Primitive Objects

Lecture 5
Sections 2.4, 2.5, 3.2

Robb T. Koether

Hampden-Sydney College

Fri, Sep 2, 2011

Outline

Outline

Points and Color

- The GPU maintains a large number of state variables, one of which is 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*()` is used to set the current color.
- The `*` represents the number of arguments and their type.
- For example, `glColor3f()` takes three **`floats`** as its parameters.

OpenGL Function Notation

Function	Arguments	Function	Arguments
`glColor3b()` `glColor3bv()`	3 bytes array of 3 bytes	`glColor4b()` `glColor4bv()`	4 bytes array of 4 bytes
`glColor3d()` `glColor3dv()`	3 **`doubles`** array of 3 **`doubles`**	`glColor4d()` `glColor4dv()`	4 **`doubles`** array of 4 **`doubles`**
`glColor3f()` `glColor3fv()`	3 **`floats`** array of 3 **`floats`**	`glColor4f()` `glColor4fv()`	4 **`floats`** array of 4 **`float`**
`glColor3i()` `glColor3iv()`	3 **`ints`** array of three **`ints`**	`glColor4i()` `glColor4iv()`	4 **`ints`** array of 4 **`ints`**
`glColor3s()` `glColor3sv()`	3 **`shorts`** array of 3 **`shorts`**	`glColor4s()` `glColor4sv()`	4 **`shorts`** array of 4 **`shorts`**
`glColor3ub()` `glColor3ubv()`	3 unsigned bytes array of 3 unsigned bytes	`glColor4ub()` `glColor4ubv()`	4 unsigned bytes array of 4 unsigned bytes
`glColor3ui()` `glColor3uiv()`	3 unsigned **`ints`** array of 3 unsigned **`ints`**	`glColor4ui()` `glColor4uiv()`	4 unsigned **`ints`** array of 4 unsigned **`ints`**
`glColor3us()` `glColor3usv()`	3 unsigned **`shorts`** array of 3 unsigned **`shorts`**	`glColor4us()` `glColor4usv()`	4 unsigned **`shorts`** array of 4 unsigned **`shorts`**

The `glColor*()` Function

- The functions `glColor3*()` take parameters representing the red, green, and blue levels of the color.
- The functions `glColor4*()` take one more parameter representing the transparency.
- The functions `glColor*v()` take a single parameter that is an array name.

OpenGL Function Notation

- Why does OpenGL not overload a single color function `glColor()`?

The glColor*() Function

- To draw a red object, make the function call

```
glColor3f(1.0, 0.0, 0.0);
```

  before drawing the object.
- To draw yellow objects, use

```
glColor3f(1.0, 1.0, 0.0);
```

Outline

Primitive Objects

- OpenGL will draw a number of primitive objects.
  - Points
  - Lines
  - Triangles
  - Quadrilaterals (deprecated in v. 3.0)
  - Polygons (deprecated in v. 3.0)

Primitive Objects

- It also draws some “extended” primitive objects
  - Line loops
  - Line strips
  - Triangle strips
  - Quadrilateral strips (deprecated in v. 3.0)
  - Triangle fans

Drawing Primitive Objects

- 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.

The Polygon Mode

- The polygon can be either filled or outlined.
- Call the function `glPolygonMode()`, passing either or `GL_LINE`, before drawing the polygon.
- The default is `GL_FILL`.

Drawing Primitive Objects

Example (Draw a Red Triangle)

```
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();
```

Drawing Primitive Objects

Example (Drawing Primitive Objects)

- Lecture 5 Demo 1 - Lines and polygons in grayscale.
- Lecture 5 Demo 2 - Lines and polygons in solid color.
- Lecture 5 Demo 3 - Lines and polygons in blended colors.

