MAC420/5744: Introdução à Computação Gráfica

Marcel P. Jackowski
mjack@ime.usp.br

Aula #14: Mapeamento de textura

Some slides by Steve Marschner, Cornell CS4620/5620

Primeira definição

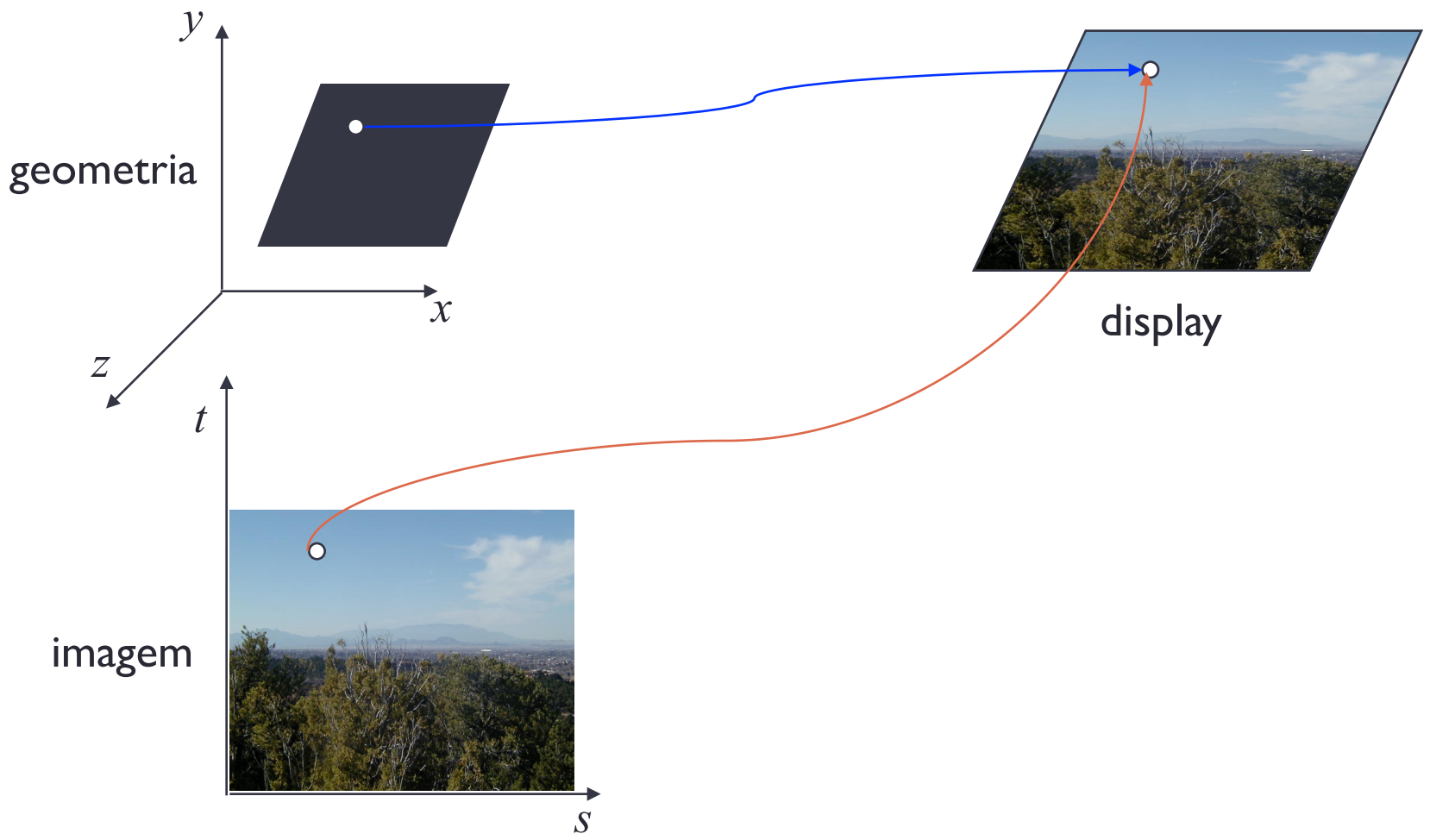
Texture mapping: a technique of defining surface properties (especially shading parameters) in such a way that they vary as a function of position on the surface.

Definição final

Texture mapping: a set of techniques for defining functions on surfaces, for a variety of uses.

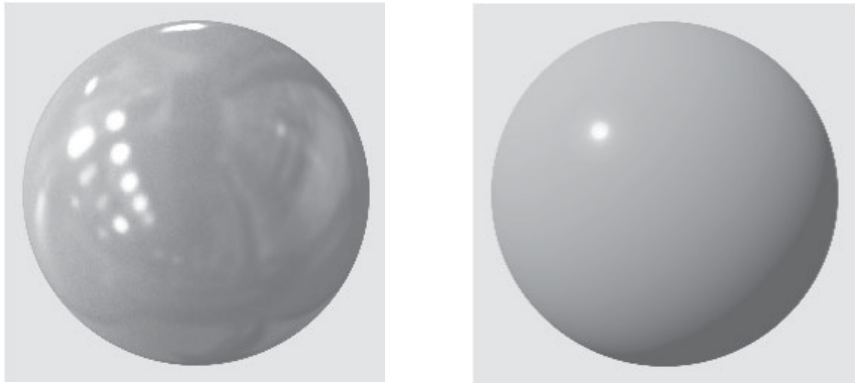
- Let's look at some examples of more general uses of texture maps.

Mapeamento de textura



Mapeamento de reflexão

- Early (earliest?) non-decal use of textures
- Appearance of shiny objects
  - Phong highlights produce blurry highlights for glossy surfaces.
  - A polished (shiny) object reflects a sharp image of its environment.
- The whole key to a shiny-looking material is providing something for it to reflect.



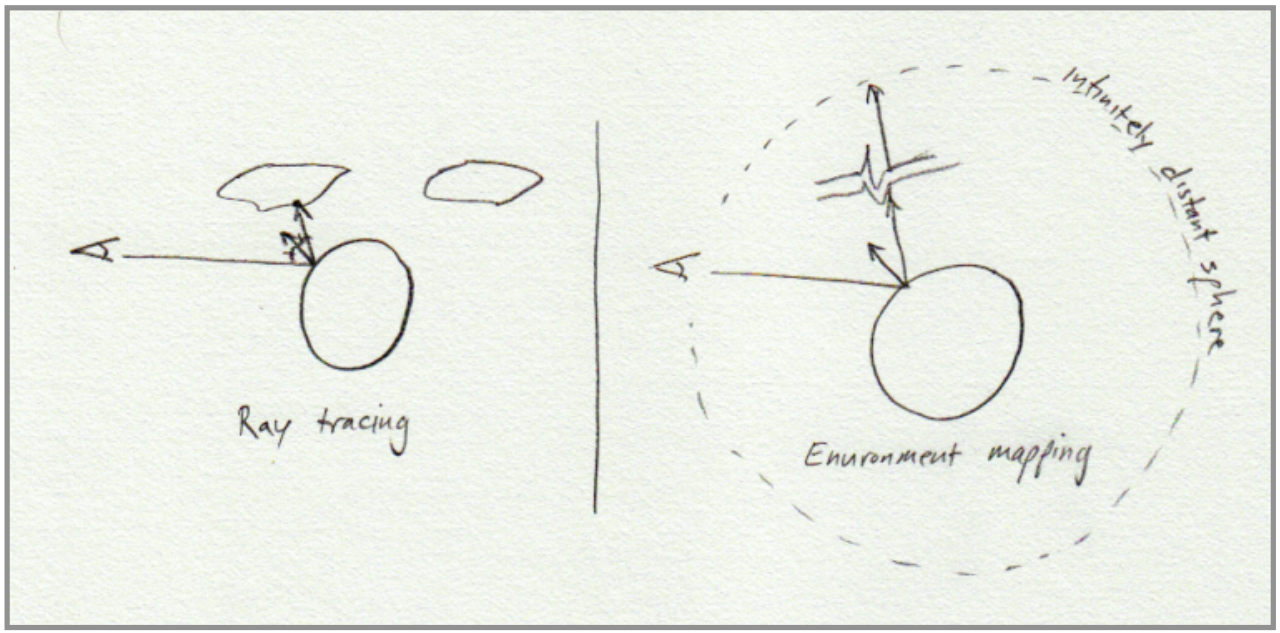
(a) (b)

[Dror, Willsky, & Adelson 2004]

Figure 2. (a). A shiny sphere rendered under photographically acquired real-world illumination. (b). The same sphere rendered under illumination by a point light source.

Mapeamento de reflexão

- From ray tracing we know what we'd like to compute
  - trace a recursive ray into the scene—too expensive
- If scene is infinitely far away, depends only on direction
  - a two-dimensional function

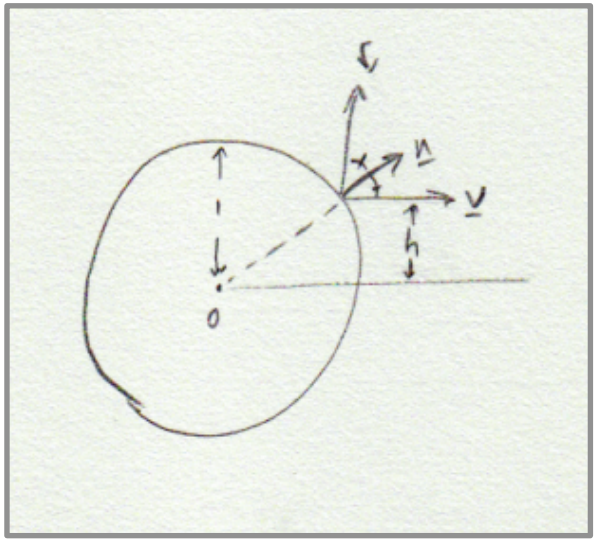


Mapeamento de ambiente

- A function from the sphere to colors, stored as a texture.

