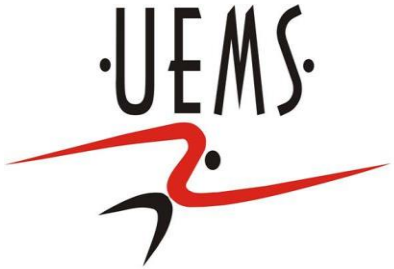
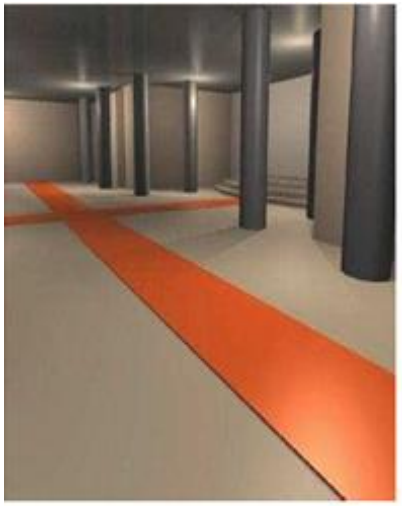


Computação Gráfica – Textura

Profa. Mercedes Gonzales Márquez

Textura

- Modelos de iluminação não são suficientes para descrever todas as características observáveis em uma superfície, usa-se uma técnica chamada *mapeamento de textura,* que é mais eficiente do que usar apenas geometria.



Textura

- Estes valores pré-computados são tipicamente organizados em um arranjo multidimensional de *texels (texture elements)* em um espaço próprio, denominado espaço de textura.

- A idéia básica é reproduzir sobre a superfície do objeto as propriedades de alguma função ou mapeamento bidimensional.

Textura

- mapeamento de uma textura proporcionando sensação de um tampo de mármore.

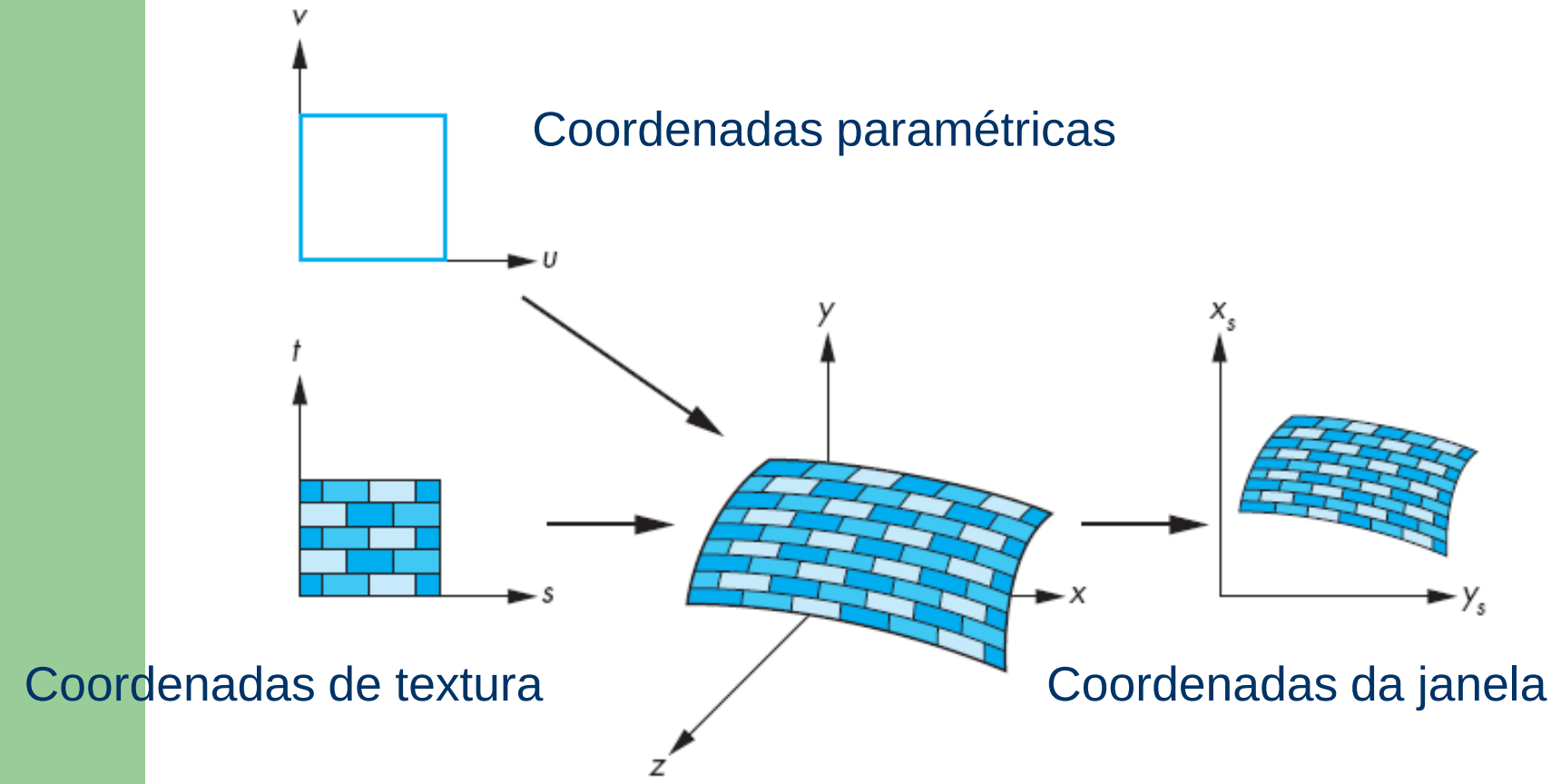
Mapeamento de textura

- O estudo de mapeamentos possui três aspectos distintos e complementares:
  - Criação dos objetos a serem mapeados (ex. textura).
  - Desenvolvimento das técnicas de mapeamento.
  - Cálculo do mapeamento

Criação de texturas

- A criação de texturas exige uma combinação de processos científicos elaborados e uma boa dose de talento artístico . Essencialmente existem 3 métodos para criação de texturas:
  - Escaneamento de imagens reais
  - Síntese a partir de imagens reais
  - Através de algoritmos

Espaços envolvidos na textura



Função de Mapeamento

- Problema básico é como encontrar os mapas.
  - Considere um mapeamento a partir das coordenadas de textura a um ponto da superfície.
  - Precisariamos de três funções:
    - x = x(s,t)
    - y = y(s,t)
    - z = z(s,t)
- Mas, na verdade precisamos da inversa.

