Computação Gráfica (IME 04-10842)
2022.2

Iluminação e Texturas no OpenGL

Gilson. A. O. P. Costa (IME/UERJ)

gilson.costa@ime.uerj.br

Conteúdo

- OpenGL buffers
- Iluminação
- Materiais
- Textura
- Quádricas
- Exemplos

Framebuffer

- O OpenGL possui um conjunto de ***buffers*****,** coletivamente chamados de ***framebuffer***, que determinam a cor de cada pixel da tela.
- Os ***buffers*** são matrizes com uma ou mais dimensões/profundidade, e altura e largura iguais à da tela/monitor.
- A quantidade de *buffers* padrões depende do sistema de janelas e hardware disponível, mas geralmente existem quatro tipos: ***color buffers***, ***depth buffer***, ***stencil buffer*** e ***accumulator buffer***.

Color Buffers

- Os ***color buffers*** registram as cores dos pixels da tela, que podem ser definidas através de mapas de cores, valores RGB, ou RGBA (*Alpha* = transparência).
- Implementações OpenGL que suportam visão estéreo possuem *buffers* de cores para as imagens da direita (*right buffer*) e da esquerda (*left buffer*).
- Quando *buffers* duplos (*double buffering*) estão habilitados, há um *buffer* "da frente" (*front buffer*), visível, e um *buffer* "de trás" (*back buffer*).
- Quando se usa *buffers* duplos, o desenho é realizado no *back buffer*, que substitui o *front buffer* quando desejado, e.g., via **`glutSwapBuffers()`**.

Accumulation Buffer

- O ***accumulation buffer*** contém valores RGBA para os pixels da tela.
- O *buffer* de acumulação é simplesmente um *buffer* de cor extra usado para acumular imagens compostas.
- É útil para alguns efeitos especiais, como suavização, profundidade de campo e desfoque de movimento.

Stencil Buffer

- O ***stencil buffer*** é usado para restringir a o desenho a certas partes da tela.
- Exemplo: desenhar uma cena de dentro de um carro em movimento.
- Pode-se desenhar na parte “de dentro” ou “de fora” do ***stencil buffer***.

Depth Buffer

- O ***depth buffer*** registra a profundidade (distância do olho/câmera) de cada pixel da tela.
- O *depth buffer* é usado para controlar a sobreposição de objetos.
- Quando um objeto é desenhado, seus respectivos pixels só são pintados no *color buffer* se as profundidades são menores do que as indicadas no *depth buffer* (caso em que a informação do *depth buffer* é alterada).

Depth Buffer

- Inicialmente, os valores de profundidade são especificados para serem o maior possível através do comando **`glClear(GL_DEPTH_BUFFER_BIT)`**.
- Habilitando o *depth-buffering* através dos comandos **`glutInitDisplayMode(GLUT_DEPTH | ...)`** e **`glEnable(GL_DEPTH_TEST)`**, antes de cada pixel ser desenhado é feita uma comparação com o valor de profundidade já armazenado.
- Se o valor de profundidade for menor, o pixel é desenhado e o valor de profundidade é atualizado, caso contrário o pixel é desprezado.

Iluminação

- Em OpenGL a cor de uma fonte de luz é caracterizada pela quantidade de vermelho (R), verde (G) e azul (B) que ela emite.
- A luz em uma cena pode vir de várias fontes de luz (até 10 fontes) que podem ser "ligadas" ou "desligadas" individualmente.
- A luz pode vir de uma direção ou posição (por exemplo, uma lâmpada) ou como resultado de várias reflexões.
- Pode-se definir também uma luz ambiente: não é possível determinar de onde ela vem, mas ela desaparece quando a fonte de luz é desligada.

Iluminação

No OpenGL pode-se especificar quatro componentes independentes para cada fonte de luz:

- Ambiente: resultado da luz refletida no ambiente; é uma luz que vem de todas as direções.
- Difusa: luz que vem de uma direção, atinge uma superfície e é refletida em todas as direções; assim, parece possuir o mesmo brilho independente de onde a câmera está posicionada.
- Especular: luz que vem de uma direção e tende a ser refletida numa única direção.
- Emissiva: simula a luz que se origina de um objeto; a cor emissiva de uma superfície adiciona intensidade ao objeto, mas não é afetada por qualquer fonte de luz; ela também não introduz luz adicional na cena.