[Blinn & Newell 1976]

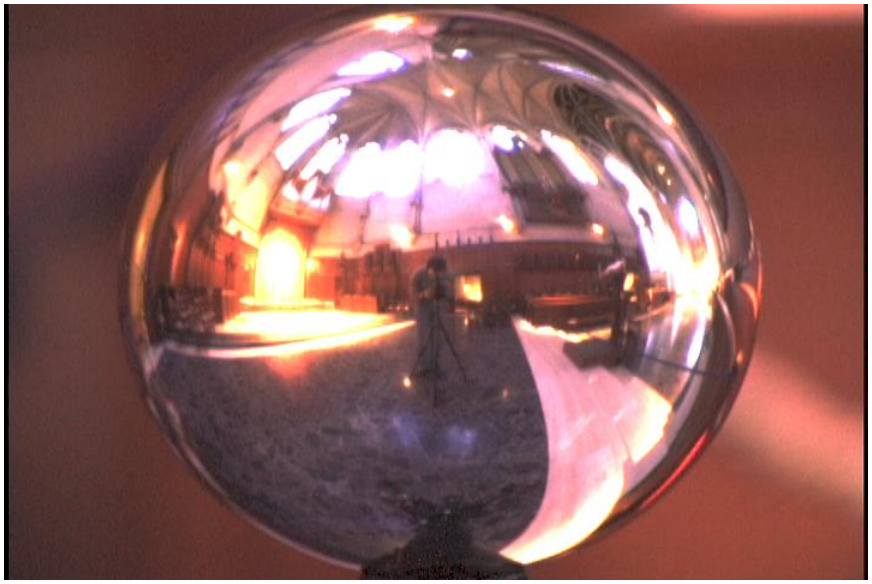
Mapeamento esférico



Mapas de ambiente

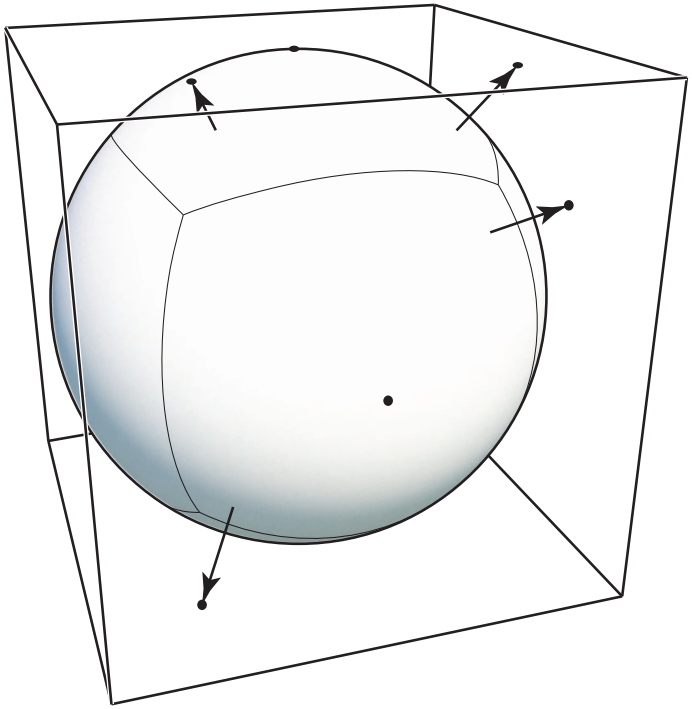
[Paul Debevec]

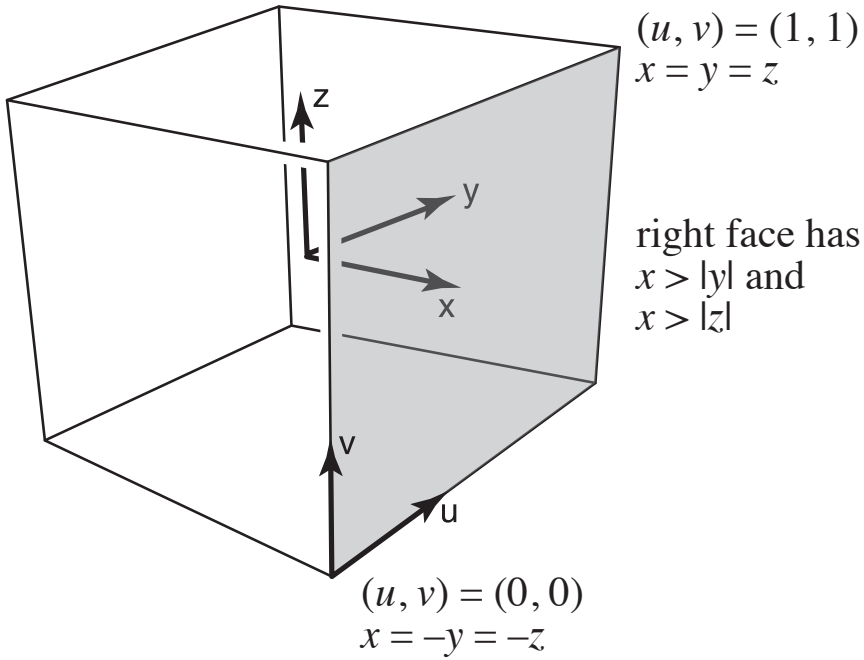
Light probes



http://ict.debevec.org/~debevec/HDRShop/tutorial/tutorial5.html#part1

Mapeamento cúbico



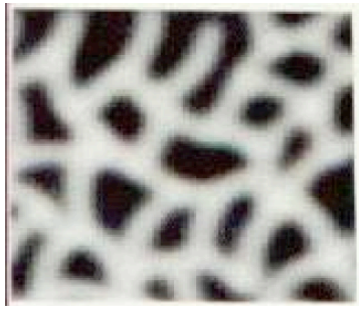


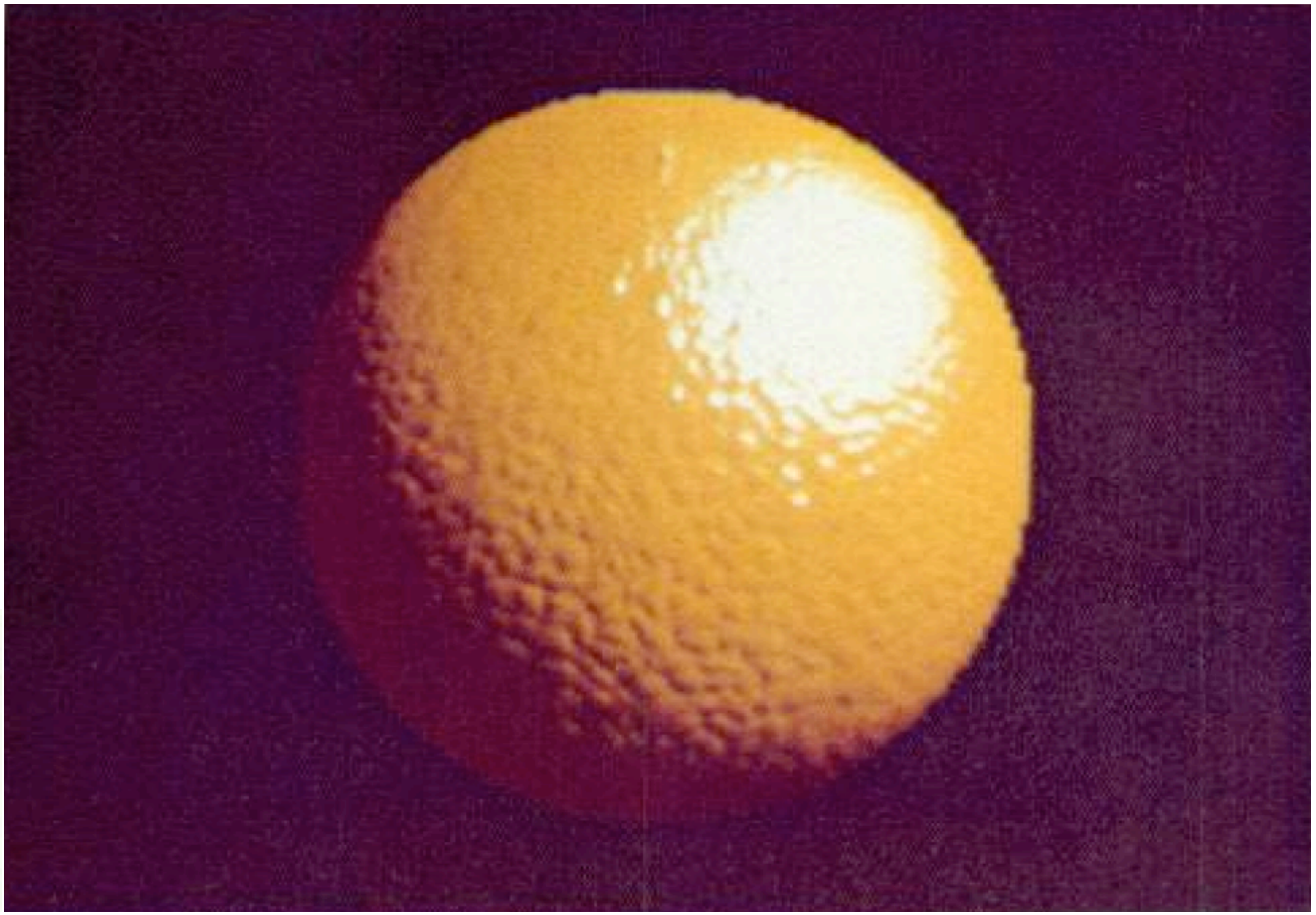
a direction vector maps to the point on the cube that is along that direction.
The cube is textured with 6 square texture maps.

Mapeamento cúbico

[Emil Persson]

Bump mapping

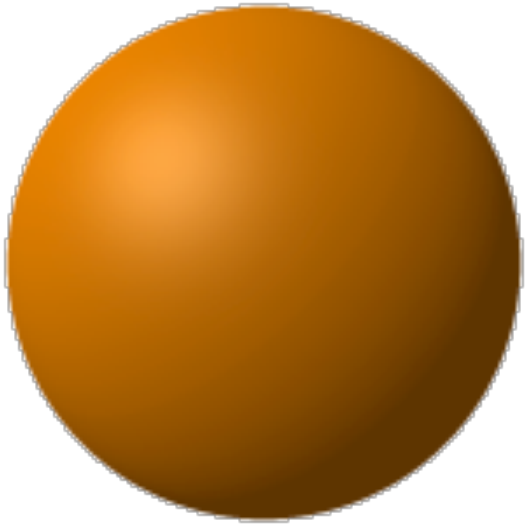




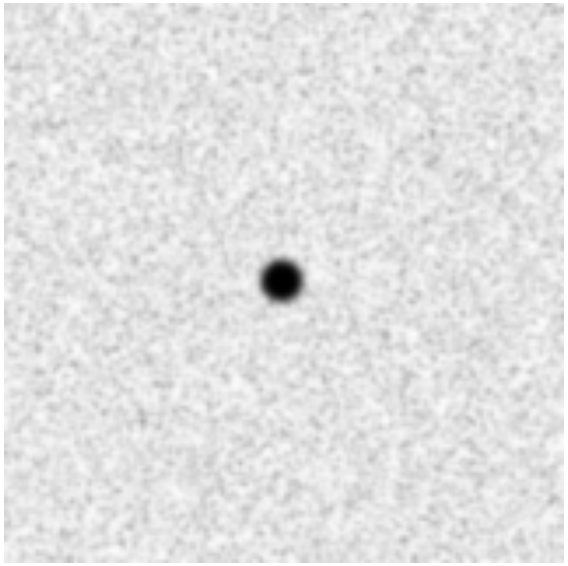
[Blinn 1978]

Mapeamento de rugosidade

- A imagem a ser mapeada é utilizada para fazer uma perturbação do vetor normal à superfície.