Função de Mapeamento inverso

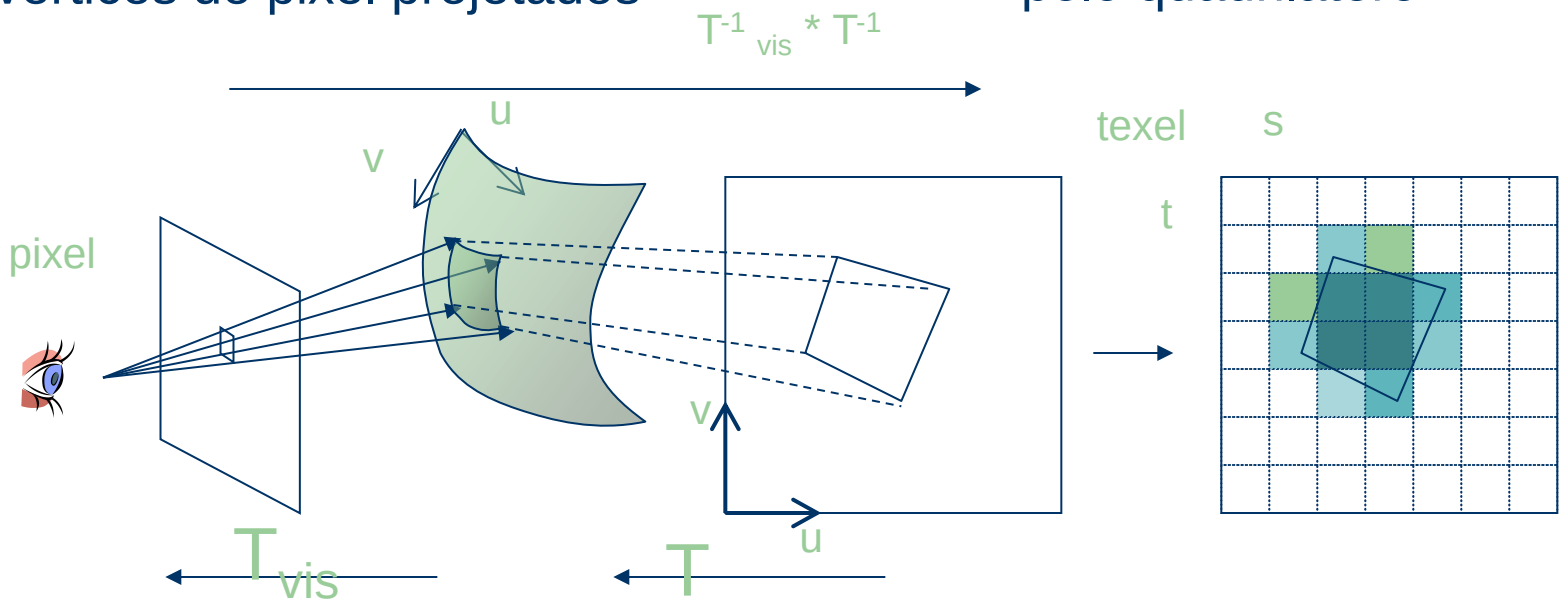
- Dado um pixel, nós precisamos saber qual ponto no objeto corresponde a ela.
- Dado um ponto no objeto, nós precisamos saber qual ponto na textura corresponde a ele.
  - Precisamos de um mapa da forma:
    - s = s(x,y,z)
    - t = t(x,y,z)
- Tal mapeamento é difícil de encontrar.

Função de Mapeamento inverso

- Dado um pixel, nós precisamos saber qual ponto no objeto corresponde a ela.
- Dado um ponto no objeto, nós precisamos saber qual ponto na textura corresponde a ele.
  - Precisamos de um mapa da forma:
    - s = s(x,y,z)
    - t = t(x,y,z)
- Tal mapeamento é difícil de encontrar.

Processo de Mapeamento de Texturas

- Projeção do pixel sobre a superfície
  - Pontos da superfície correspondentes aos vértices do pixel
- Parametrização
  - Coordenadas paramétricas dos vértices do pixel projetados
- Mapeamento inverso
  - Coordenadas dos vértices no espaço de textura
- Média
  - Cor média dos “Texels” proporcional à área coberta pelo quadrilátero



