

OpenGL Handout

OpenGL Examples

OpenGL is an interface for drawing polygons. This interface has been implemented on multiple operating systems and is supported by multiple graphics cards. More importantly, OpenGL will be the interface that you will be using for many assignments in CS123. The support code takes care of initializing the library and setting up the necessary windows. You are responsible for getting OpenGL to draw polygons on the screen. The following code draws a triangle with vertices $(-1, 0, 0)$, $(1, 0, 0)$, and $(0, 1, 0)$:

```
glBegin(GL_TRIANGLES);
    glVertex3f(-1.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
glEnd();
```

In the code above the `GL_TRIANGLES` parameter passed to `glBegin()` specifies that every set of three vertices submitted should be interpreted as a new triangle. You will probably always want to use the `GL_TRIANGLES` parameter for assignments in CS123 because we expect that you will only be drawing triangles. The `glVertex3f()` function call submits a vertex to OpenGL. The `glVertex3f()` function must always be called in between calls to `glBegin()` and `glEnd()`. Another restriction is that some of the only OpenGL functions that you can call in between a `glBegin()/glEnd()` block are `glNormal3f()` and `glColor3f()` (See the OpenGL rules section on the last page for a concise list of rules. There are many valid functions, but you don't need most of them for cs123 assignments). When a vertex is submitted, the current color, normal, and lights are all used to calculate the color of that vertex. (This calculation for the vertex is called "lighting": remember that lighting a vertex is different from eventually shading the interior of the triangle.) You will use the `IScene` class to turn lights on and off, but you are responsible for setting colors and normals yourself. In the following code we set the current color to white before lighting any of the three vertices.

```
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
    glVertex3f(-1.0, 0.0, 0.0);
    glVertex3f(1.0, 0.0, 0.0);
    glVertex3f(0.0, 1.0, 0.0);
glEnd();
```

Note that if we want all three vertices to have the same color we only need to set the current color once. Every time a vertex is submitted, the current color will still be white in the above example. OpenGL keeps track of a global color state; this is why you may hear OpenGL referred to as a state machine. The short version: *you only need to set colors or normals when they change from one vertex to the next.*

Note that there are restrictions on what OpenGL functions you may call within a `glBegin()/glEnd()` block, but there are no restrictions on calling non-OpenGL functions. The following example shows

how a non-OpenGL function can be called between `glBegin()` and `glEnd()` function calls. The following example also shows how to turn on a light using `IScene` function calls in the `initialize()` method. You cannot call `IScene` methods like `lightOn()` within a `glBegin()/glEnd()` block because these support code methods will call OpenGL functions, and calling general OpenGL functions is illegal within a `glBegin()/glEnd()` block is illegal. (See the OpenGL rules section on the last page for a concise list of rules).

```
void ISceneSubclass::initialize()
{
    IScene::initialize();

    // turn on one light with reasonable defaults
    lightOn(0);
    lightPos(0, 0.0, 0.0, 2.0);
    lightColor(0, 1.0, 1.0, 1.0);
    lightFunc(0, 0.0, 1.0, 0.1);
}

// this function must be called between glBegin() - glEnd() function
// calls because it calls the glVertex3f() function
void ISceneSubclass::submitTriangle(float xOffset)
{
    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(-1.0 + xOffset, 0.0, 0.0);

    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(1.0 + xOffset, 0.0, 0.0);

    glNormal3f(0.0, 0.0, 1.0);
    glVertex3f(xOffset, 1.0, 0.0);
}

// this draws 4 triangles along the x-axis
void ISceneSubclass::drawScene()
{
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_TRIANGLES);
    for (i = 0; i < 4; i++) {
        submitTriangle((i - 2.0) * 3.0);
    }
    glEnd();
}
```

OpenGL Rules

- All `glVertex3f()` calls must be between calls to `glBegin()` and `glEnd()`.
- `glNormal3f()` and `glColor3f()` can be called either inside or outside of a `glBegin()/glEnd()` block.
- `glVertex3f()`, `glNormal3f()` and `glColor3f()` are the only OpenGL functions that should be called within a `glBegin()/glEnd()` block. If you violate this rule and call another OpenGL function **it probably will not work**. To be on the safe side you should probably avoid calling support code functions within `glBegin()/glEnd()` blocks because the support may itself call OpenGL functions. So for instance if you call `IScene::setTransform()` within a `glBegin()/glEnd()` block **nothing will happen**.

Further Information

Note that `glVertex4f()` also exists which lets you set the fourth homogenous coordinate `w`. Besides this the two functions are identical. The most complete source of information for OpenGL is available in a book commonly referred to as the “Red Book” (named from its red cover). You can access the full contents of the book online for free! See the resources page for a link.

If you are interested in drawing triangles faster or doing neat effects like transparency then talk to a CS123 TA on hours.