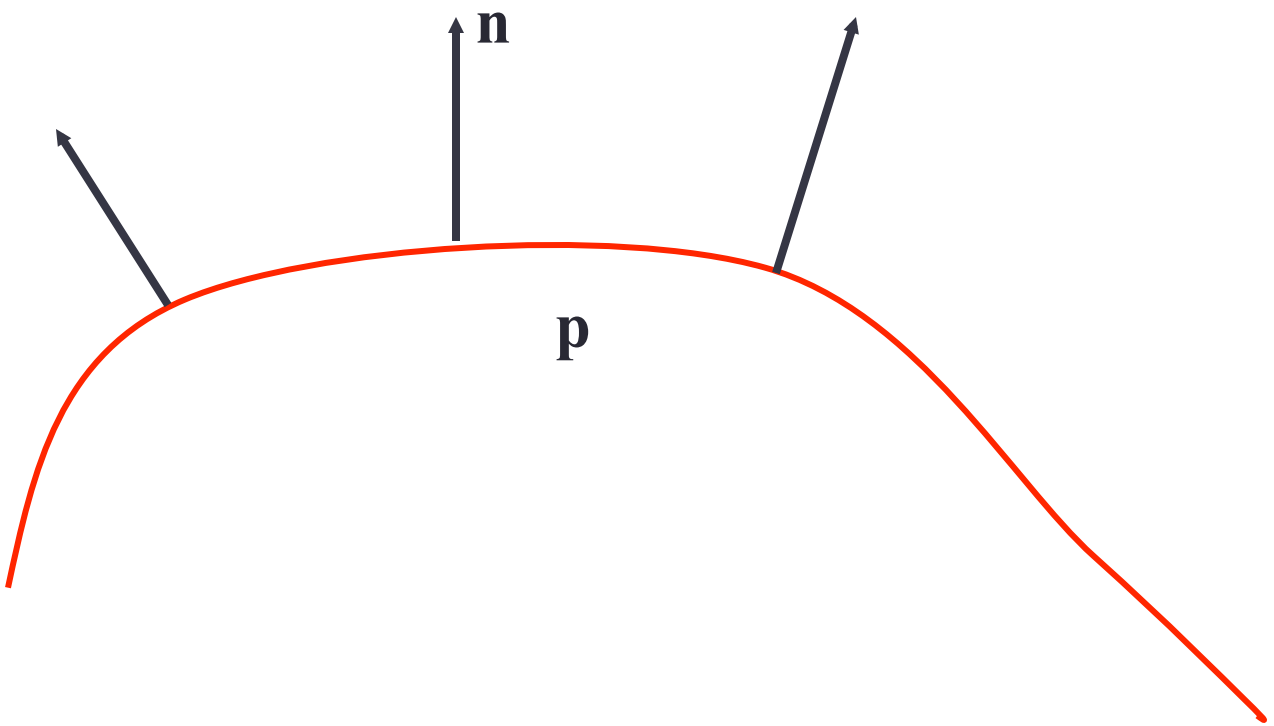
geometria



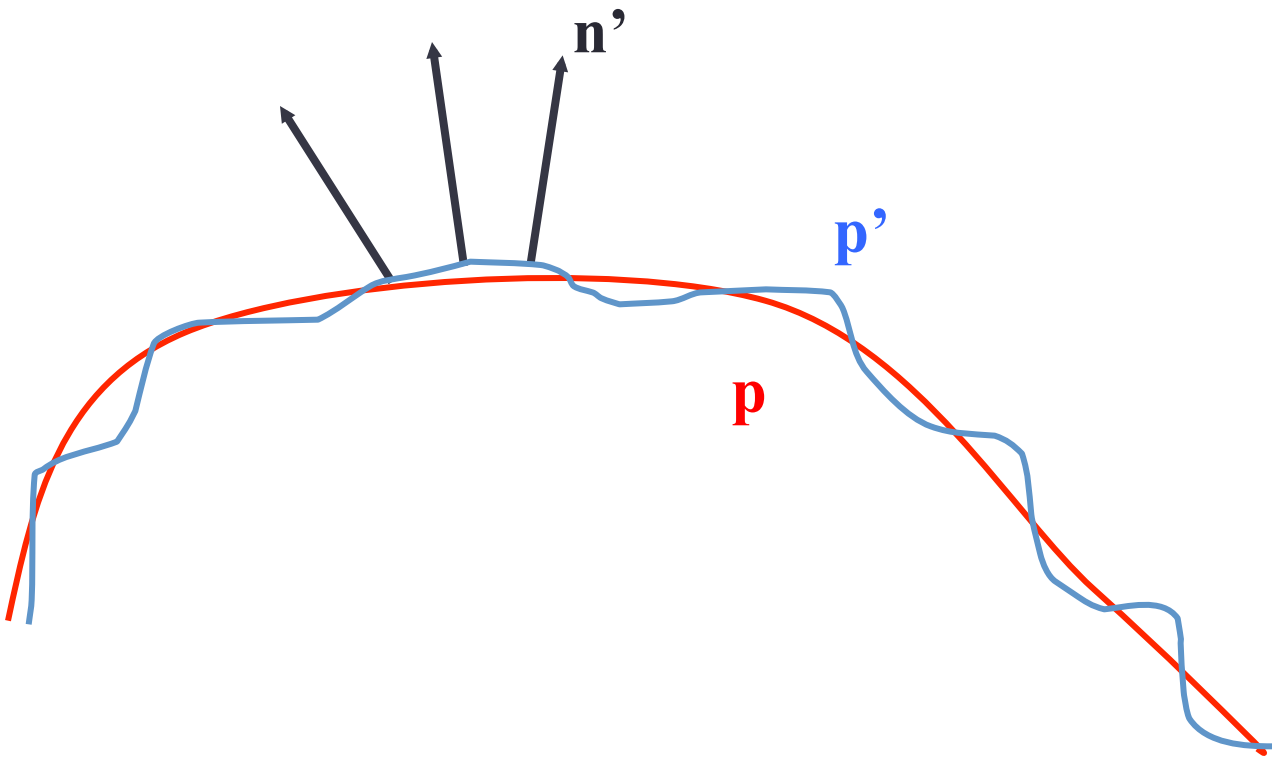
imagem

mapeamento

Superfície suave



Superfície rugosa



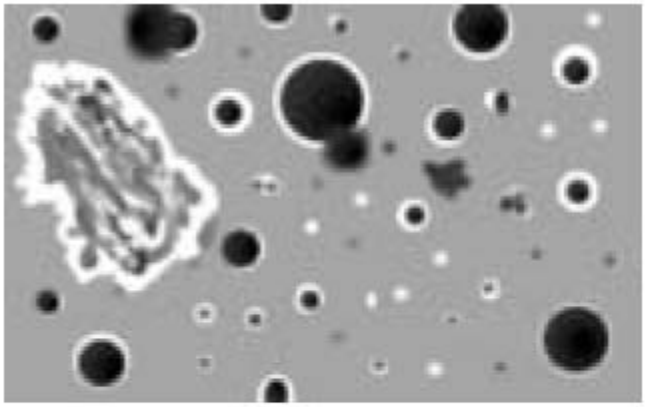
Mapeamento de rugosidade

- A perturbação do vetor é definido como:

$$Q(u,v) = P(u,v) + I(u,v) \cdot \mathbf{n}(u,v)$$

$$\mathbf{n}_1 = \frac{\partial Q}{\partial u} \times \frac{\partial Q}{\partial v}$$

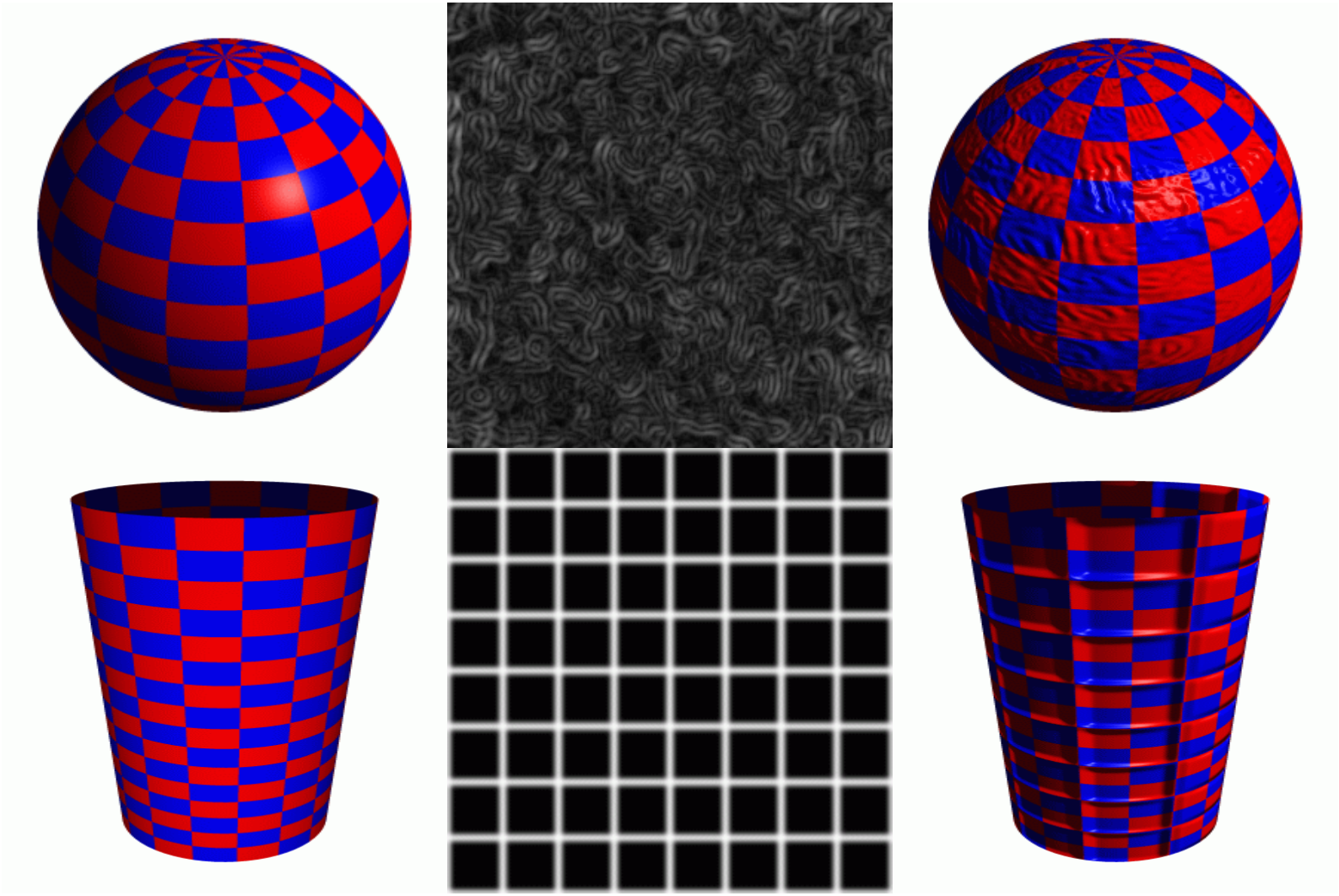
$$\mathbf{n}' = \mathbf{n}_1 / |\mathbf{n}_1|$$