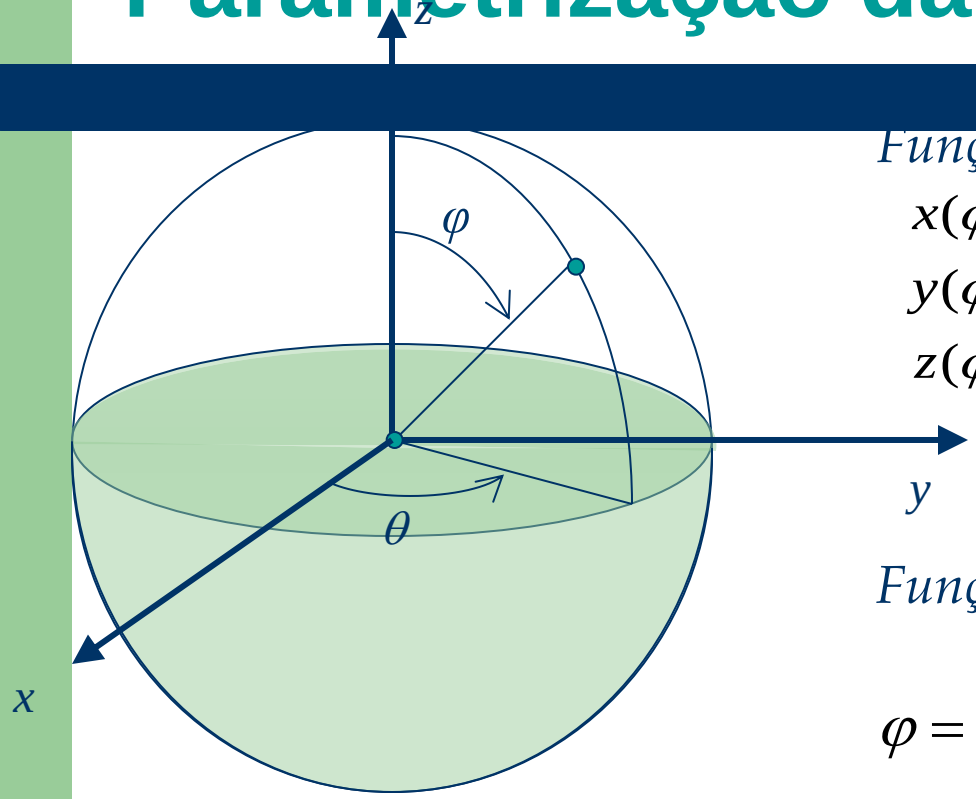
Função de Mapeamento

- Retorna o ponto do objeto correspondente a cada ponto do espaço de textura

  $(x, y, z) = F(s, t)$

- Corresponde à forma com que a textura é usada para “embrulhar” (*wrap*) o objeto
  - Na verdade, na maioria dos casos, precisamos de uma função que nos permita “desembrulhar” (*unwrap*) a textura do objeto, isto é, a inversa da função de mapeamento
- Se a superfície do objeto pode ser descrita em forma paramétrica esta pode servir como base para a função de mapeamento

