# Técnicas OpenGL

Renato Rodrigues Oliveira da Silva

Abril 2009

## Sumário

- Animação
- Iluminação e Tonalização
- Texturas

# Animação

- Animação Tradicional
  - Sequência de imagens em alta velocidade
  - Velocidade de exibição (*frame rate*) varia de acordo com a mídia utilizada
- Importante garantir que a aparência da imagem não mude muito rapidamente
  - *Aliasing* Temporal

3

# Animação - OpenGL

- Redesenhar a cena continuamente
- Cada quadro deve ser ligeiramente diferente, criando a ilusão de movimento
  - Movimento dos objetos
  - Cor
  - Forma etc
- Em 3D pode-se também alterar a posição do observados

4

## Animação - OpenGL

- É preciso forçar uma atualização contínua da tela
  - Necessário invocar função que é chamada em intervalos de tempo predeterminados
- Deve-se evitar que a imagem fique "piscando" com o contínuo redesenho
  - Utiliza-se 2 *buffers* de exibição:

**glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB)**

5

## Animação - OpenGL

- Um dos *buffers* é preenchido enquanto o outro é exibido
  - São trocados pelo comando **glutSwapBuffers()**

6

## Animação - OpenGL

- A GLUT oferece uma função *callback* que é invocada em um tempo predeterminado

```
void glutTimerFunc(int msecs,
                   void (*func)(int value),
                   int value)
```

7

## Animação - OpenGL

- void glutTimerFunc (int msecs,
  void *timer(int value),
  int value)
  - int msecs: Tempo em milisegundos
  - void *timer(int value): Função de animação
  - int value: Valor passado para a função timer

8

## Animação - OpenGL

**Código de exemplo na CoTeia**

**ExemploPongOpenGL.c**

9

## Animação - OpenGL

**Exercício**

Alterar o código **Cubo3D.c**

Fazer uma animação do cubo girando em torno de seu próprio eixo

10

# Iluminação

11

## Iluminação – Fontes de Luz

- Modelo de iluminação define a natureza de uma fonte de luz e sua interação com a cena
- Três tipos de fontes de luz podem ser incluídos em uma cena 3D
  - Luz Pontual
  - Luz Direcional
  - Luz tipo Spot

12

# Iluminação – Fontes de Luz

- Em OpenGLvoid

void glLightModelfv(GLenum pname,
TYPE *param)
// pname – característica do ponto de luz
// param – valor



# Iluminação – Fontes de Luz

```
// cor
GLfloat luzDifusa[4]={0.7,0.7,0.7,1.0};
// posicao
GLfloat posicaoLuz[4]={0.0, 50.0, 50.0, 1.0};

glLightfv(GL_LIGHT0, GL_DIFFUSE, luzDifusa );
glLightfv(GL_LIGHT0, GL_POSITION, posicaoLuz );
```

# Iluminação – Fontes de Luz

**Código de Exemplo CoTeia**

Exemplo3DComIluminacao.cpp

15

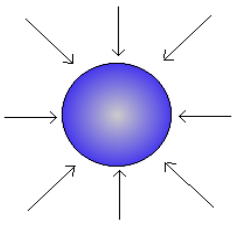
# Iluminação
# Modelos de Reflexão

- Descrevem a interação dos raios de luz com uma superfície
- Considera as propriedades da superfície e a natureza da fonte de luz incidente
  - Cor do objeto
  - Cor da fonte de luz
  - Posição da fonte de luz
  - Etc
- Modelos utilizados: reflexão ambiente, difusa e especular

16

# Iluminação
# Modelos de Reflexão

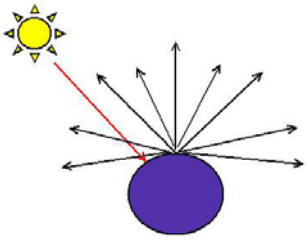
- Reflexão Ambiente
  - Considera-se a existência de uma fonte de luz não direcional
  - Origina-se da interação e da reflexão dos raios de luz com todas as superfícies da cena