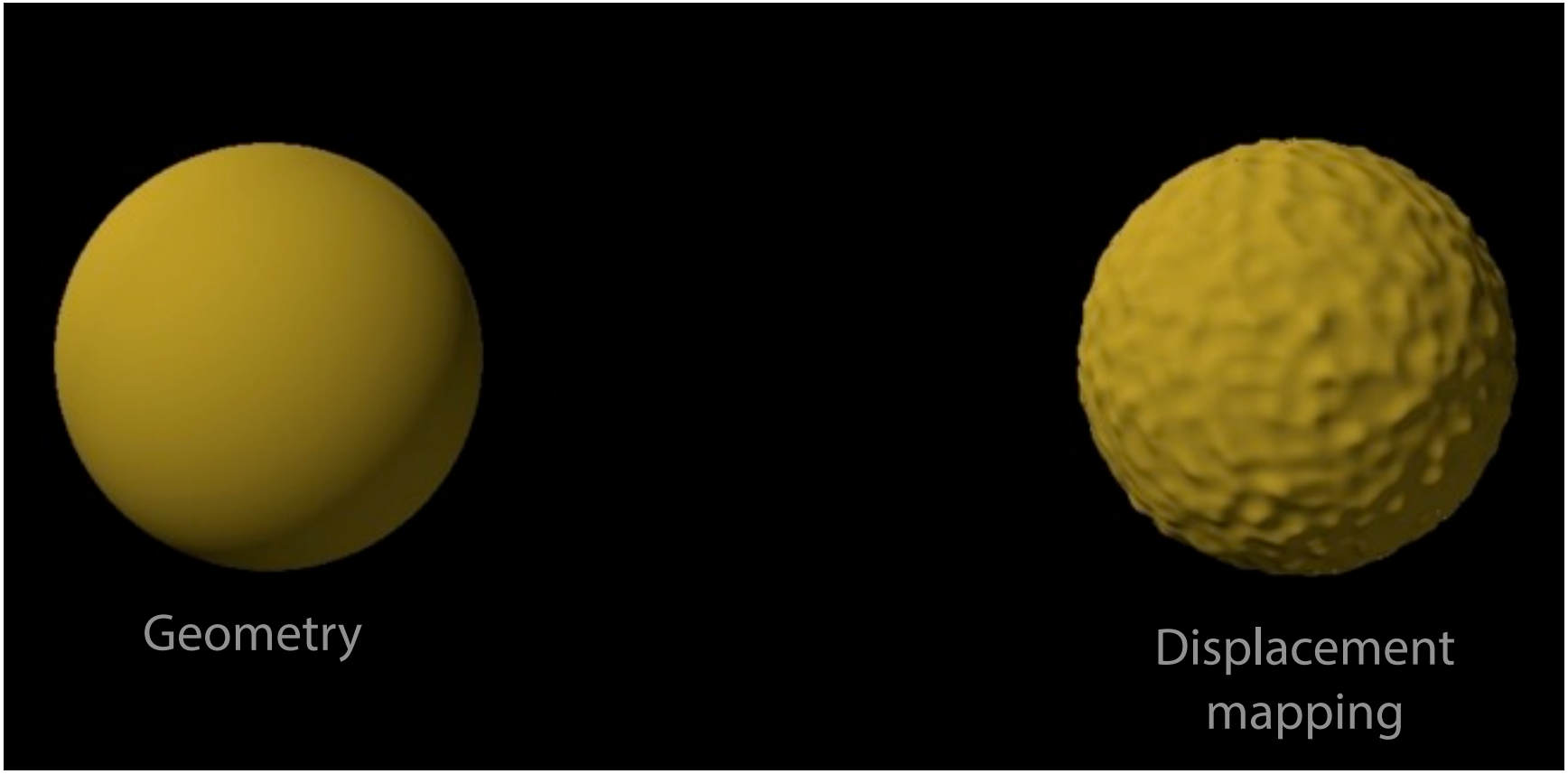
(a) Textura fonte.

(b) Mapeamento.

Exemplos



Displacement mapping



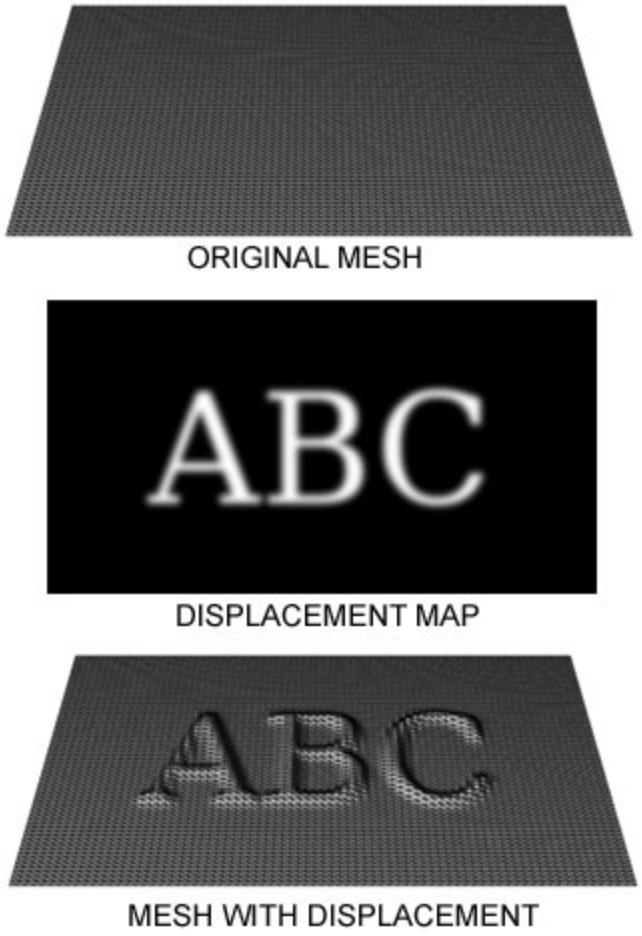
Mapeamento de deslocamento

- A imagem a ser mapeada é utilizada para fazer uma perturbação na geometria da superfície.

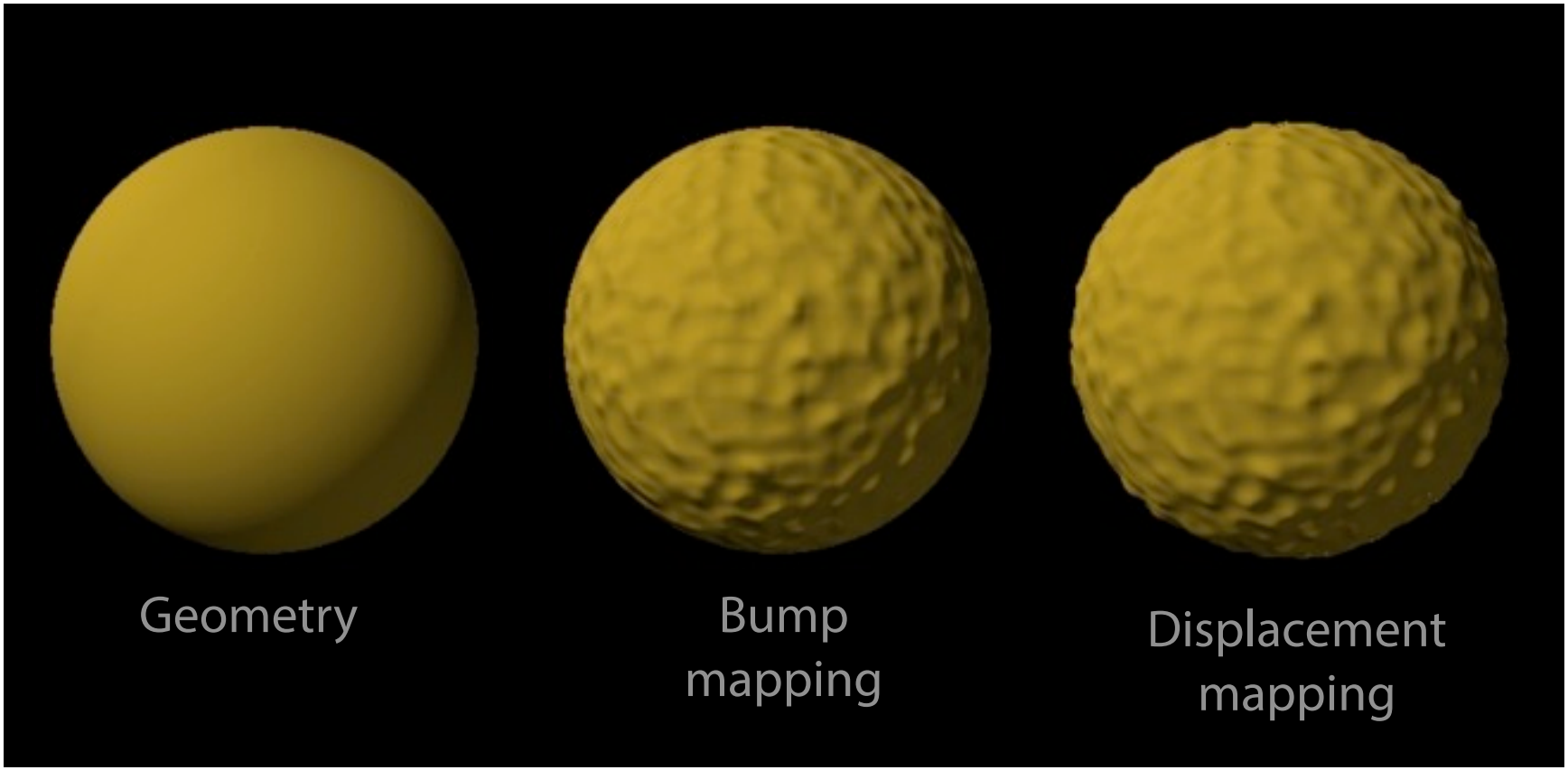
$$P(u,v) = P(u,v) + I(u,v) \cdot \mathbf{n}(u,v)$$

$$\mathbf{n}_1 = \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v}$$

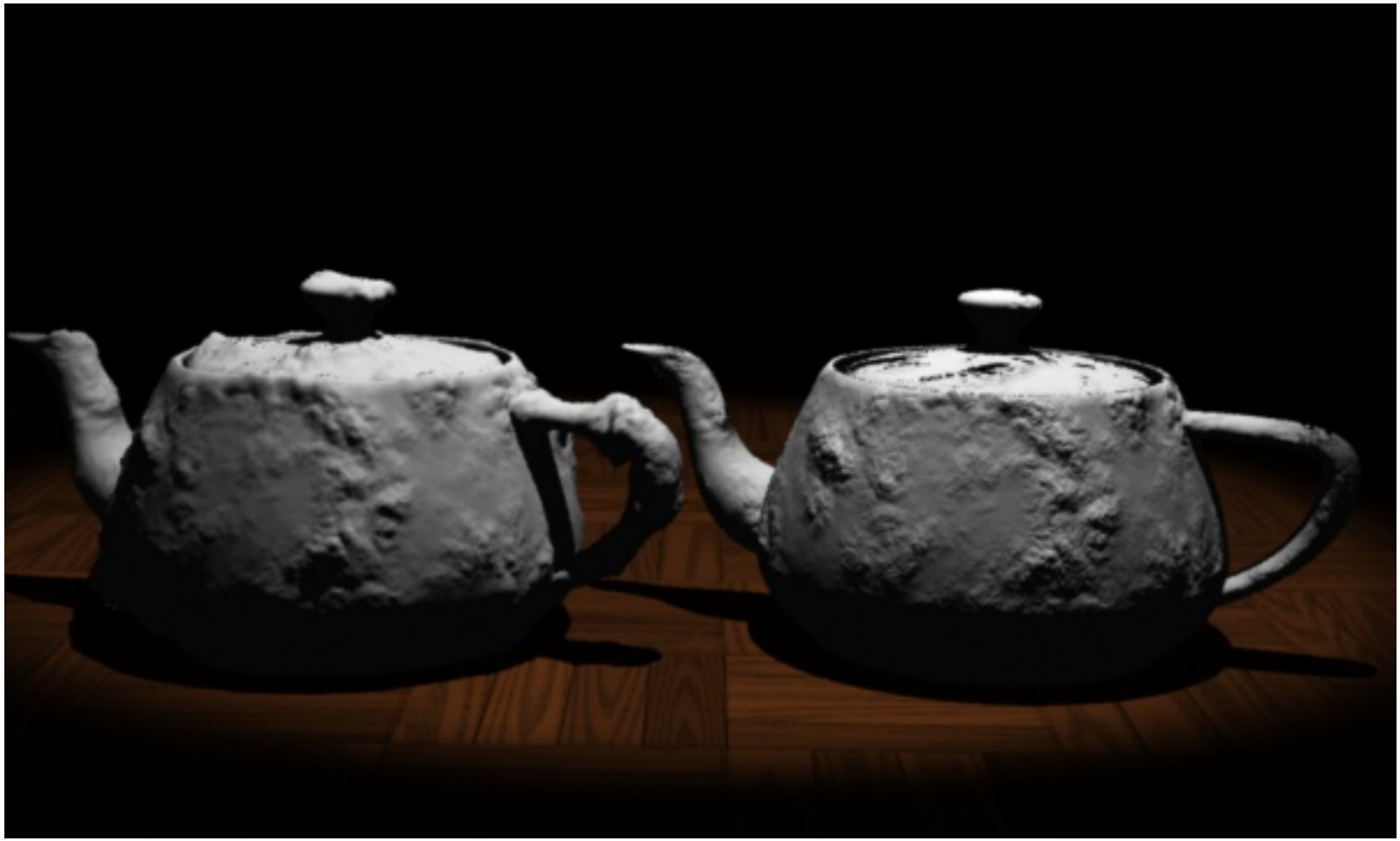
$$\mathbf{n}' = \mathbf{n}_1 / |\mathbf{n}_1|$$