Parametrização da Esfera



$$\varphi = \pi \cdot t$$

$$\theta = 2\pi \cdot s$$

Função de mapeamento

$$x(\varphi, \theta) = \sin\varphi \cos\theta$$

$$y(\varphi, \theta) = \sin\varphi \sin\theta$$

$$z(\varphi, \theta) = \cos\varphi$$

Função de mapeamento inversa

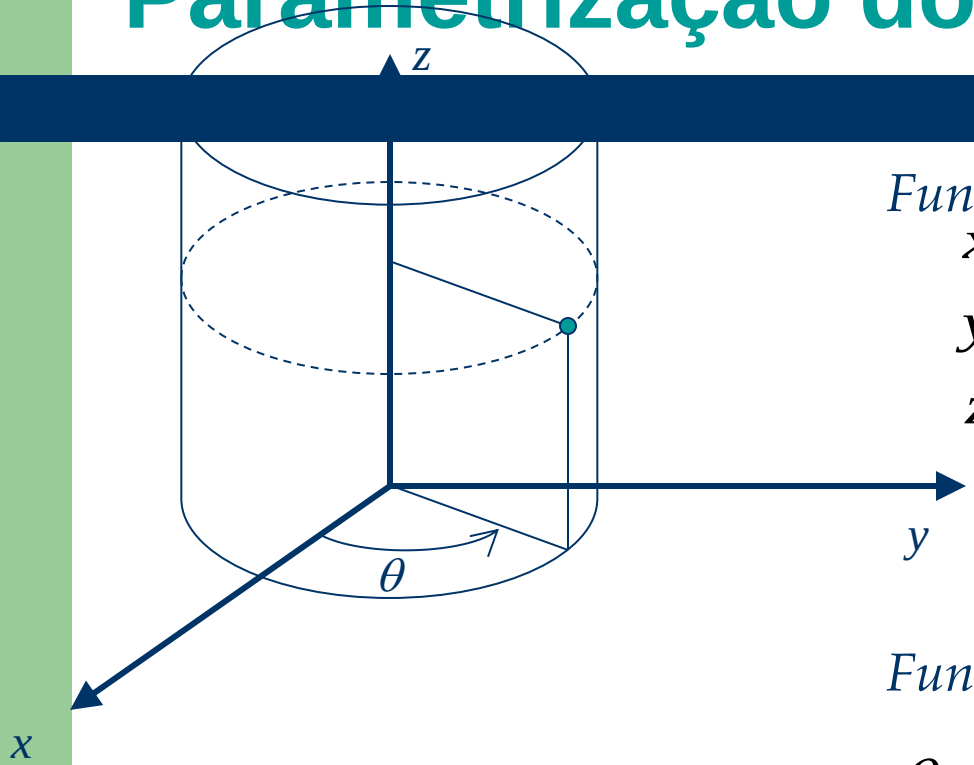
$$\varphi = \arccos z$$

$$\theta = \arctan\frac{y}{x}$$

$$t = \frac{\arccos z}{\pi}$$

$$s = \frac{\arctan\frac{y}{x}}{2\pi}$$

Parametrização do Cilindro



$$\theta = 2\pi \cdot s$$

Função de mapeamento

$$x = \cos\theta$$

$$y = \sin\theta$$

$$z = z$$

$$z = t$$

Função de mapeamento inversa

$$\theta = \arctan\frac{y}{x}$$

$$z = z$$

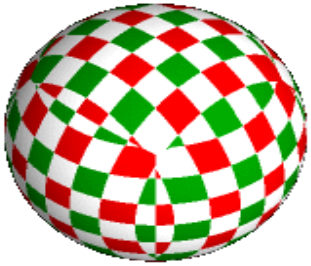
$$s = \frac{\theta}{2\pi}$$

$$t = z$$

Exemplos

Parametrização cúbica

Projetada em uma esfera



Projetada em um cilindro

Exemplos

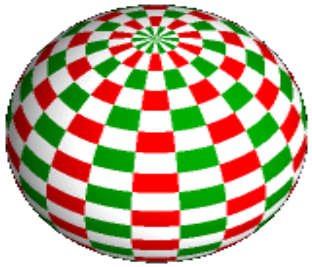
Parametrização cilíndrica

Projetada em uma esfera

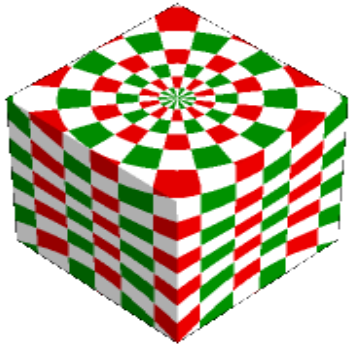
Projetada em um cubo

Exemplos

Parametrização esférica



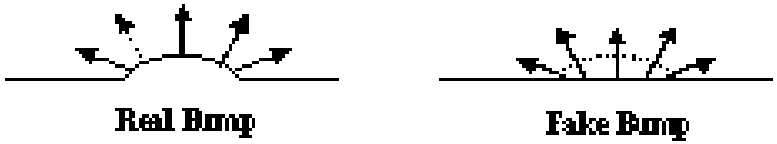
Projetada em um cubo



Projetada em um cilindro

Bump mapping

- Mapeamento básico de textura pinta numa superfície suave.
- Nesta técnica a imagem mapeada é utilizada para fazer uma perturbação do vetor normal à superfície antes de calcular a iluminação, resultando em um efeito visual de superfície rugosa.



- Supondo que a imagem é dada por b(u,v), a superfície por p(u,v) e a normal N(u,v) temos que o vetor perturbado é dado por:

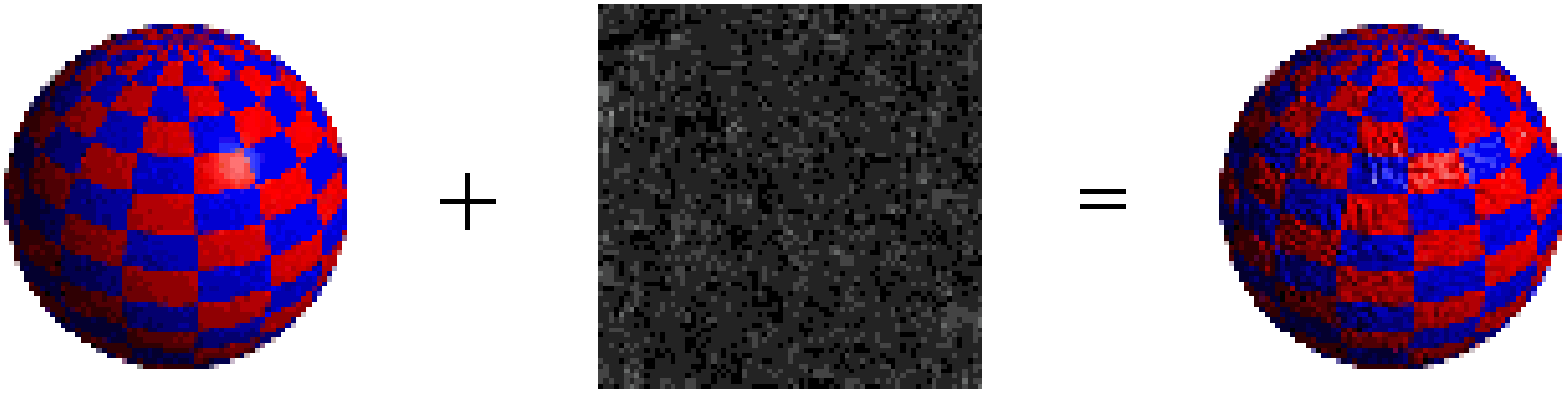
$$q(u,v) = p(u,v) + b(u,v) * N(u,v)$$

$$N_1 = \frac{\partial q}{\partial u} \times \frac{\partial q}{\partial v}$$

$$N = \frac{N1}{|N1|}$$

Bump mapping

- A superfície não muda realmente, sombreamento faz parecer mudada.

