Iluminação no OpenGL

Hitoshi - DCC/IME

Iluminação no OpenGL

- Para se obter efeitos de iluminação no OpenGL, são necessários os seguintes passos:
  - Definir o modelo de coloração (shading)
    - Smooth or flat
  - Definir as luzes, suas posições e propriedades
  - Definir a propriedade dos materiais

Iluminação e modelo de coloração

- glLightModel*()
- glLightModelf(Glenum pname, Glfloat param);
- glLightModelfv(Glenum pname,
  const Glfloat * params);

Onde pname:
GL_LIGHT_MODEL_AMBIENT:
define a luz ambiente

Como pintar polígonos

- glShadeModel(GL_SMOOTH)
  - ß
- glShadeMode(GL_FLAT)
  - A coloração é a mesma em toda a superfície

Modelos de interpolação

- Interpolação de Gouraud
  - A função de interpolação mantêm a cor em cada vértice
- Interpolação de Phong
  - As normais de cada vértice são interpoladas
  - Gera resultados mais realistas
- O OpenGL usa Gouraud interpolation

Iluminação no OpenGL

- Para utilizar luz no OpenGL, você precisa habilitar as fontes através de glEnable(GL_LIGHTING)
  - E desligar usando glDisable (GL_LIGHTING)
- O OpenGL permite 8 fontes de luz:
  - GL_LIGHT0 a GL_LIGHT7

Exemplo em C

- glClearColor(0.0, 1.0, 0.0, 1.0); // intentionally background
- glEnable(GL_NORMALIZE); // normalize normal vectors
- glShadeModel(GL_SMOOTH); // do smooth shading
- glEnable(GL_LIGHTING); // enable lighting

- // ambient light (red)
- GLfloat ambientIntensity[4] = {0.9, 0.0, 0.0, 1.0};
- glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientIntensity);

Exemplo

- Fonte de luz 0
  - Posição em (2, 4, 5, 1)
  - Cor ambiente = (0.9, 0, 0)
  - Cor difusa e especular (1.0, 1.0, 1.0)
- Normalmente essas cores são feitas iguais. Nesse exemplo apenas mostramos que podem ser diferentes
- Não há uma unidade

- // set up light 0 properties
- GLfloat lt0Intensity[4] = {1.0, 1.0, 1.0, 1.0}; // white
- glLightfv(GL_LIGHT0, GL_DIFFUSE, lt0Intensity);
- glLightfv(GL_LIGHT0, GL_SPECULAR, lt0Intensity);
- GLfloat lt0Position[4] = {2.0, 4.0, 5.0, 1.0}; // location
- glLightfv(GL_LIGHT0, GL_POSITION, lt0Position);

- // attenuation params (a,b,c)
- glLightf (GL_LIGHT0, GL_CONSTANT_ATTENUATION, 0.0);
- glLightf (GL_LIGHT0, GL_LINEAR_ATTENUATION, 0.0);
- glLightf (GL_LIGHT0, GL_QUADRATIC_ATTENUATION, 0.1);
- glEnable(GL_LIGHT0);

Definição dos materiais

- a renderização depende da interação da luz com a superfície dos objetos
- As propriedades são associadas aos vértices
  - Em smooth shading, a coloração entre vértices é interpolada
  - Em flat shading a cor do 1o vértice é utilizada

Objetos

- Todo objeto em OpenGL é um polígono
- Cada face pode ser pintada (front / back)
- Em geral, só precisamos pintar a face frontal
  - A face frontal é definida pela ordem anti-horária dos vértices.

definição de um triângulo

```
glBegin(GL.GL_TRIANGLES);
glMaterialfv(...); // propriedades do material
   // ou glColor
glNormal3f(...); // definição da normal
glVertex3d(....); // vértices do triângulo
   // demais vértices
glEnd;
```

definição do material

glMaterialfv(GLenum face, GLenum pname, GLfloat *param);

face: GL_FRONT, GL_FRONT_AND_BACK

pname: GL_DIFUSE, GL_SPECULAR, GL_AMBIENT, GL_EMISSION, GL_SHINESS)

Outras opções

- Você pode habilitar/desabilitar:
  - GL_CULL_FACE: backface culling é o processo de eliminar as superfícies que estão de "costas" para o observador. O OpenGL permite que você escolha a face a ser eliminada: GL_FRONT ou GL_BACK
  - GL_NORMALIZE: automaticamente normaliza todos os vetores normais para que tenha comprimento unitário.

Outras dicas

- Antes de desenhar, limpe o color buffer bit e o depth buffer bit

  gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);

  - o color buffer bit limpa o canvas, pintando-o com a cor de fundo definida por glClearColor(...);
  - o depth buffer bit limpa o buffer de profundidade, para que apenas as superfícies visíveis mais próximas ao observador sejam renderizadas.