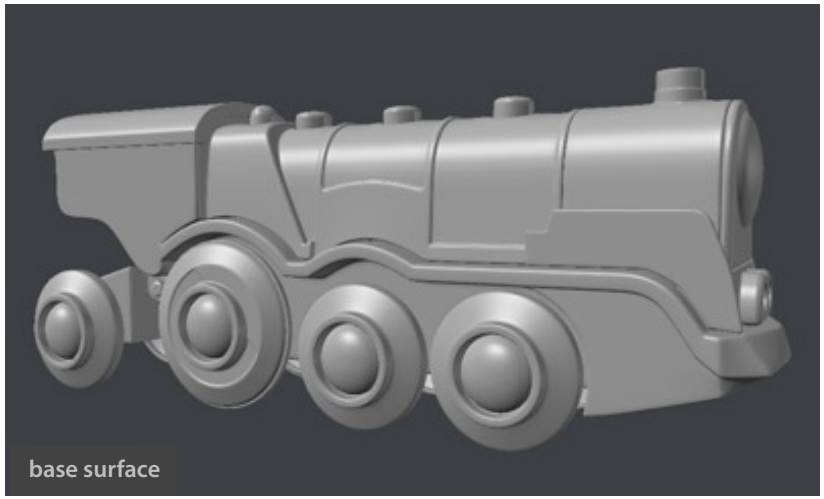
Displacement vs. *bump mapping*



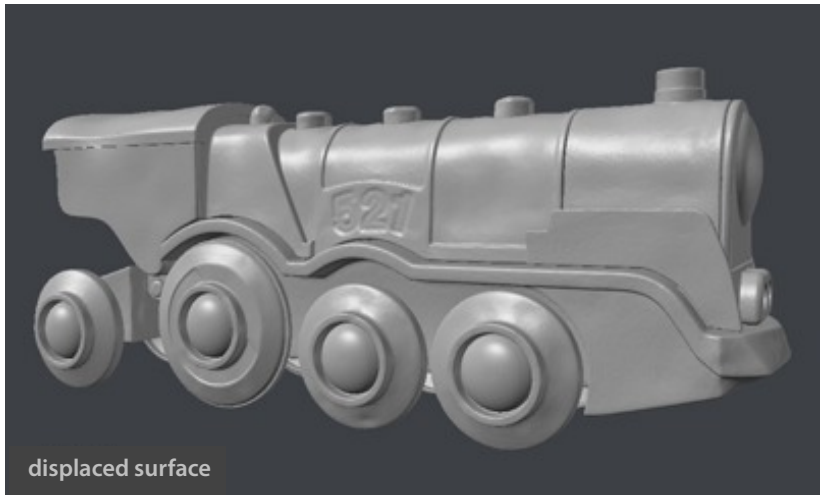
Comparação



Displacement mapping


base surface


hand-painted displacement map (detail)



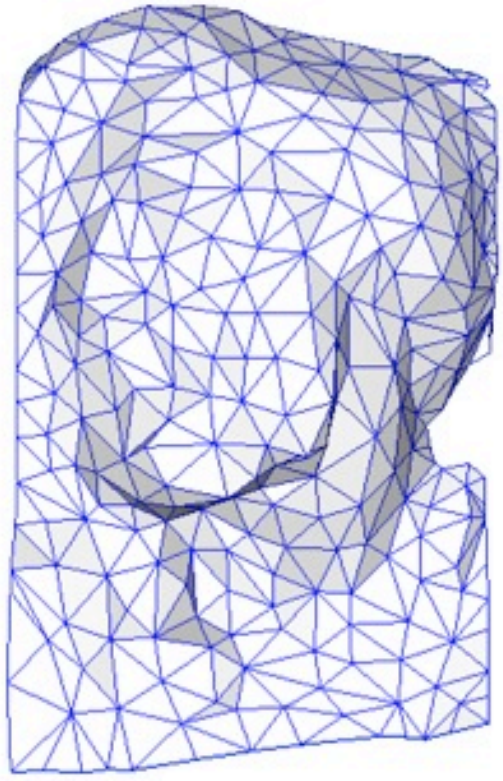
displaced surface

Paweł Filip
tolas.wordpress.com

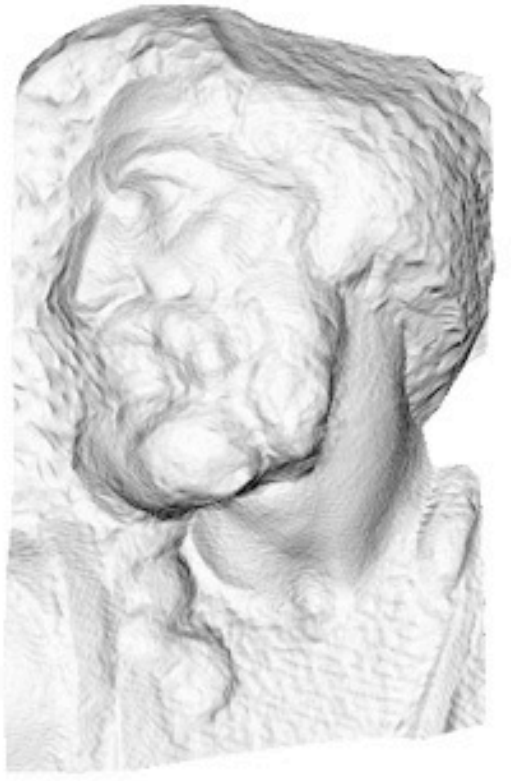

521
fryrender
physically-based render engine
©2007 Pawel Filip

Normal mapping

original mesh
4M triangles



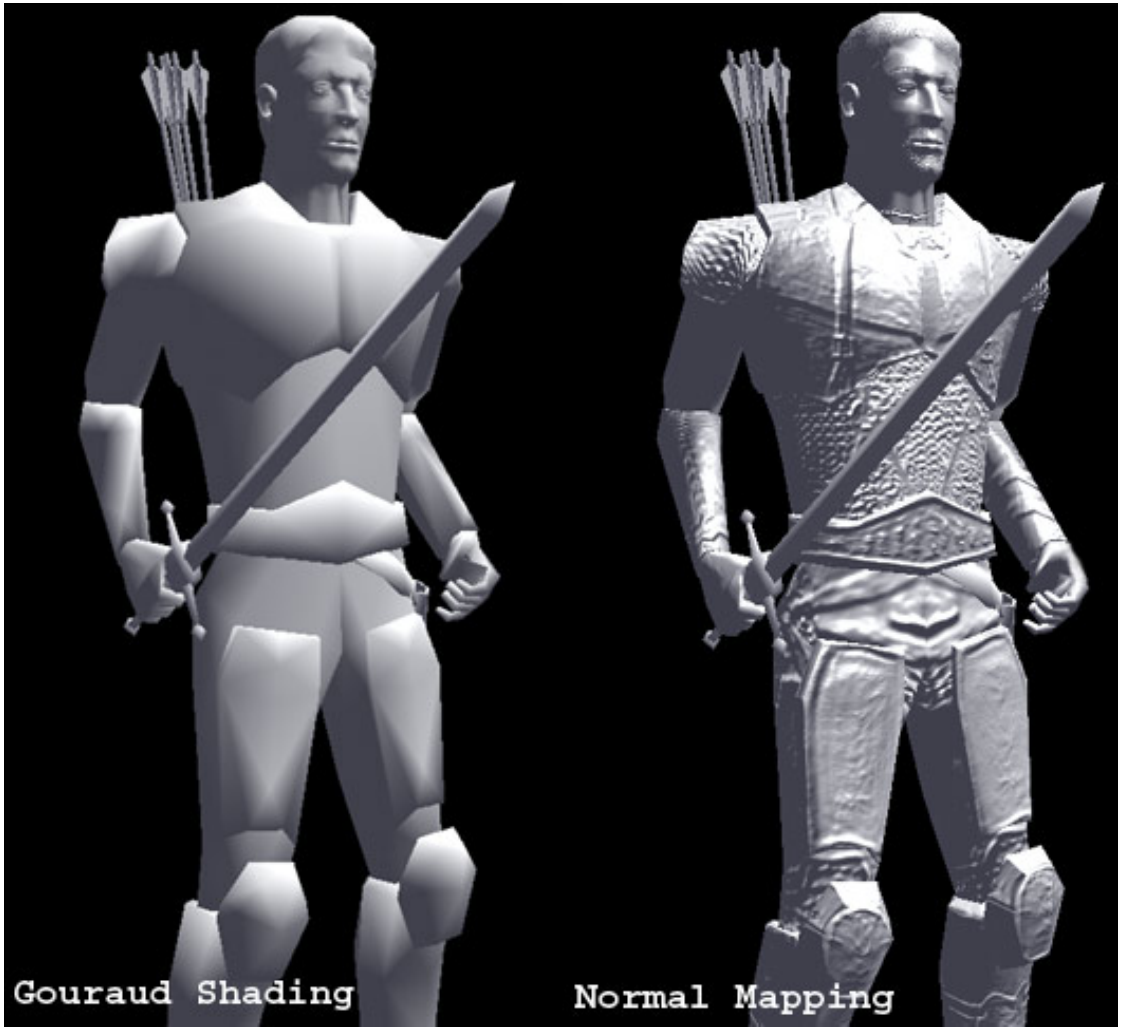
simplified mesh
500 triangles



simplified mesh
and normal mapping
500 triangles

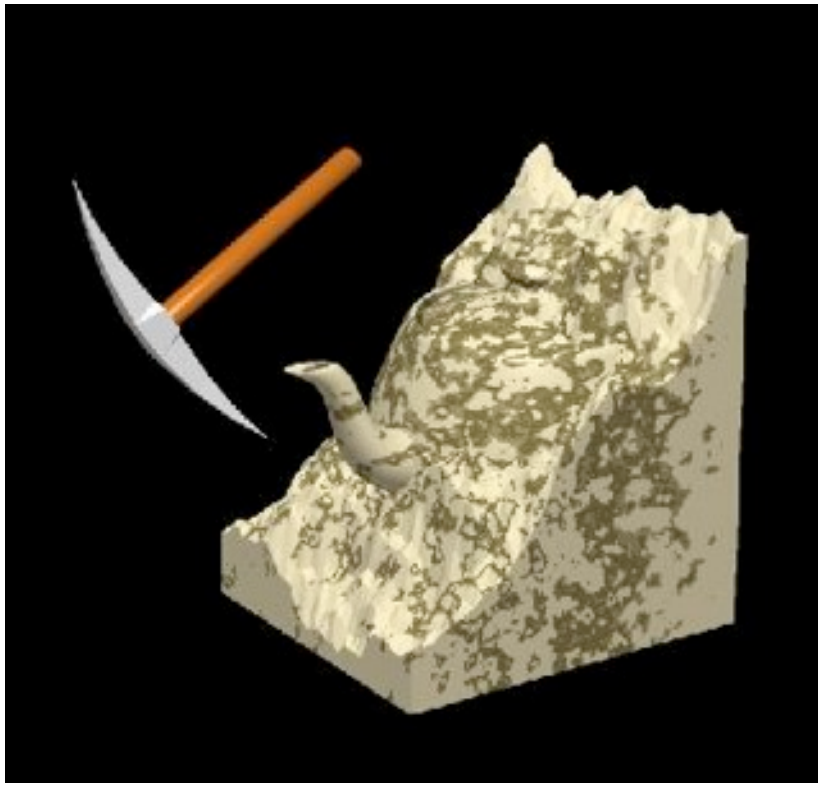
[Paolo Cignoni]