17

# Iluminação
# Modelos de Reflexão

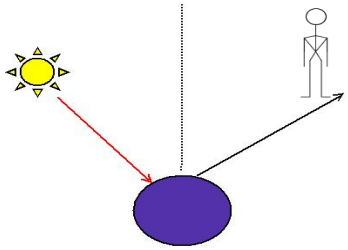
- Reflexão Difusa
  - Os objetos absorvem e refletem, em todas as direções, parte da luz incidente
  - Depende da cor do material
  - Cria o efeito de *dégradé* nos objetos



18

# Iluminação
# Modelos de Reflexão

- Reflexão Especular
  - Responsável pela geração do “ponto de brilho” dos objetos
  - Depende da cor do objeto, posição da fonte de luz e do observador



19

# Iluminação
# Modelos de Reflexão

```
// Capacidade de brilho do material
GLfloat especularidade[4]={1.0,1.0,1.0,1.0};
GLint especMaterial = 60;

// Define a refletância do material
glMaterialfv(GL_FRONT,GL_SPECULAR,
especularidade);
// Define a concentração do brilho
glMateriali(GL_FRONT,GL_SHININESS,
especMaterial);
```

20

# Iluminação
# Modelos de Reflexão

**Código de Exemplo CoTeia**

Exemplo3DComIluminacao.c

(alterar parâmetros da fonte de luz)

21

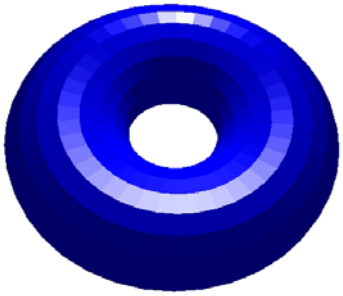
# Iluminação
# Modelos de Tonalização

- Determina a cor de cada ponto que compõe a superfície de um objeto
- Define a variação de cor ao longo das faces dos objetos da cena
- Depende das propriedades da superfície e da iluminação
- A OpenGL suporta os modelos Flat e Gourand

22

# Iluminação
# Modelos de Tonalização

- Flat Shading
  - É o modelo mais simples (+ rápido)
  - Cada face tem apenas um valor de intensidade



23

# Iluminação
# Modelos de Tonalização

- Gourand Shading
  - Exibe uma transição suave entre as faces que compõem os objetos
  - Interpola as intensidades das cores ao longo de cada face

24

## Iluminação
## Modelos de Tonalização

- Em OpenGL

void glShadeModel (Glenum mode)

// mode = GL_FLAT ou GL_SMOOTH (Gourand)



## Iluminação
## Modelos de Tonalização

**Código de Exemplo Coteia**

Exemplo3DComIluminacao.cpp

(alterar modelo de tonalização)

# Texturas

27

## Texturas

- O termo refere-se a uma imagem bidimensional que é “colada” à superfície de um objeto
- Podem ter uma, duas ou três dimensões

28

## Texturas

- Inicialmente deve-se gerar uma identificação única para a textura

```
void glGenTextures(GLsizei n,
                   GLuint *textures)
// n – quantos identificadores devem ser gerados
//textures – vetor de inteiros que guardam o valor dos identificadores gerados
```

29

## Texturas

- A seguir deve-se especificar qual é a textura corrente

```
void glBindTexture (GLenum target,
                    GLuint texture)
//target – indica o tipo de textura (para 2d, GL_TEXTURE_2D)
//texture – identificador da textura
```

30

## Texturas

- Especificar o modo de aplicação da textura corrente

```
void glTexEnvi(GLenum target,
               GLenum pname,
               Glint param);
// target = GL_TEXTURE_ENV
// pname = GL_TEXTURA_ENV_MODE
// param = GL_MODULATE (iluminação),
   GL_REPLACE (sem iluminação)...
```

31

## Texturas

- Pode-se utilizar a biblioteca **bibutil** para facilitar
- Possui uma estrutura para especificar os parâmetros das imagens
- Possui funções para aplicar os parâmetros

32

## Texturas

- Estrutura TEX:

```
typedef struct {
    int ncomp; // numero de componentes
    int dimx; // largura
    int dimy; // altura
    Gluint texid; //identificação
    unsigned char *data; // conteúdo imagem
} TEX;
```