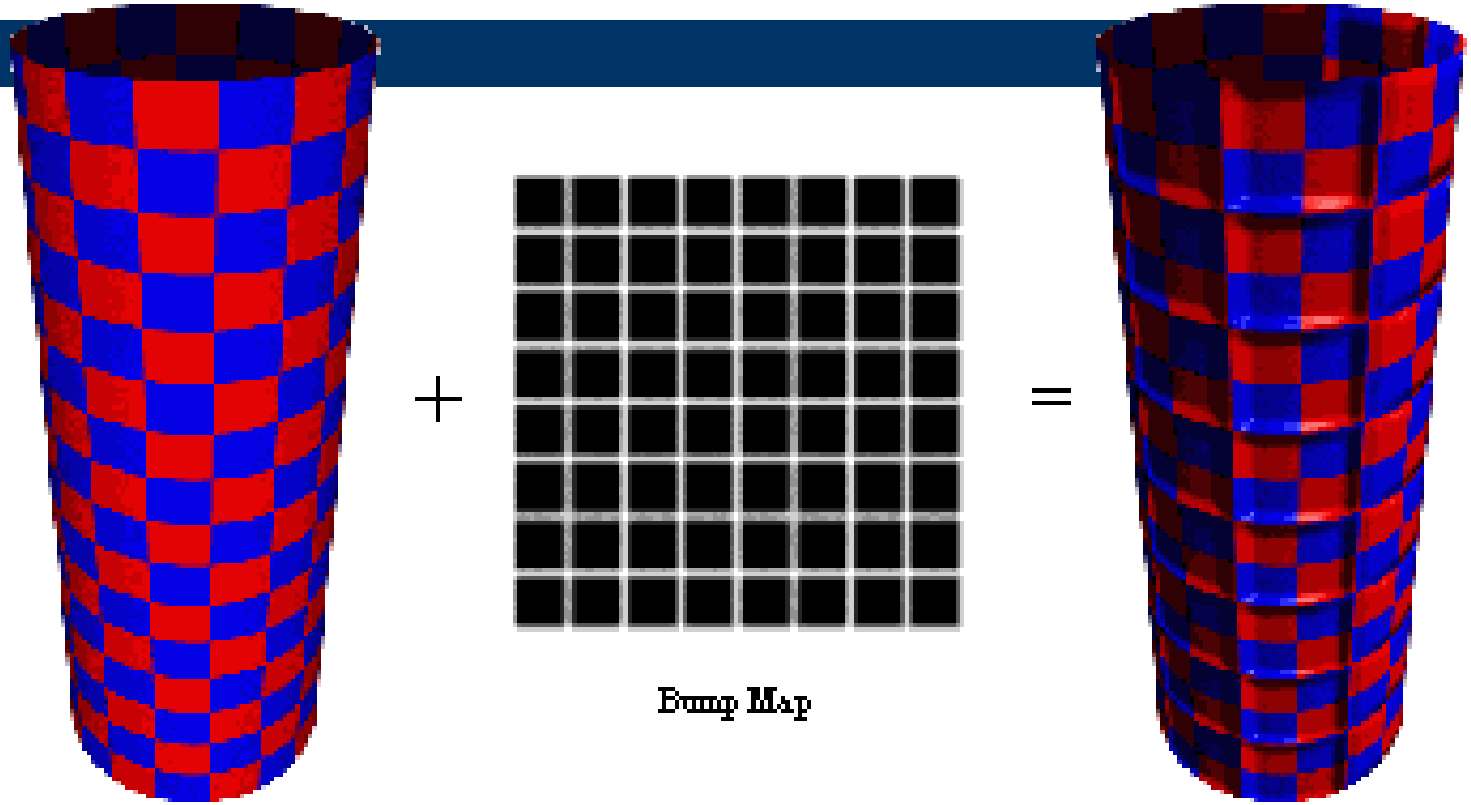


Esfera com mapa de texturas difuso

Bump map

Esfera com mapa de texturas difuso + bump map

Exemplo de “Bump mapping”



Cilindro c/ mapa de texturas difuso

Cilindro c/ mapa de texturas difuso + bump map

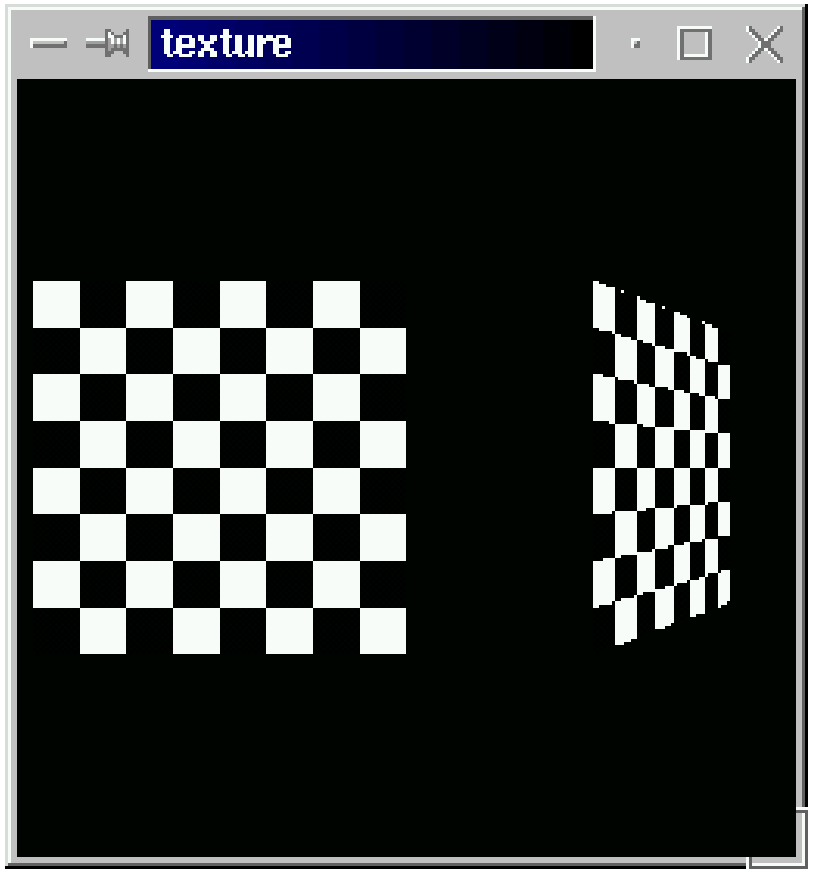
Mapa de deslocamentos (displacement mapping)

- Uma desvantagem do mapeamento da rugosidade é o que ao observarmos a silhueta da superfície não vemos os detalhes da geometria que foram mapeados.
- Uma solução consiste em deslocar realmente a superfície
- Uso do mapa de texturas para deslocar cada ponto na superfície
  - valor de textura diz quanto mover na direção normal à superfície

Aplicando textura em OpenGL

- Três passos
  1. Especificar textura
     - Ler ou gerar a imagem
     - Carregar a textura
  2. Mapear coordenadas da textura a coordenadas de vértices
  2. Especificar parâmetros de textura
     - Embrulhamento e filtragem

Resultado



Exemplo: tabuleiro

```
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glut.h>
#include <stdlib.h>
#include <stdio.h>

/*  Create checkerboard texture  */
#define checkImageWidth 64
#define checkImageHeight 64
static GLubyte checkImage[checkImageHeight][checkImageWidth][4];
static GLuint texName;
```

Cria textura para o tabuleiro

```
void makeCheckImage(void)  {
   int i, j, c;

   for (i = 0; i < checkImageHeight; i++) {
      for (j = 0; j < checkImageWidth; j++) {
         c = ((((i&0x8)==0)^((j&0x8))==0))*255;
         checkImage[i][j][0] = (GLubyte) c;
         checkImage[i][j][1] = (GLubyte) c;
         checkImage[i][j][2] = (GLubyte) c;
         checkImage[i][j][3] = (GLubyte) 255;
      }
   }
}
```