Normal mapping



Texturas tridimensionais

- Texture is a function of (u, v, w)
  - can just evaluate texture at 3D surface point
  - good for solid materials
  - often defined procedurally



[Wolfe / SG97 Slide set]

Texturas em OpenGL

- Três passos principais
  - Especificação da textura
    - Leitura ou geração da imagem
    - Atribuição de uma textura
    - Habilitação
  - Atribuição de coordenadas da textura aos vértices
    - Responsabilidade da aplicação
  - Especificar parâmetros da textura
    - Modo de repetição ("wrapping"), filtragem

Especificação da textura

- Definimos uma textura como um array de *texels* (*texture elements*) na memória principal
- Podemos usar uma imagem em um formato padrão, como JPEG
  - Imagem escaneada ou fotografada
  - Gerada pelo código da aplicação
- O WebGL dá suporte somente à mapas de textura 2D
  - Não é necessário habilitá-las (como no OpenGL desktop)
  - O OpenGL desktop disponibiliza mapas de texturas de 1-3D

Definindo imagem como textura

```
gl.TexImage2D( target, level, components,
    w, h, border, format, type, texels );
```

`target:` tipo da textura, e.g. `GL_TEXTURE_2D`

`level:` usado em mipmapping

`components:` elementos por texel

`w, h:` largura e comprimento dos `texels` em pixels

`border:` usado para suavização

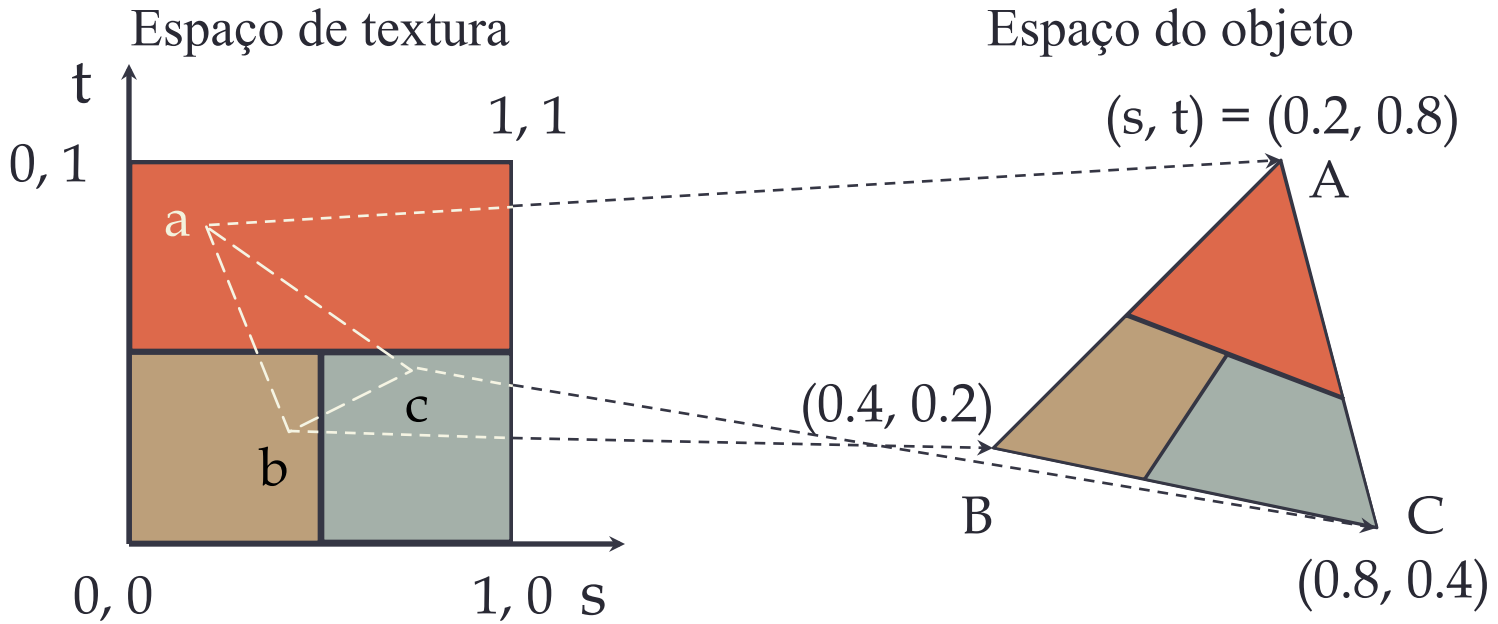
`format and type:` formato e tipo dos dados

`texels:` ponteiro para o buffer de *texels*

```
gl.TexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0, GL_RGB,
 GL_UNSIGNED_BYTE, texels);
```

Mapeamento de textura

- Baseada em coordenadas paramétricas
- Especificada como atributo em cada vértice



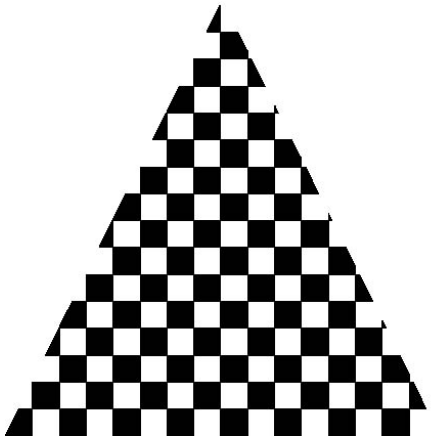
Parâmetros e faces de um cubo

```
var texCoord = [
    vec2(0, 0),
    vec2(0, 1),
    vec2(1, 1),
    vec2(1, 0)
];

function quad(a, b, c, d) {

     pointsArray.push(vertices[a]);
     colorsArray.push(vertexColors[a]);
     texCoordsArray.push(texCoord[0]);

     pointsArray.push(vertices[b]);
     colorsArray.push(vertexColors[a]);
     texCoordsArray.push(texCoord[1]);

  // etc
```

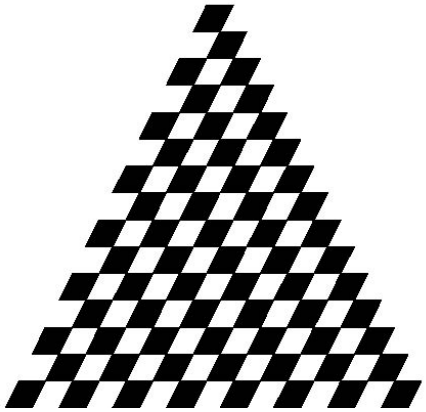
Interpolação

- O OpenGL utiliza interpolação para determinar os texels apropriados a partir de coordenadas de textura

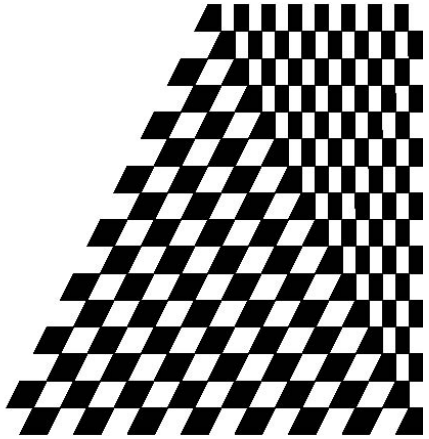
seleção adequada de coordenadas



má seleção de coordenadas

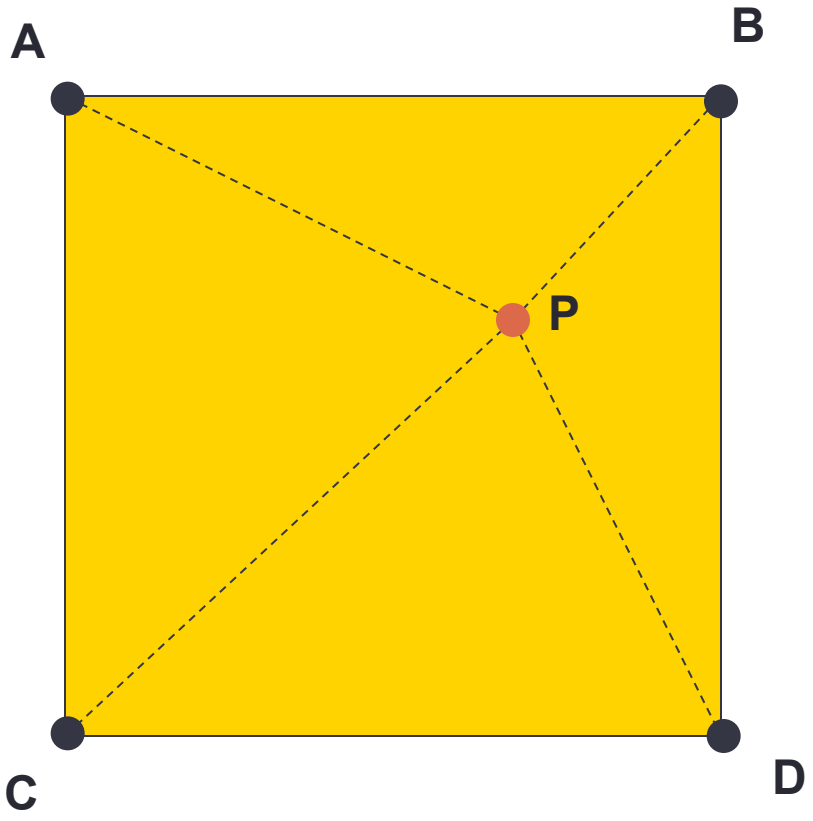


textura mapeada em trazepóide ilustrando efeitos da interpolação bilinear



Nearest neighbor

- O valor de P é determinado pelo ponto mais próximo;
- Medidas:
  - Distância Euclideana
  - Distância de Manhattan