Materiais

- No OpenGL assume-se que cada superfície de um objeto é composta de um material com várias propriedades.
- O material pode emitir luz, refletir parte da luz incidente em todas as direções, ou refletir parte da luz incidente numa única direção, como um espelho.
- A influência da luz sobre a superfície dos materiais/objetos segue um determinado modelo de colorização: **`glShadeModel()`**.
- Existem dois modelos de colorização: um polígono preenchido/sólido pode ser desenhado com uma única cor (**`GL_FLAT`**), ou com uma variação de tonalidades (**`GL_SMOOTH`**), também chamado de modelo de colorização de Gouraud.

Materiais

- A cor do superfície de um objeto depende da porcentagem de luz vermelha (R), verde (G) e azul (B) que ele reflete.
- Para um material/superfície, pode-se definir as proporções refletidas destas componentes de cor: se $\mathrm{R}=1$, $\mathrm{G}=0.5$ e $\mathrm{B}=0$ o material reflete toda luz vermelha incidente, metade da luz verde e nada da azul.
- Simplificadamente, a luz que chega no observador é dada por ($\mathrm{LR}\times\mathrm{MR}$, $\mathrm{LG}\times\mathrm{MG}$, $\mathrm{LB}\times\mathrm{MB}$), onde ($\mathrm{LR}$, LG, LB) são os componentes da luz e (MR, MG, MB) os componentes do material.

Exemplo #7: Iluminação × Materiais

```
#include <stdlib.h>
#include <gl/glut.h>
GLfloat fov, fAspect;
// Função callback chamada para fazer o desenho
void Desenha(void) {
      // Limpa a janela e o depth buffer
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluPerspective(fov, fAspect, 0.4, 500);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      gluLookAt(0, 80, 200, 0, 0, 0, 0, 1, 0);
      // Desenha o teapot com a cor corrente
      glColor3f(0.6f, 0.4f, 0.1f);
      glutSolidTeapot(50.0f);
      glutSwapBuffers();
}
```

Exemplo #7: Iluminação × Materiais

```
// Inicializa parâmetros de iluminação
void Inicializa(void) {
      // Características da fonte de luz
      GLfloat luzAmbiente[4] = { 0.2, 0.2, 0.2, 1.0 };
      GLfloat luzDifusa[4] = { 0.7, 0.7, 0.7, 1.0 };
      GLfloat luzEspecular[4] = { 1.0, 1.0, 1.0, 1.0 };
      GLfloat posicaoLuz[4] = { 0.0, 50.0, 50.0, 1.0 };

      // Ativa o uso da luz ambiente
      glLightModelfv(GL_LIGHT_MODEL_AMBIENT, luzAmbiente);

      // Define os parâmetros da luz de número 0
      glLightfv(GL_LIGHT0, GL_AMBIENT, luzAmbiente);
      glLightfv(GL_LIGHT0, GL_DIFFUSE, luzDifusa);
      glLightfv(GL_LIGHT0, GL_SPECULAR, luzEspecular);
      glLightfv(GL_LIGHT0, GL_POSITION, posicaoLuz);
```

Exemplo #7: Iluminação × Materiais

```
        // Características do material
        GLfloat especularidade[4] = { 1.0,1.0,1.0,1.0 };
        GLint especMaterial = 60;
        glMaterialfv(GL_FRONT, GL_SPECULAR, especularidade);
        glMateriali(GL_FRONT, GL_SHININESS, especMaterial);

        glShadeModel(GL_SMOOTH);      // Modelo de colorização de Gouraud
        glEnable(GL_COLOR_MATERIAL); // Habilita a cor do material
        glEnable(GL_LIGHTING);        // Habilita o uso de iluminação
        glEnable(GL_LIGHT0);          // Habilita fonte de número 0
        glEnable(GL_DEPTH_TEST);      // Habilita o depth-buffering

        // Especifica que a cor de fundo da janela será preta
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

        fov = 45;
}
```