Inicializa parâmetros de textura

```
void init(void) {
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel(GL_FLAT);
   glEnable(GL_DEPTH_TEST);
   makeCheckImage();
   glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
   glGenTextures(1, &texName);
   glBindTexture(GL_TEXTURE_2D, texName);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
   glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, checkImageWidth, checkImageHeight, 0,
                GL_RGBA, GL_UNSIGNED_BYTE, checkImage);
}
```

Mostra o tabuleiro

```
void display(void) {
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glEnable(GL_TEXTURE_2D);
   glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
   glBindTexture(GL_TEXTURE_2D, texName);
   glBegin(GL_QUADS);
   glTexCoord2f(0.0, 0.0); glVertex3f(-2.0, -1.0, 0.0);
   glTexCoord2f(0.0, 1.0); glVertex3f(-2.0, 1.0, 0.0);
   glTexCoord2f(1.0, 1.0); glVertex3f(0.0, 1.0, 0.0);
   glTexCoord2f(1.0, 0.0); glVertex3f(0.0, -1.0, 0.0);
   glTexCoord2f(0.0, 0.0); glVertex3f(1.0, -1.0, 0.0);
   glTexCoord2f(0.0, 1.0); glVertex3f(1.0, 1.0, 0.0);
   glTexCoord2f(1.0, 1.0); glVertex3f(2.41421, 1.0, -1.41421);
   glTexCoord2f(1.0, 0.0); glVertex3f(2.41421, -1.0, -1.41421);
   glEnd();    glFlush();
   glDisable(GL_TEXTURE_2D);
}
```

Muda a forma da janela
Trata evento de teclado

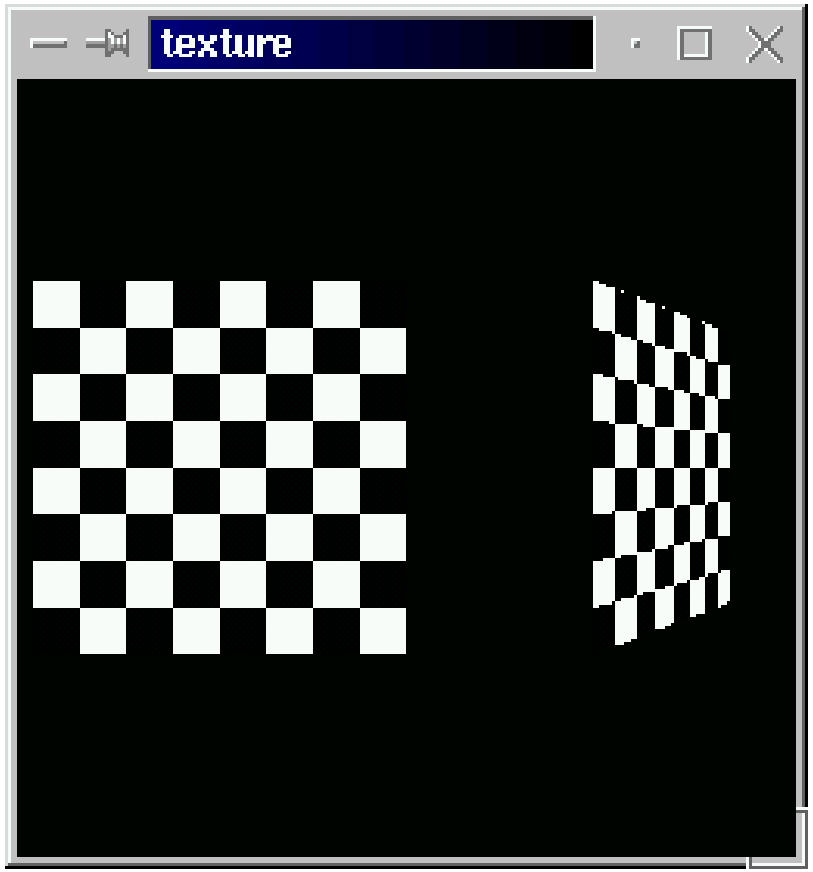
```
void reshape(int w, int h){
   glViewport(0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(60.0, (GLfloat) w/(GLfloat) h,
1.0, 30.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glTranslatef(0.0, 0.0, -3.6);
}
```

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

Rotina principal

```
int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize(250, 250);
   glutInitWindowPosition(100, 100);
   glutCreateWindow(argv[0]);
   init();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutKeyboardFunc(keyboard);
   glutMainLoop();
   return 0;
}
```

Resultado



Entendendo melhor

- No programa checker a textura é gerada ou carregada ?
- Qual é o trecho de código do programa checker.c que especifica a textura? Como a textura é especificada?

  Dica: Altere as definições das variáveis c e checkImage no procedimento makeCheckImage.
- Modifique o padrão de textura criando o seu próprio.

Entendendo melhor

- Em display():
  - glEnable() habilita uso de textura.
  - glTexEnv*() coloca o modo de desenho em GL_DECAL (polígonos são desenhados usando cores do mapa de textura, ao invés de considerar qual a cor que os polígonos deveriam ser desenhados sem a textura).
  - Dois polígonos são desenhados (note que as coordenadas de textura são especificadas com as coordenadas de vértices).
  - glTexCoord*() é similar a glNormal() (determina normais).
  - glTexCoord*() acerta as coordenadas de textura correntes; qualquer vértice subsequente terá aquelas coordenadas de textura associadas com ele até que glTexCoord*() seja chamada novamente.