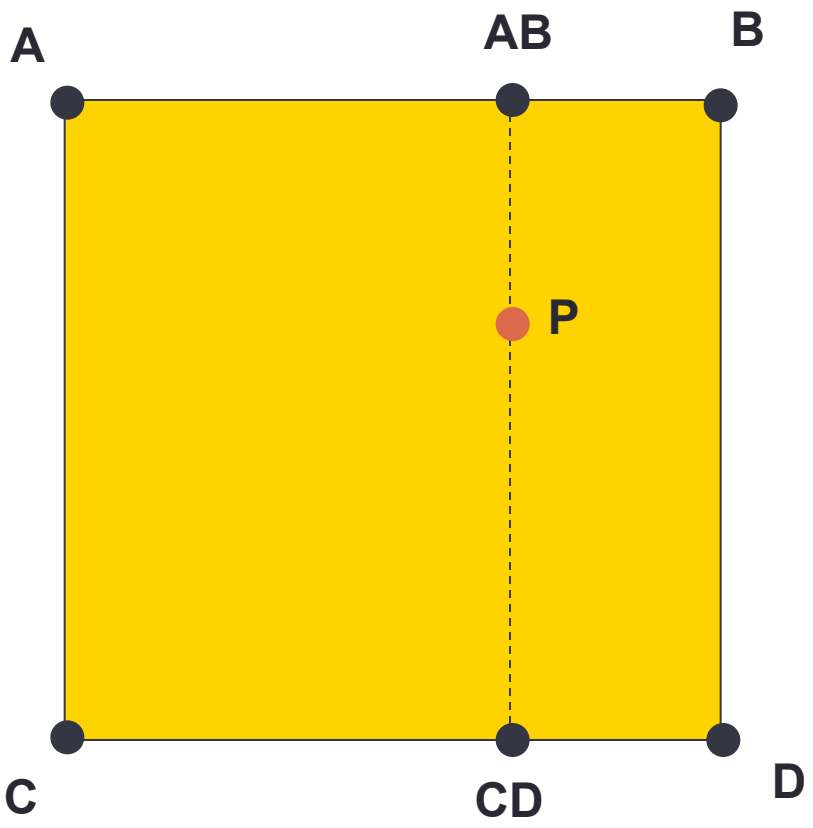


Nearest neighbor

- Atribua a intensidade no ponto (*X*,*Y*) como a intensidade do ponto mais próximo:
  - I(*X*,*Y*)=(round(*X*), round(*Y*)).
- Rápido,porém produz *aliasing* nas bordas.

Interpolação bilinear

- Utilizando interpolação linear, determina-se:
  - AB
  - CD
- Interpola-se o valor de P considerando AB e CD;



Interpolação bilinear

Original

Resampled

Convolução cúbica

- Na convolução cúbica, 16 pixels são usados para determinar a cor de um fragmento
- Calculada linha a linha, coluna a coluna

$$f(X) = I(u-1)f_{-1} + I(u)f_0 + I(u+1)f_1 + I(u+2)f_2,$$

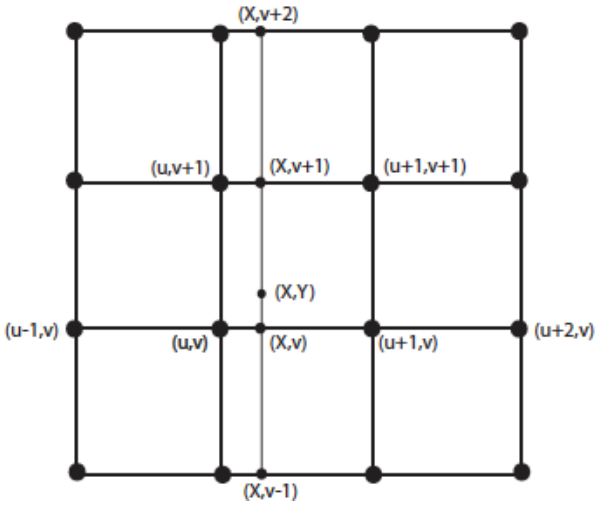
$$f_{-1} = -\frac{1}{2}t^3 + t^2 - \frac{1}{2}t,$$

$$f_0 = \frac{3}{2}t^3 - \frac{5}{2}t^2 + 1,$$

$$f_1 = -\frac{3}{2}t^3 + 2t^2 + \frac{1}{2}t,$$

$$f_2 = \frac{1}{2}t^3 - \frac{1}{2}t^2,$$

$$t = X - u.$$



Convolução cúbica

Original

Resampled

Comparação

- Vamos supor que uma textura seja rotacionada 36 vezes com um incremento angular de 10º e então subtraída da imagem original.

nearest-neighbor

bilinear

cubic convolution

Utilizando objetos de textura

1. Especificação de objetos de textura
2. Configurar modo de filtragem
3. Configurar função de mapeamento
4. Configurar modo de repetição (“wrapping”)
5. Associar objeto de textura
6. Especificar coordenadas de textura

Parâmetros de configuração

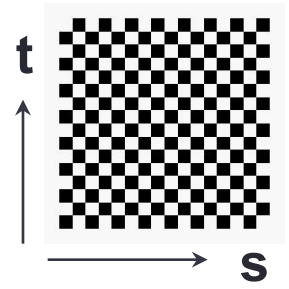
- O OpenGL/WebGL possuem uma variedade de parâmetros de configuração para determinar como a textura será aplicada:
  - Parâmetros de *wrapping* indicarão o que acontecerá se *s* e *t* estiverem fora do intervalo [0,1]
  - Modos de filtragem nos permitirão escolher entre a amostragem por área ou pontual
  - *Mipmapping* nos permitirá associar texturas para diferentes resoluções

Modo de repetição (*wrapping*)

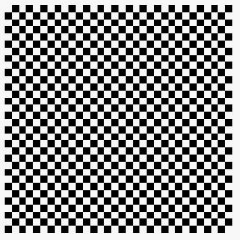
Clamping: if s, t > 1 use 1, if s, t < 0 use 0

Wrapping: use s, t modulo 1

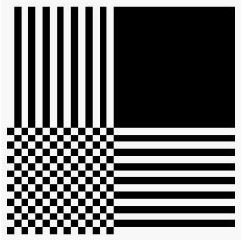
```
gl.texParameteri(gl.TEXTURE_2D,
     gl.TEXTURE_WRAP_S, gl.CLAMP )
gl.texParameteri(gl.TEXTURE_2D,
     gl.TEXTURE_WRAP_T, gl.REPEAT )
```



texture



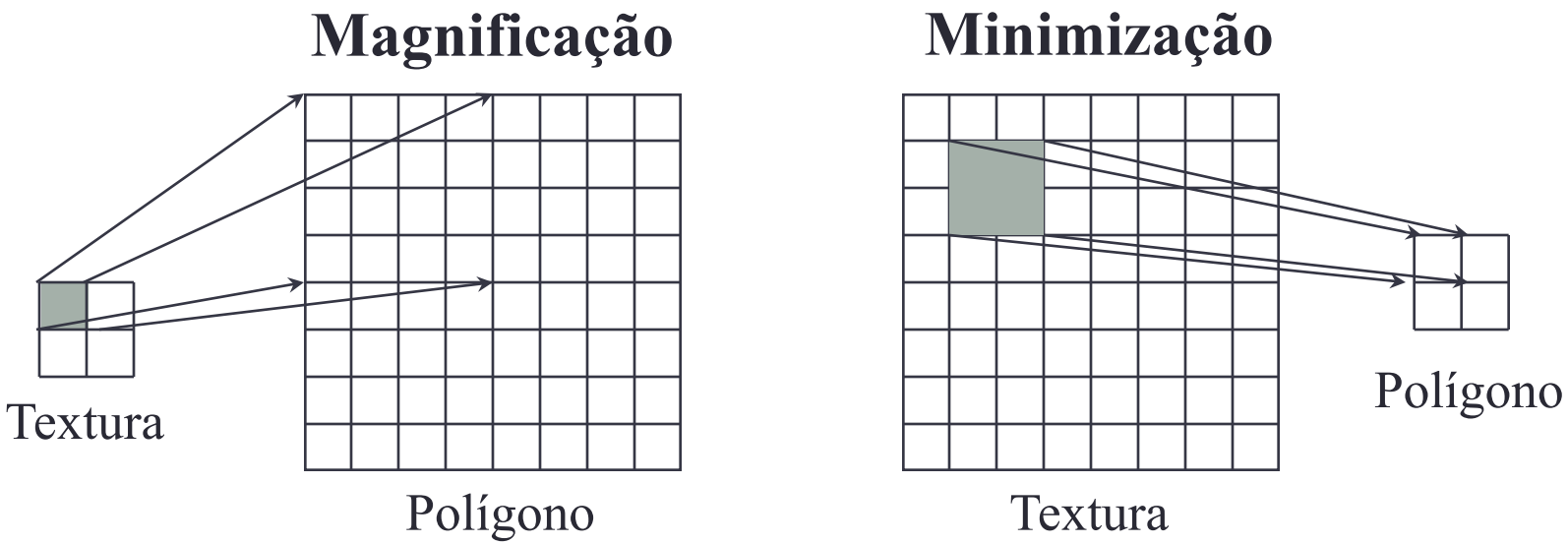
gl.REPEAT wrapping



gl.CLAMP wrapping

Magnificação e minimização

Can use point sampling (nearest texel) or linear filtering (2 x 2 filter) to obtain texture values



Filtragem

Modos determinados através da chamada:

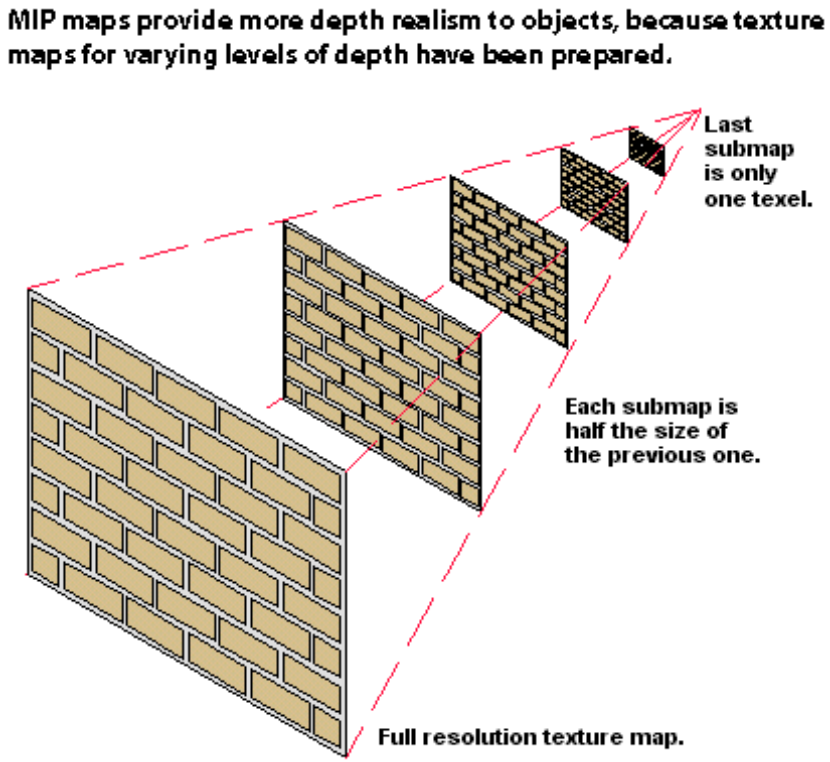
```
gl.TexParameteri( target, type, mode )
```

```
gl.TexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
                 GL_NEAREST);

gl.TexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
                 GL_LINEAR);
```

O uso da filtragem linear requer uma borda de um *texel* extra para filtragem nas arestas