Exemplo #7: Iluminação × Materiais

```
// Função callback chamada quando o tamanho da janela é alterado
void AlteraTamanhoJanela(GLsizei w, GLsizei h){
      if (h == 0) h = 1; // Para previnir uma divisão por zero
      glViewport(0, 0, w, h); // Especifica o tamanho da viewport
      fAspect = (GLfloat)w / (GLfloat)h; // Correção de aspecto
      glutPostRedisplay(); // Redesenha
}
// Função callback chamada para gerenciar eventos do mouse
void GerenciaMouse(int button, int state, int x, int y){
      if (button == GLUT_LEFT_BUTTON) // Zoom-in
            if (state == GLUT_DOWN) {if (fov >= 10) fov -= 5;}
      if (button == GLUT_RIGHT_BUTTON) // Zoom-out
            if (state == GLUT_DOWN) {if (fov <= 130) fov += 5;}
      glutPostRedisplay(); // Redesenha
}
```

Exemplo #7: Iluminação × Materiais

```
// Programa Principal
int main(int argc, char** argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
      glutInitWindowSize(400, 350);
      glutCreateWindow("Bule Iluminado");
      glutDisplayFunc(Desenha);
      glutReshapeFunc(AlteraTamanhoJanela);
      glutMouseFunc(GerenciaMouse);
      Inicializa();
      glutMainLoop();
}
```

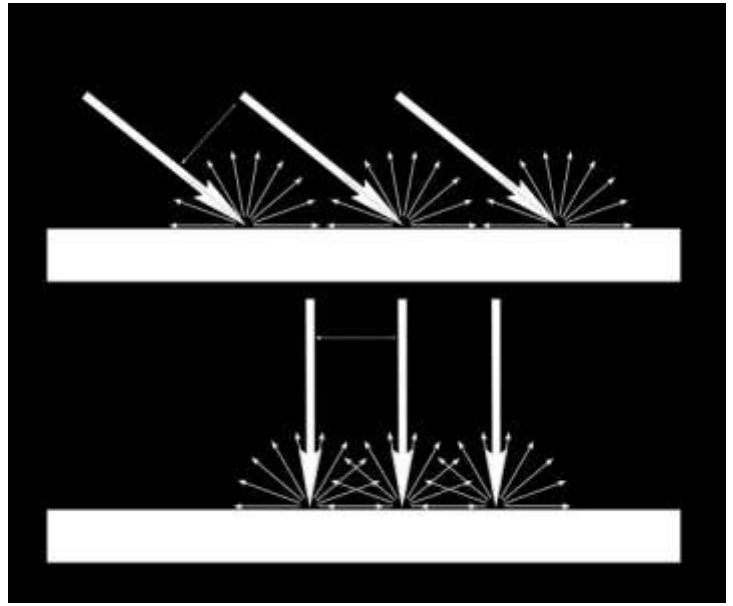
Exemplo #7: Iluminação × Materiais

Experimente:

- Em tempo de execução, altere o tamanho da janela criada pelo programa.
- Pressione os botões direito e esquerdo do mouse.
- Altere as características da fonte de luz e do material/superfície do bule.
- Altere o modelo de colorização para **`GL_FLAT`**.

Normais

- As primitivas 3D do GLUT e GLU já vem preparadas para a iluminação.
- Mas quando se cria objetos 3D a partir de primitivas básicas do OpenGL, deve-se estabelecer para cada superfície plana do objeto 3D seu vetor normal.
- O vetor normal à superfície é usados pelo OpenGL para calcular a quantidade de luz que incide sobre a superfície.