Aplicando texturas (detalhe)

- Especificar texturas em objetos de textura
- Setar filtros de textura
- Setar funções de textura
- Setar o modo de embrulhamento da textura
- Criar/Ligar objeto de textura
- Habilitar textura
- Especificar coordenadas de textura
  - Coordenadas podem também ser geradas

Objetos de Textura

- Uma imagem por objeto de textura
- Gera nomes de textura

```
glGenTextures( n, *texIds );
```

- Cria objetos de textura com dados de textura

```
glBindTexture( target, id );
```

- Liga texturas antes de usá-las

```
glBindTexture( target, id );
```

Especificando Imagem de Textura

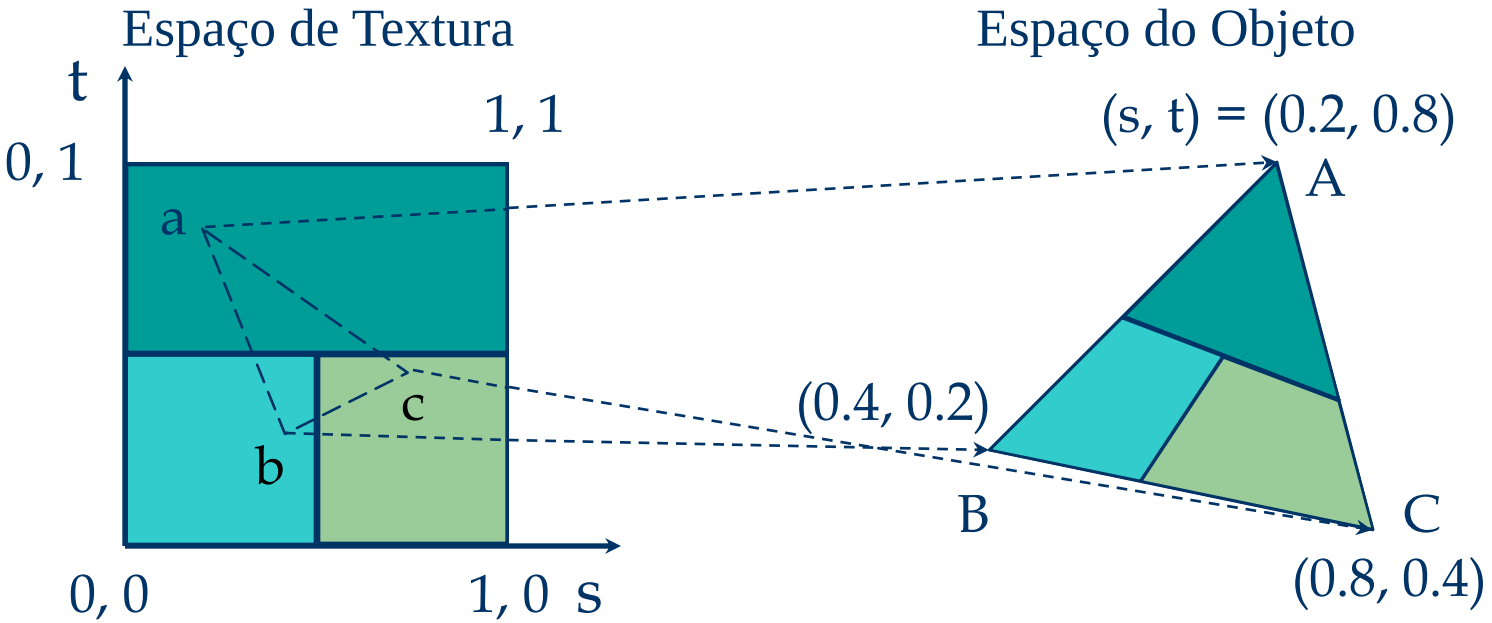
- Define uma imagem de textura a partir de um arranjo de texels

```
glTexImage2D( target, level, components,
    w, h, border, format, type, *texels );
```

  - Dimensão da imagem deve ser potência de 2

Mapeando a Textura

- Baseado em coordenadas paramétricas de textura
- Chamar `glTexCoord*()` para cada vértice



Modos de aplicação de texturas

- Modo de filtro
  - minificação ou magnificação
  - Filtros especiais para Mipmap
  - Modos de embrulhamento (clamp ou repeat)
- Funções de textura
  - Como misturar a cor primitiva com a cor da textura
    - blend, modulate ou replace texels

Filtragem

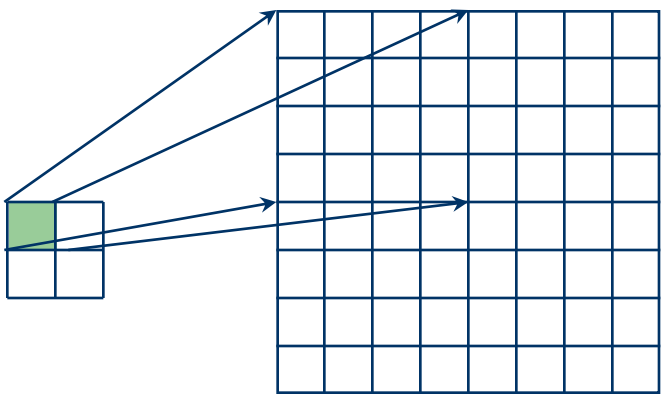
GL_TEXTURE_2D
GL_TEXTURE_1D

GL_TEXTURE_MAG_FILTER
GL_TEXTURE_MIN_FILTER

GL_NEAREST
GL_LINEAR
GL_NEAREST_MIPMAP_NEAREST
GL_NEAREST_MIPMAP_LINEAR
GL_LINEAR_MIPMAP_NEAREST
GL_LINEAR_MIPMAP_LINEAR

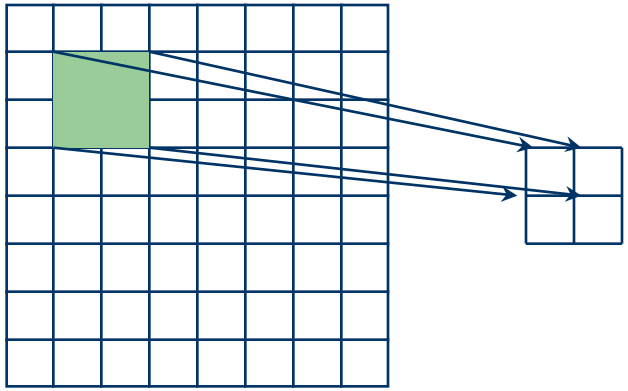
Exemplo:

```
glTexParameteri( target, type, mode );
```