Wrapping Direction

- When drawing polygons (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)

```
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();
```

Outline

Primitive Objects

Example (Primitive Objects)

- Lecture 5 Demo 4 - The points in the Sierpinski Gasket.
- Lecture 5 Demo 5 - A 3D figure consisting of line segments.
- Lecture 5 Demo 6 - A twisted ribbon made from a triangle strip.
- Lecture 5 Demo 7 - A cone made from a triangle fan.

Outline

Other Shapes

- All objects that are drawn are composed of these primitive objects.
- For example, a “circle” can be drawn as a polygon with so many sides that it looks round.
- 40 sides is usually sufficient.

Other Shapes

- We can use the deprecated `GL_POLYGON` type or we can use the `GL_TRIANGLE_FAN` type if we want to fill the circle.
- If we want only to outline the circle, then we can use `GL_POLYGON` (with `GL_LINE`) or `GL_LINE_LOOP`.

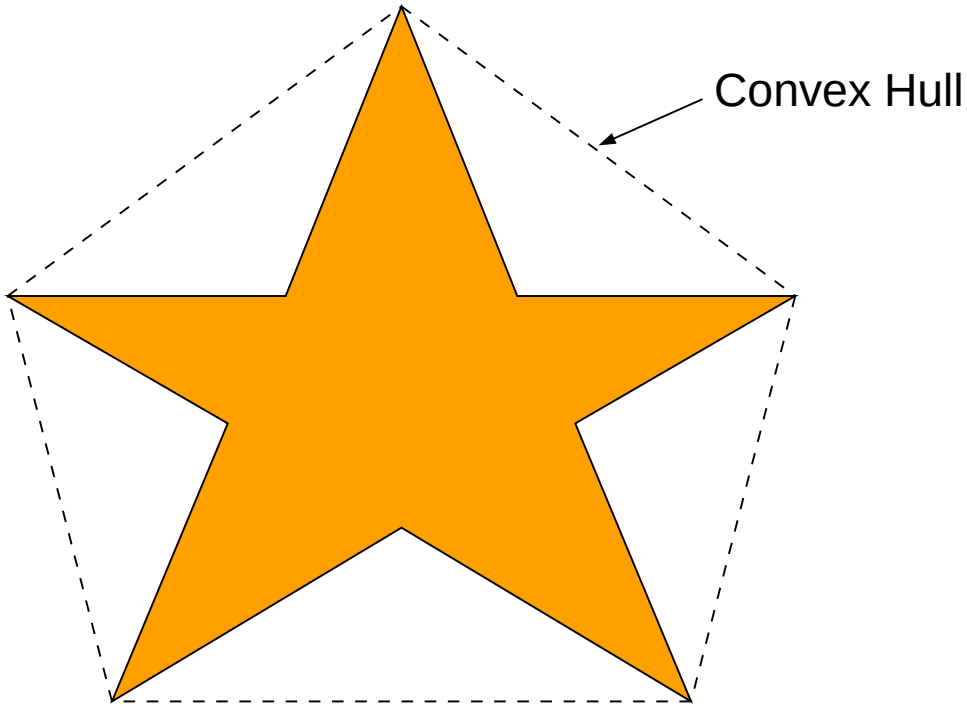
A Circle

Example (A Circle)

- Lecture 5 Demo 8 - A circle.

A Star

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



A Star

Example (A Star)

- Lecture 5 Demo 9 - A star.

A Sphere

Example (A Sphere)

- Curved surfaces consist of may small facets, each of which is a polygon.
- We can draw a sphere using many triangle strips along the latitude lines and triangle fans at the poles

A Sphere

Example (A Sphere)

- Lecture 5 Demo 10 - A shiny sphere.

The GLUT Library

- GLUT - Graphics Library Utility Toolkit.
- 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.

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

A GLUT Sphere

Example (A GLUT Sphere)

- Lecture 5 Demo 11 - The GLUT sphere.

Outline

Homework

Homework

- Read Sec. 2.4 – primitive objects and their attributes.
- Read Sec. 2.5 – color.
- Read Sec. 3.2 – 3D primitives.