Exemplo #8: Normais

```
#include <stdlib.h>
#include <GL/glut.h>

GLfloat xRotated, yRotated, zRotated; //angulos de rotacao
bool normal; //indica se normais estão sendo usadas

// define a fonte de luz (LIGHT0)
void initLighting(void) {
      GLfloat lightposition[] = { 3.0, 3.0, 3.0, 0.0 };
      glDepthFunc(GL_LESS);
      glEnable(GL_DEPTH_TEST);
      glEnable(GL_LIGHT0);
      glEnable(GL_LIGHTING);
      glLightfv(GL_LIGHT0, GL_POSITION, lightposition);
      glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
      glEnable(GL_COLOR_MATERIAL);
      glClearColor(0.0, 0.0, 1.0, 0.0);
}
```

Exemplo #8: Normais

```
// callback para redimensionar janela glut
void resizeWindow(int x, int y){
      if (y == 0 || x == 0) return;
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      glFrustum(-1.5, 1.5, -1.5, 1.5, 3.0, 10.0);
      glMatrixMode(GL_MODELVIEW);
      if (x>y) glViewport(0, 0, y, y);
      else glViewport(0, 0, x, x);
}
// callback de teclado
void keyboard(unsigned char key, int x, int y){
      switch (key) {
      case 27: exit(0); break;
      case 'n': normal = !normal;
      Default: break;
      }
}
```

Exemplo #8: Normais

```
// callback idle
void idleFunc(void) {
      yRotated += 0.01;
      zRotated += 0.01;
      drawCube();
}

// desenha cubo
void drawCube(void){
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glTranslatef(0.0f, 0.0f, -5);
      glRotatef(yRotated, 0, 1, 0);
      glRotatef(zRotated, 0, 0, 1);
      glColor3f(1.0f, 1.0f, 0.0f);
```

Exemplo #8: Normais

```
glBegin(GL_QUADS);                          // Face posterior
if (normal) glNormal3f(0.0, 0.0, 1.0);      // Normal da face
glVertex3f(1.0f, 1.0f, 1.0f);
glVertex3f(-1.0f, 1.0f, 1.0f);
glVertex3f(-1.0f, -1.0f, 1.0f);
glVertex3f(1.0f, -1.0f, 1.0f);
glEnd();

glBegin(GL_QUADS);                          // Face frontal
if (normal) glNormal3f(0.0, 0.0, -1.0);     // Normal da face
glVertex3f(1.0f, 1.0f, -1.0f);
glVertex3f(1.0f, -1.0f, -1.0f);
glVertex3f(-1.0f, -1.0f, -1.0f);
glVertex3f(-1.0f, 1.0f, -1.0f);
glEnd();
```

Exemplo #8: Normais

```
glBegin(GL_QUADS);                          // Face lateral esquerda
if (normal) glNormal3f(-1.0, 0.0, 0.0);     // Normal da face
glVertex3f(-1.0f, 1.0f, 1.0f);
glVertex3f(-1.0f, 1.0f, -1.0f);
glVertex3f(-1.0f, -1.0f, -1.0f);
glVertex3f(-1.0f, -1.0f, 1.0f);
glEnd();

glBegin(GL_QUADS);                          // Face lateral direita
if (normal) glNormal3f(1.0, 0.0, 0.0);      // Normal da face
glVertex3f(1.0f, 1.0f, 1.0f);
glVertex3f(1.0f, -1.0f, 1.0f);
glVertex3f(1.0f, -1.0f, -1.0f);
glVertex3f(1.0f, 1.0f, -1.0f);
glEnd();
```

Exemplo #8: Normais

```
        glBegin(GL_QUADS);                          // Face superior
        if (normal) glNormal3f(0.0, 1.0, 0.0);      // Normal da face
        glVertex3f(-1.0f, 1.0f, -1.0f);
        glVertex3f(-1.0f, 1.0f, 1.0f);
        glVertex3f(1.0f, 1.0f, 1.0f);
        glVertex3f(1.0f, 1.0f, -1.0f);
        glEnd();

        glBegin(GL_QUADS);                          // Face inferior
        if (normal) glNormal3f(0.0, -1.0, 0.0);     // Normal da face
        glVertex3f(-1.0f, -1.0f, -1.0f);
        glVertex3f(1.0f, -1.0f, -1.0f);
        glVertex3f(1.0f, -1.0f, 1.0f);
        glVertex3f(-1.0f, -1.0f, 1.0f);
        glEnd();

        glutSwapBuffers();
}
```

