OpenGL

CS116B

Chris Pollett

May 4, 2004.

Outline

- OpenGL for the stuff in Chapter 10

OpenGL -Lighting

- We have already discussed a little how to do lighting in OpenGL and HW1 and HW2 solutions have examples. Here is a quick review. To make a light:

  glLightfv(lightName, lightProperty, propertyValue);

  where lightName is GL_LIGHT0, GL_LIGHT1, …
- Then use:

  glEnable(lightName);
- Properties we can set are:

  GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION.
- To use a light, we need the line:

  glEnable(GL_LIGHTING);
- To control attenuation we can set:

  glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.5);

  We can also set GL_ LINEAR_ ATTENUATION and GL_QUADRATIC_ATTENUATION terms.

OpenGL- Directional Light Global Lighting

To create a spotlight in OpenGL:

- We need to say the direction the light is pointing:

  GLfloat dirVector[] = {1.0, 0.0, 0.0};

  glLightfv(GL_LIGHT2, GL_SPOT_DIRECTION, dirVector);
- Say what angle to cut it off:

  glLightf(GL_LIGHT2, GL_SPOT_CUTOFF, 30.0);
- Say how fast it falls off with angle:

  glLightf(GL_LIGHT2, SPOT_EXPONENT, 2.5);
- To set the global ambient lighting of a scene we can use:

  GLfloat global[] = {0.0, 0.0, .3, 1}; // 1 is alpha blending value

  glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global);

OpenGL Materials

- The Hw1, Hw2, Hw3 solutions also give examples of textures.
- The coefficients which OpenGL uses to multiply against the ambient, diffuse, and specular intensity are determined by the materials property.
- To set the current value used by OpenGL one can use the command:

  glMaterial*(surfFace, surfProperty, propertyValue);

  Here surfFace is one of GL_FRONT, GL_BACK, GL_FRONT_AND_BACK

  Here surfProperty is one of GL_EMISSION, GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR

  Finally, the propertyValue, is an glFloat array giving the coefficients for the RGBA values.

OpenGL Atmospheric Effects

- OpenGL supports the creation of fog.
- To use Fog we begin with the command: glEnable(GL_FOG);
- The fog color (default is black) is set with: glFogfv(GL_FOG_COLOR, rgbaColor);
- The way that fog attenuate with distance is set with:
- glFog(GL_FOG_MODE, GL_EXP); /*Or can use GL_EXP2 if we want a quadratic in exponent. Another option is GL_LINEAR */

OpenGL Surface Rendering

- In OpenGL we can select between constant intensity and Gouraud shading:

  glShadeModel(surfRenderingMethod);

  //GL_FLAT for constant; GL_SMOOTH for Gouraud
- In addition, we can by hand set the normal to be used by all subsequent glVertex command using glNormal.

  For example,

```
glNormal3fv(normalVector);
glBegin(GL_TRIANGLES);
   glVertex3fv(vertex1);
   glVertex3fv(vertex2);
   glVertex3fv(vertex3);
glEnd();
```

OpenGL Halftoning Operations

Dithering can be enable/disabled using:

glEnable(GL_DITHER);

and

glDisable(GL_DITHER);

OpenGL Textures

- OpenGL supports 1D, 2D, 3D textures.
- The basic commands to do 1D texturing are:

```
// say what texture to use
glTextureImage1D(GL_TEXTURE_1D, 0, GL_RGBA, nTexColors, 0,
    dataFormat, dataType, lineTexArray);
/*the first 0 is an offset into our array, nTexColors - is the size of our
    array (must be power of 2). The second 0 says no border.
    dataformat is used to say the color format for the texture. For
    example, GL_RGBA. datatype could be GL_UNSIGNED_BYTE,
    GL_INT, GL_FLOAT, finally lineTexArray is our array*/
//say how texture should be reduced/enlarged to fill an area
glTexParamteri(GL_TEXTURE_1D, GL_TEXTURE_MAG_FILTER,
    GL_NEAREST);
glTexParamteri(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER,
    GL_NEAREST);
glEnable(GL_TEXTURE_1D); //enable
glBegin(GL_LINES);
    glTexCoord1f(0.0);   glVertex3fv(pt1);
    glTexCoord1f(1.0);   glVertex3fv(pt1);
glEnd();
```

2D and 3D Textures in OpenGL

- The idea is similar to the 1D case except now to set up use:

glTextureImage2D(GL_TEXTURE_2D, 0, GL_RGBA, nWidth, nHeight, 0, dataFormat, dataType, surfTexArray);

glEnable(GL_TEXTURE_2D);

//Width and height must both be a power of 2.

- Filter step now take 2D INSTEAD OF 1D:

glTexParamteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);

glTexParamteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

- We now use glTexCoord2f(x,y); // where x and y between 0 and 1.0
- 3D is similar now have glTexImage3D, etc ..
- More texture stuff in book….