triangle)

```c
glColor3f(1.0, 0.0, 0.0);
glBegin(GL_TRIANGLES);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
glEnd();
```
Example (Drawing Primitive Objects)

- The code.
- The executable.
Wrapping Direction

- When drawing a polygon (including triangles and quadrilaterals), the wrapping should be counterclockwise.
- Counterclockwise wrapping defines the front face.
- Clockwise wrapping defines the back face.
Drawing Two Triangles

Example (Draw a Red and a Green Triangle)

```c
glBegin(GL_TRIANGLES);
    // Draw a red triangle
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
    // Draw a green triangle
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glVertex3f(2.0, 0.0, 0.0);
    glVertex3f(1.0, 1.0, 0.0);
    glEnd();
```
Triangle Orientation

Example (Triangle Orientation)

- The code.
- The executable.
Examples

- Now, using a set of 12 points, each with a different color, we will draw each kind of primitive object.
Points, Lines, and Triangles

Example (Points)
- The code.
- The executable.

Example (Lines)
- The code.
- The executable.

Example (Triangles)
- The code.
- The executable.
Quadrilaterals, Polygons, and Triangle Fans

Example (Quadrilaterals)
- The code.
- The executable.

Example (Polygons)
- The code.
- The executable.

Example (Triangle Fans)
- The code.
- The executable.
Line Loops and Line Strips

Example (Line Loop)
- The code.
- The executable.

Example (Line Strip)
- The code.
- The executable.
Triangle Strips and Quadrilateral Strips

Example (Triangle Strip)
- The code.
- The executable.

Example (Quadrilateral Strips)
- The code.
- The executable.
Other Shapes

- All objects that are drawn are composed of these primitive objects.
- For example, a “circle” is a polygon with so many sides that it looks round.
- 40 sides is usually sufficient.
- Curved surfaces consist of many small facets, each of which is a polygon.
A Circle

Example (A Circle)

- The code.
- The executable.
A Star

- OpenGL assumes that all polygons are convex.
- If we pass OpenGL a star-shaped polygon, it might draw a pentagon.
Example (A Star)

- The code.
- The executable.
Example (A Sphere)

- The code.
- The executable.

The GLUT library, generally speaking, contains functions that are related to the windowing system.

However, it also contains special functions for drawing spheres and few other solids.

```c
glutSolidSphere(radius, stacks, slices)
```

```c
glutWireSphere(radius, stacks, slices)
```
A GLUT Sphere

Example (A GLUT Sphere)

- The code.
- The executable.
Homework

- Read Sec. 2.4 – primitive objects and their attributes.
- Read Sec. 2.5 – color.
- Read Sec. 4.2 – 3D primitives.