Exemplo #8: Normais

```
int main(int argc, char** argv)
{
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
      glutInitWindowSize(240, 240);
      glutInitWindowPosition(100, 100);
      glutCreateWindow("Cubo iluminado");
      initLighting();
      glutDisplayFunc(drawCube);
      glutReshapeFunc(resizeWindow);
      glutKeyboardFunc(keyboard);
      glutIdleFunc(idleFunc);
      glutMainLoop();
      return 0;
}
```

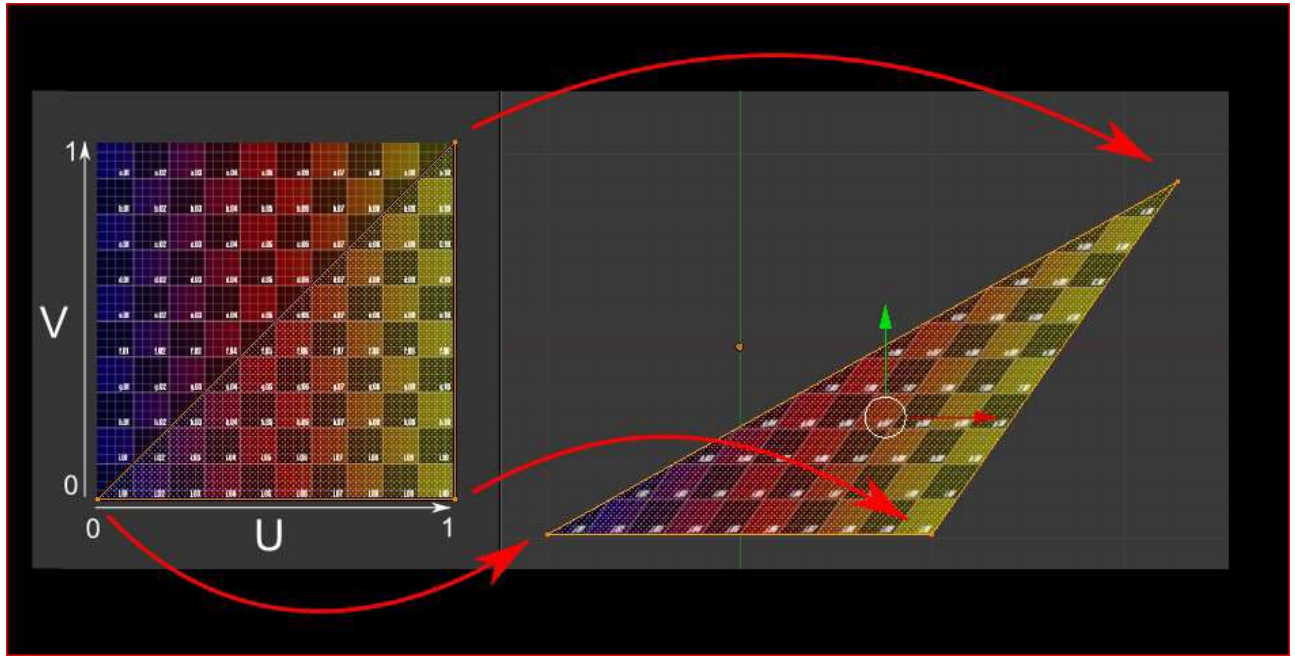
Exemplo #8: Normais

Experimente:

- Em tempo de execução, pressione a tecla “n”.
- Altere as características da fonte de luz e do material/superfícies do cubo.

Texturas

- No OpenGL uma textura pode ser lida de um ou mais imagens digitais.
- A textura desejada pode ser mapeada para os vértices da primitiva OpenGL a ser desenhada.



Texturas

- No OpenGL uma textura pode ser lida de um ou mais imagens digitais.
- A textura desejada pode ser mapeada para os vértices da primitiva OpenGL a ser desenhada.
- Para modelos 3D do Glut, pode-se usar também objetos/superfícies quádricas (**`GLUquadric`**) associadas a texturas **`gluQuadricTexture()`**.
- As texturas associadas às quádricas “envolvem” os modelos 3D do Glut.