33

## Texturas

- Função para carregar a imagem na memória

```
TEX* CarregaTextura(char *path,
                    boolean mipmaps);
// path = caminho para o arquivo
// mipmaps = indica se fará ou não o uso
   da técnica de mipmaps (qualidade)
```

34

## Texturas

- Deve-se também ativar a variável de estado GL_TEXTURE_2D
  - glEnable(GL_TEXTURE_2D)
- O mapeamento é realizado pela associação das coordenadas de textura a cada vértice da superfície
  - De forma semelhante à associação de cores por meio de glColor3f

35

## Texturas

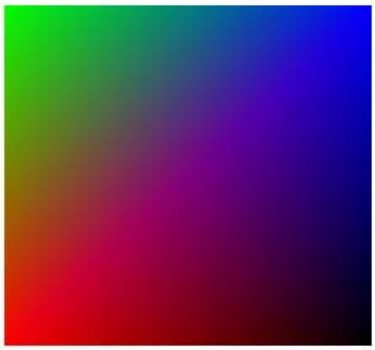
- Mapeamento de quadrilátero com cores

```
glBegin(GL_QUADS)
   glColor3f(1.0f, 0.0f, 0.0f); //vermelho
   glVertex2f(-45.0f, -15.0f);
   glColor3f(0.0f, 1.0f, 0.0f); //verde
   glVertex2f(-45.0f, 15.0f);
   glColor3f(0.0f, 0.0f, 1.0f); //azul
   glVertex2f(-15.0f, 15.0f);
   glColor3f(0.0f, 0.0f, 0.0f); //preto
   glVertex2f(-15.0f, -15.0f);
glEnd();
```

36

## Texturas

- Mapeamento de quadrilátero com cores



37

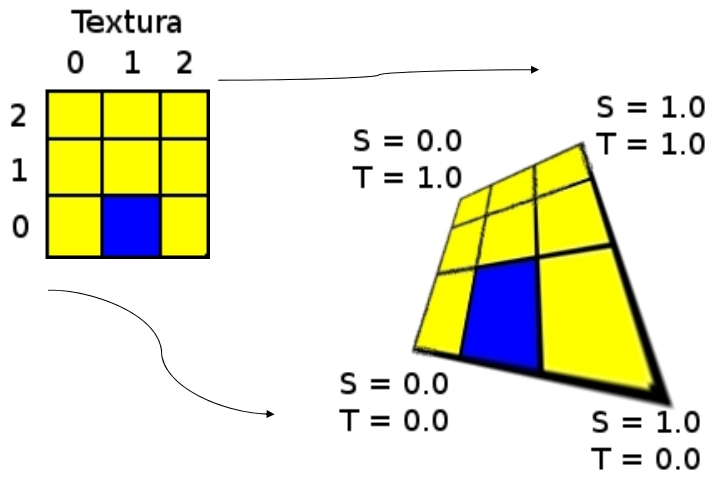
## Texturas

- void glTexCoord2f (GLfloat s, GLfloat t)

// Coordenadas da textura a serem associadas com o vértice corrente

38

# Texturas




39

# Texturas

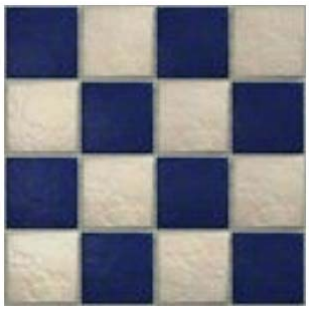
- Mapeamento de quadrilátero com textura

```
glBegin(GL_QUADS);
   glTexCoord2f(0,0);
   glVertex3f(-1,-1,0);
   glTexCoord2f(1,0);
   glVertex3f(1,-1,0);
   glTexCoord2f(1,1);
   glVertex3f(1, 1,0);
   glTexCoord2f(0,1);
   glVertex3f(-1, 1,0);
- glEnd();
```

40

# Texturas

- Mapeamento de quadrilátero com textura





# Texturas

**Código de Exemplo CoTeia**

**ExemploTextura2D.cpp**

## Texturas

**Exercício**

Alterar o código **Cubo3DColorido.c** para utilizar texturas

43

## Bibliografia


- Cohen, M. , Manssour, I. H, OpenGL – Uma Abordagem Prática e Objetiva, 2006, Novatec Editora


44