Textura Polígono

Magnificação

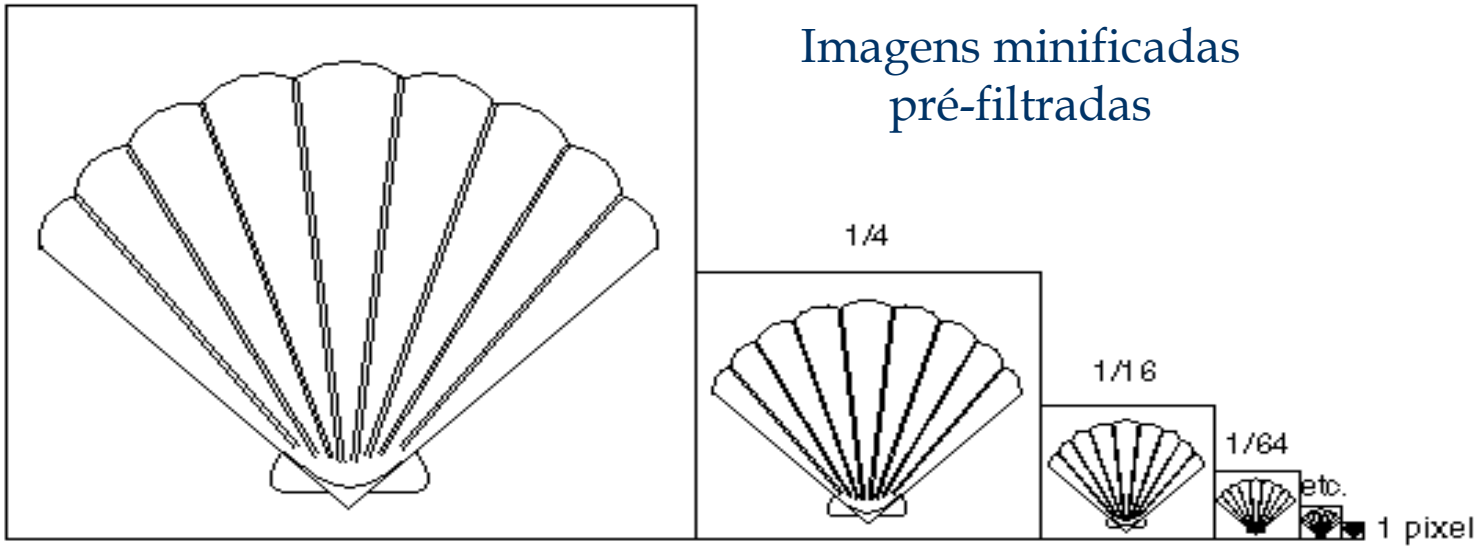


Textura Polígono

Minificação

Texturas Mipmap

Textura original



Texturas Mipmap

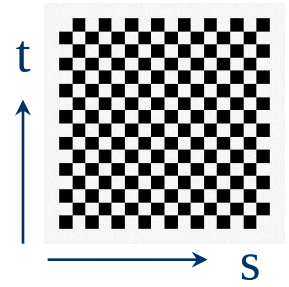
- Permite que texturas de diferentes níveis de resolução sejam aplicadas de forma adaptativa
- Reduz aliasing devido a problemas de interpolação
- O nível da textura na hierarquia mipmap é especificada durante a definição da textura

```
glTexImage*D( GL_TEXTURE_*D, level, … )
```

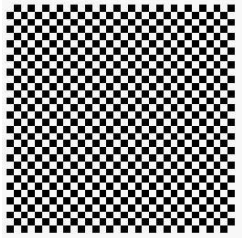
Modos de Repetição

- Exemplo:

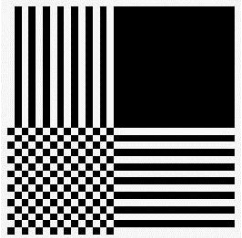
```
glTexParameteri( GL_TEXTURE_2D,
     GL_TEXTURE_WRAP_S, GL_CLAMP )
glTexParameteri( GL_TEXTURE_2D,
     GL_TEXTURE_WRAP_T, GL_REPEAT )
```



textura



GL_REPEAT



GL_CLAMP

Funções de Textura

- Controla como a cor da textura afeta a cor do pixel

  `glTexEnv{fi}[v](`*`GL_TEXTURE_ENV, prop, param`*` )`
- Modos (*prop* = *`TEXTURE_ENV_MODE`*)
  - `GL_MODULATE`
  - `GL_BLEND`
  - `GL_REPLACE`
- Cor a ser misturada (`GL_BLEND`)
  - Especificada com *prop* = *`GL_TEXTURE_ENV_COLOR`*

Exemplos de textura

Exemplos:

- loadTextures.cpp (carrega image textura externa lauch.bmp e cria uma textura sintética xadrez) e alternadamente as mapeia em um retângulo.
- fieldAndSky.cpp (textura de grama e textura de céu mapeadas em um retângulo horizontal e vertical respectivamente).
- fieldAndSkyFiltered (fieldAndSky com Filtro MipMap)
- texturedTorus.cpp (superfície paramétrica torus com textura).
- texturedTorpedo.cpp (torpedo com partes da sua geometria modelada com superfícies bézier com textura e com quádricas (cilindro) texturizada.

Exemplos de textura

Exemplos:

- fieldAndSkyLit.cpp (Este programa, baseado no fieldAndSky.cpp adiciona uma fonte de luz direcional (sol) cuja direção e intensidade podem ser controladas).