Exemplo #9: Texturas

```
#include <stdlib.h>
#include <GL/glut.h>
#include "RgbImage.h"
// variaveis globais
GLfloat xRotated, yRotated, zRotated; //angulos de rotacao
GLuint  texture[1];                   // id da textura
char* filename = "./textura.bmp"; //arquivo com a textura
// callback para redimensionar janela glut
void resizeWindow(int x, int y) {
      if (y == 0 || x == 0) return;
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      glFrustum(-1.5, 1.5, -1.5, 1.5, 3.0, 10.0);
      glMatrixMode(GL_MODELVIEW);
      if (x>y) glViewport(0, 0, y, y);
      else glViewport(0, 0, x, x);
}
```

Exemplo #9: Texturas

```
void loadTextureFromFile(char *filename) {// lê textura de arquivo
      RgbImage theTexMap(filename);
      glGenTextures(1, &texture[0]); // Cria a textura
      glBindTexture(GL_TEXTURE_2D, texture[0]);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
      glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
      glTexImage2D(GL_TEXTURE_2D,    //Sempre GL_TEXTURE_2D
            0,                       //Nível de detalhe da textura (0)
            3,                       //Número de componentes de cor
            theTexMap.GetNumCols(), //Largura
            theTexMap.GetNumRows(), //Altura
            0,                       //Bordas (deve ser 0)
            GL_RGB,                  //Formato interno: RGB
            GL_UNSIGNED_BYTE,        //Pixels armazenados como unsigned
            theTexMap.ImageData()); //A imagem/pixels da textura
}
```

Exemplo #9: Texturas

```
// desenha cubo
void drawScene(void) {
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glBindTexture(GL_TEXTURE_2D, texture[0]);

      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glTranslatef(0.0, 0.0, -5);
      glRotatef(yRotated, 0, 1, 0);
      glRotatef(zRotated, 0, 0, 1);

      glBegin(GL_QUADS);                    // Face posterior
      glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
      glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
      glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
      glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
      glEnd();
```

Exemplo #9: Texturas

```
glBegin(GL_QUADS);                          // Face frontal
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
glEnd();
glBegin(GL_QUADS);                          // Face lateral esquerda
glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
glEnd();
glBegin(GL_QUADS);                          // Face lateral direita
glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, -1.0f, -1.0f);
glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, 1.0f, 1.0f);
glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
glEnd();
```

Exemplo #9: Texturas

```
        glBegin(GL_QUADS);                      // Face superior
        glTexCoord2f(0.0f, 1.0f); glVertex3f(-1.0f, 1.0f, -1.0f);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-1.0f, 1.0f, 1.0f);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(1.0f, 1.0f, 1.0f);
        glTexCoord2f(1.0f, 1.0f); glVertex3f(1.0f, 1.0f, -1.0f);
        glEnd();
        glBegin(GL_QUADS);                      // Face inferior
        glTexCoord2f(1.0f, 1.0f); glVertex3f(-1.0f, -1.0f, -1.0f);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(1.0f, -1.0f, -1.0f);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(1.0f, -1.0f, 1.0f);
        glTexCoord2f(1.0f, 0.0f); glVertex3f(-1.0f, -1.0f, 1.0f);
        glEnd();

        glutSwapBuffers();
}
```

Exemplo #9: Texturas

```
// callback de teclado
void keyboard(unsigned char key, int x, int y) {
	switch (key) {
	case 27:
		exit(0);
		break;
	default:
		break;
	}
}

// callback idle
void idleFunc(void) {
	yRotated += 0.01;
	zRotated += 0.01;
	drawScene();
}
```

