LEI / LCC
Departamento de Informática
Universidade do Minho

Lighting

Lights, Materials and Normals

OpenGL - Normals

- To compute lighting a normal vector per vertex is required.
- The normal vector is a vector which is perpendicular to the surface.

```
glBegin(GL_TRIANGLE);
   glNormal3f(0,1,0);
   glVertex3f(0,0,0);
   glNormal3f(0,1,0);
   glVertex3f(0,0,1);
   glNormal3f(0,1,0);
   glVertex3f(1,0,0);
glEnd();
```

When using the same normal for every vertex:

```
glBegin(GL_TRIANGLE);
        glNormal3f(0,1,0);
        glVertex3f(0,0,0);
        glVertex3f(0,0,1);
        glVertex3f(1,0,0);
glEnd();
```

OpenGL – Normals and VBOs

- VBO Init
  - Step 1 a) Enable Buffers

```
glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
```

OpenGL – Normals and VBOs

- VBO Init

  – Step 1 b – Allocate and fill the vertex and normal arrays

```
// vertex array
float *vertexB;
// fill the array
…
// normal array
float *normalB;
// fill the array
...
```

  – Step 1 c (optional) – Allocate and fill the index array

```
unsigned int *indices;
...
```

OpenGL – Normals and VBOs

- VBO Init

- Step 1 d : Create the VBOs

```
GLuint buffers[2];
// two buffers: vertex coordinates and normals
float *vertexB, *normalB;
...
// create two buffers
glGenBuffers(2, buffers);

// bind and copy data
glBindBuffer(GL_ARRAY_BUFFER,buffers[0]);
glBufferData(GL_ARRAY_BUFFER,arraySize, vertexB, GL_STATIC_DRAW);

glBindBuffer(GL_ARRAY_BUFFER,buffers[1]);
glBufferData(GL_ARRAY_BUFFER, arraySize, normalB,GL_STATIC_DRAW);
```

OpenGL – Normals and VBOs

- Draw with VBOs
  - Step 2 a – Semantics
    - For each buffer: what will it be used for

```
glBindBuffer(GL_ARRAY_BUFFER,buffers[0]);
glVertexPointer(3,GL_FLOAT,0,0);

glBindBuffer(GL_ARRAY_BUFFER,buffers[1]);
// normals always have 3 components
glNormalPointer(GL_FLOAT,0,0);
```

OpenGL – Normals and VBOs

- Draw with VBOs
  - Step 2 b: Drawing
    - With an index list

```
glDrawElements(GL_TRIANGLES, count ,GL_UNSIGNED_INT, indices);
```

    - Without an index list

```
glDrawArrays(GL_TRIANGLES, first, count);
```

Note: count is the number of vertices/indices to draw

OpenGL - Materials

- Setup materials:

```
glMaterialfv(GL_FRONT, componente, array);
glMaterialf(GL_FRONT,GL_SHININESS,value);     0.0..128.0
```

Component:

```
GL_DIFFUSE
GL_AMBIENT
GL_SPECULAR
GL_EMISSION
GL_AMBIENT_AND_DIFFUSE
```

- Example: diffuse color red

```
float red[4] = {0.8f, 0.2f, 0.2f, 1.0f};
glMaterialfv(GL_FRONT, GL_DIFFUSE, red);
```

OpenGL - Lighting

- Light Properties

```
glLight{if}(GL_LIGHTi, param, value1,value2, ...);
glLight{if}v(GL_LIGHTi, param, array_values);
```

- Notes:
  - 1: Considering a light stationary in the world, its position must be defined after `gluLookAt`.
  - 2: The light colors and all other properties can be defined in the initialization

OpenGL - Lighting

- Directional Light

```
GLfloat amb[4] = {0.2, 0.2, 0.2, 1.0};
GLfloat diff[4] = {1.0, 1.0, 1.0, 1.0};
GLfloat pos[4] = {0.0, 0.0 ,1.0, 0.0};

// light position
glLightfv(GL_LIGHT0, GL_POSITION, pos);
// light colors
glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff);
```

0.0 says that the "position" is actually a vector, or a direction

The direction is towards the light, not from the light

OpenGL - Lighting

- Enable/Disable individual lights (by default lights are disabled)

```
glEnable(GL_LIGHTi); // i = 0..7
glDisable(GL_LIGHTi);
```

- Enable/Disable Lighting (by default it is disabled)

```
glEnable(GL_LIGHTING);
glDisable(GL_LIGHTING);
```

- Note: These functions can be called during initialization.

Assignment

- Define the normal vectors for the cylinder
- Add all the necessary instructions to draw a cylinder lit by a directional light