Exemplo #9: Texturas

```
int main(int argc, char** argv) {
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
      glutInitWindowSize(240, 240);
      glutInitWindowPosition(100, 100);
      glutCreateWindow("Cubo com textura");
      glClearColor(0.0, 0.0, 0.0, 0.0);
      glShadeModel(GL_FLAT);
      glEnable(GL_DEPTH_TEST);
      glEnable(GL_TEXTURE_2D);
      loadTextureFromFile(filename);
      glutDisplayFunc(drawScene);
      glutReshapeFunc(resizeWindow);
      glutKeyboardFunc(keyboard);
      glutIdleFunc(idleFunc);
      glutMainLoop();
      return 0;
}
```

Exemplo #9: Texturas

Experimente:

- Altere o arquivo com a imagem da textura: `.\textura.bmp`
- Comente a linha que habilita o teste de profundidade e reexecute o programa.

Exemplo #10: Quádricas e Stencil

```
#include <windows.h>
#include <iostream>
#include <stdlib.h>
#include <GL/glut.h>
#include "RgbImage.h"
using namespace std;
char view = 's';
bool stencil = false;
float rotationAngle;
float width = 800.0;
float height = 500.0;
char* filenameEarth = "./earthmap1k.bmp"; //image file with Earth's text.
char* filenameMetal = "./textura_metal.bmp"; //image file with metal text.
GLuint _textureIdEarth;
GLuint _textureIdMetal;
GLUquadric *quadEarth;
```

Exemplo #10: Quádricas e Stencil

```
void handleKeypress(unsigned char key, int x, int y) {
      switch (key) {
      case 27: //Escape key
            exit(0); break;
      case 't': //Top view
            view = key; glutPostRedisplay(); break;
      case 's': //Side view
            view = key; glutPostRedisplay(); break;
      case 'b': //Bottom view
            view = key; glutPostRedisplay(); break;
      case 'z': //Use stencil
            stencil = !stencil;
            glutPostRedisplay();
            break;
      }
}
```

Exemplo #10: Quádricas e Stencil

```
//Makes the image into a texture, and returns the id of the texture
GLuint loadTexture(char *filename) {
	GLuint textureId;
	RgbImage theTexMap(filename); //Image with texture
	glGenTextures(1, &textureId); //Make room for our texture
	glBindTexture(GL_TEXTURE_2D, textureId);  //Which texture to edit
	//Map the image to the texture
	glTexImage2D(GL_TEXTURE_2D,   //Always GL_TEXTURE_2D
		0,                        //0 for now
		GL_RGB,                   //Format OpenGL uses for image
		theTexMap.GetNumCols(), //Width
		theTexMap.GetNumRows(), //Height
		0,                        //The border of the image
		GL_RGB,                   //Pixels are stored in RGB format
		GL_UNSIGNED_BYTE,         //Pixels are stored as unsigned
		theTexMap.ImageData()); //The actual pixel data
	return textureId; //Returns the id of the texture
}
```

Exemplo #10: Quádricas e Stencil

```
void initRendering() {
      glEnable(GL_DEPTH_TEST);
      glEnable(GL_NORMALIZE);
      glEnable(GL_COLOR_MATERIAL);
      glEnable(GL_STENCIL_TEST);
      glEnable(GL_TEXTURE_2D);
      glClearStencil(0x0);

      quadEarth = gluNewQuadric();
      _textureIdEarth = loadTexture(filenameEarth);
      _textureIdMetal = loadTexture(filenameMetal);
}
void handleResize(int w, int h) {
      glViewport(0, 0, width, height);
      defineStencilArea();
}
```

Exemplo #10: Quádricas e Stencil

```
void defineStencilArea(void) {
      glStencilMask(0xFF); //Enables writing in stencil mask
      glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
      glClear(GL_STENCIL_BUFFER_BIT);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(-8.0, 8.0, -6.0, 6.0);
      glStencilFunc(GL_ALWAYS, 0x1, 0x1);
      glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE);
      glBegin(GL_QUADS);
      glVertex2f(-5.0, -2.0); glVertex2f(-1.0, -2.0);
      glVertex2f(-1.0, 2.0);  glVertex2f(-5.0, 2.0);
      glEnd();
      glBegin(GL_QUADS);
      glVertex2f(1.0, -2.0); glVertex2f(5.0, -2.0);
      glVertex2f(5.0, 2.0); glVertex2f(1.0, 2.0);
      glEnd();
      glStencilMask(0x00); //Disables writing in stencil mask
}
```

Exemplo #10: Quádricas e Stencil

```
void drawScene() {
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluPerspective(45.0, (float)width / (float)height, 1.0, 200.0);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      switch (view) {
      case 't': //Top view
            gluLookAt(0.0f,0.0f,20.0f,0.0f,0.0f,0.0f,0.0f,-1.0f,0.0f);
            break;
      case 's': //Side view
            gluLookAt(20.0f,0.0f,0.0f,0.0f,0.0f,0.0f,0.0f,0.0f,1.0f);
            break;
      case 'b': //Bottom view
            gluLookAt(0.0f,0.0f,-20.0f,0.0f,0.0f,0.0f,0.0f,1.0f,0.0f);
      break;
      }
```

Exemplo #10: Quádricas e Stencil

```
if (stencil) {
        //Paint the metal texture (inside spacecraft)
        glPushMatrix();
        switch (view) {
        case 't': //Top view
                glRotatef(180, 1.0f, 0.0f, 0.0f);
                break;
        case 's': //Side view
                glRotatef(-90, 0.0f, 1.0f, 0.0f);
                glRotatef(-90, 0.0f, 0.0f, 1.0f);
                break;
        }
        glBindTexture(GL_TEXTURE_2D, _textureIdMetal);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
        glStencilFunc(GL_NOTEQUAL, 0x1, 0x1); //Draw outside stencil
        glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
```

Exemplo #10: Quádricas e Stencil

```
        glBegin(GL_QUADS);
        glTexCoord2f(0.0f, 0.0f); glVertex3f(-10.0, -5.0, -1.0);
        glTexCoord2f(1.0f, 0.0f); glVertex3f( 10.0, -5.0, -1.0);
        glTexCoord2f(1.0f, 1.0f); glVertex3f( 10.0,  5.0, -1.0);
        glTexCoord2f(0.0f, 1.0f); glVertex3f(-10.0,  5.0, -1.0);
        glEnd();
        glPopMatrix();
        glStencilFunc(GL_EQUAL, 0x1, 0x1); //Draw inside stencil area
        glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
    }
    else {
        glStencilFunc(GL_ALWAYS, 0x1, 0x1); //Ignore stencil area
        glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE);
    }
```

Exemplo #10: Quádricas e Stencil

```
        // Rotate and paint the Earth
        glRotatef(rotationAngle, 0.0f, 0.0f, 1.0f);
        glBindTexture(GL_TEXTURE_2D, _textureIdEarth);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
        gluQuadricTexture(quadEarth, GLU_TRUE);
        gluSphere(quadEarth, 4, 40, 40);
        glutSwapBuffers();
}
void update(int value) {
        rotationAngle += 1.0f;
        if (rotationAngle>360.f) {
                rotationAngle -= 360;
        }
        glutPostRedisplay();
        glutTimerFunc(25, update, 0);
}
```

Exemplo #10: Quádricas e Stencil

```
int main(int argc, char** argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB|GLUT_DEPTH|GLUT_STENCIL);
	glutInitWindowSize(width, height);

	glutCreateWindow("Textured Earth");
	initRendering();
	glutTimerFunc(25, update, 0);
	glutDisplayFunc(drawScene);
	glutKeyboardFunc(handleKeypress);
	glutReshapeFunc(handleResize);

	glutMainLoop();
	return 0;
}
```

Exemplo #10: Quádricas e Stencil

Experimente:

- Altere os arquivos com as imagem de texturas.
- Em tempo de execução, pressione as teclas: “t”; “b”; “s”; e “z”.
- Quando o tamanho da janela é modificado, a máscara de stencil desaparece: como resolver este problema?

Computação Gráfica (IME 04-10842)
2022.2

Iluminação e Texturas no OpenGL

Gilson. A. O. P. Costa (IME/UERJ)

gilson.costa@ime.uerj.br