MIP* mapping

- 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

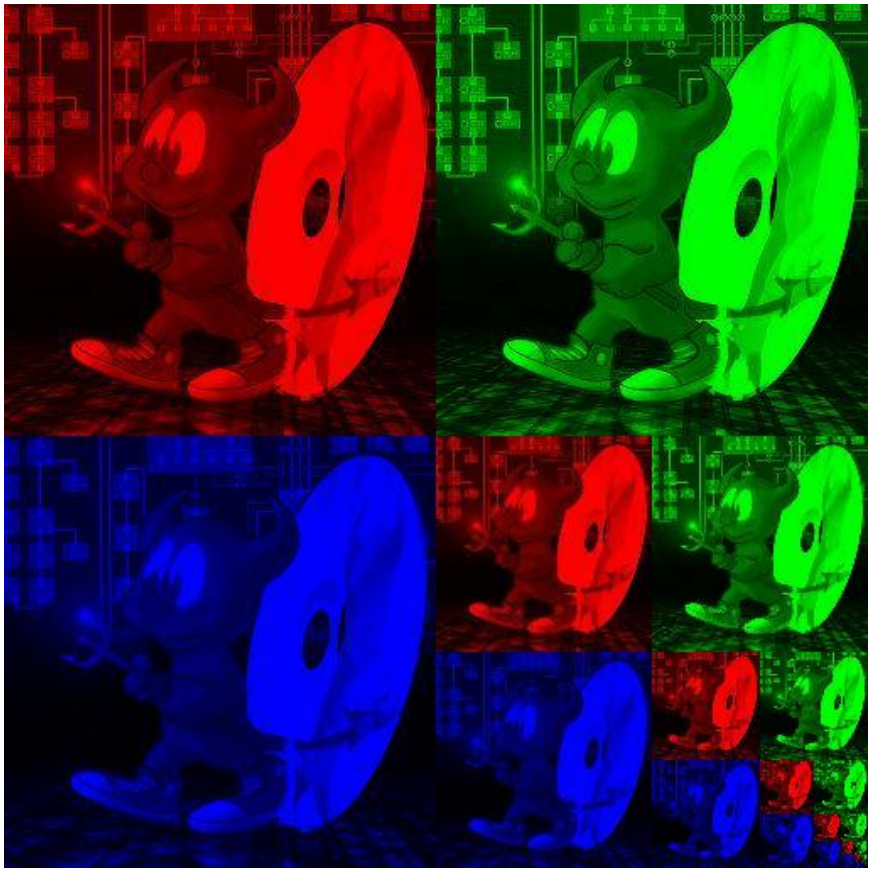



```
glTexImage*D( GL_TEXTURE_*D, level, … )
```

*multum in parvo, means multitude in a small space

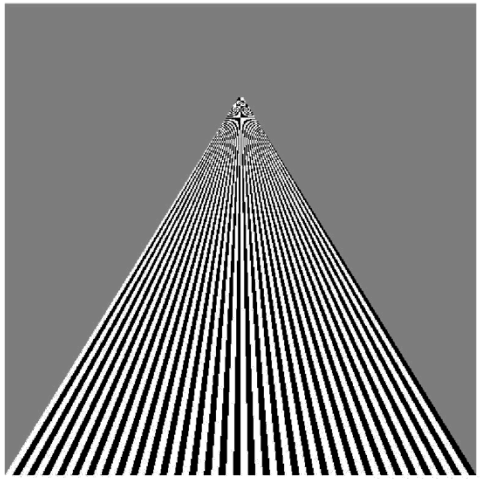
Mapeamento MIP

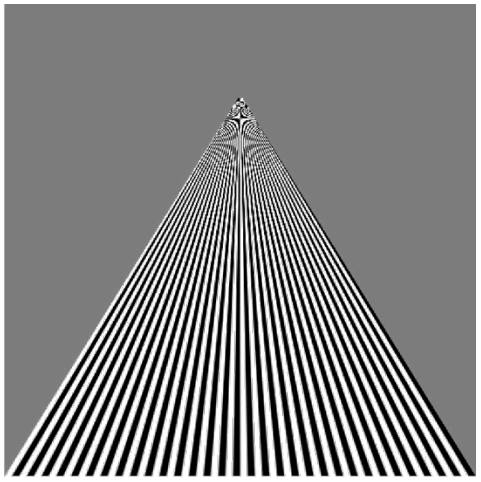
Mapeamento MIP



Exemplo: textureSquare.html/.js

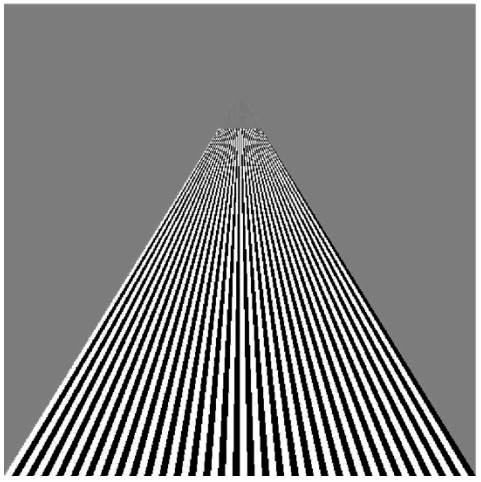
filtragem nearest-neighbor

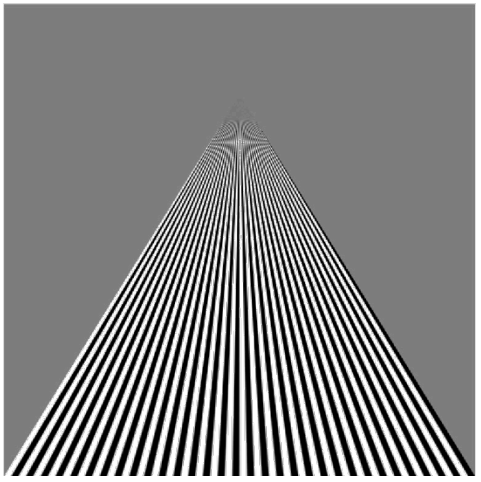




filtragem linear

filtragem nearest-neighbor mipmap





filtragem linear mipmap

Aplicando texturas

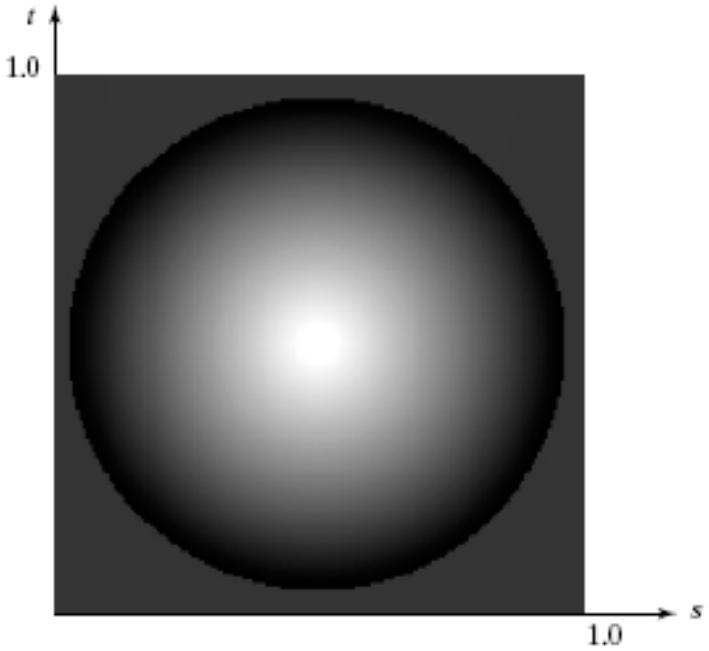
- Texturas podem ser aplicadas de diferentes formas
  - A textura pode determinar a cor do fragmento
  - Modulada com uma cor predeterminada
  - Fusão com uma cor de ambiente
- A função do pipeline antigo `glTexEnv()` que selecionava o modo de modulação foi deprecado
  - Agora devemos utilizar *fragment shader*
  - Podemos também utilizar múltiplas unidades de textura

Texturas procedurais

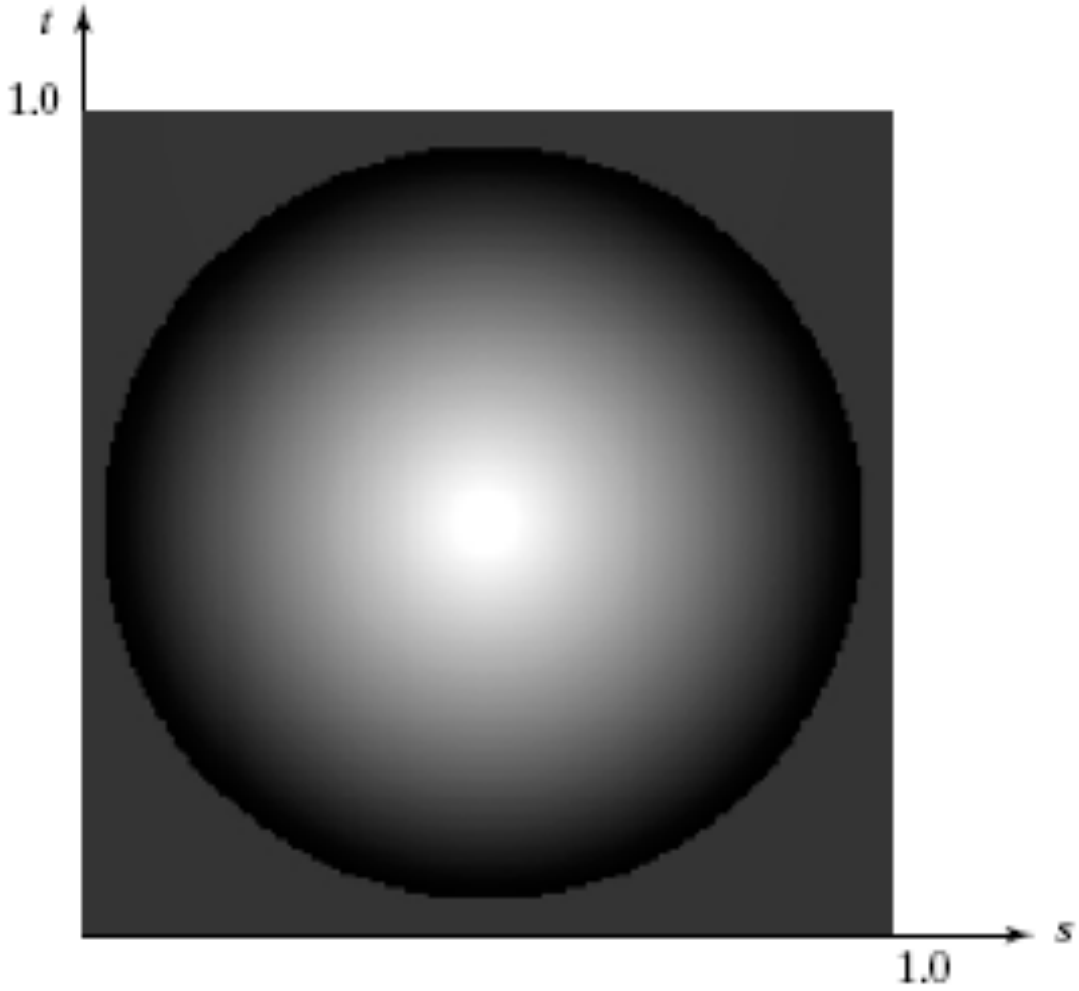
- Função para desenho de uma esfera:

```
float Esfera (float s, float t) {
    float r = sqrt((s-0.5)*(s-0.5) + (t-0.5)*(t-0.5));
    if (r <= 0.3) return 1 – r / 0.3;
          else return 0.2;
 } // 0.2 é intensidade do fundo
```

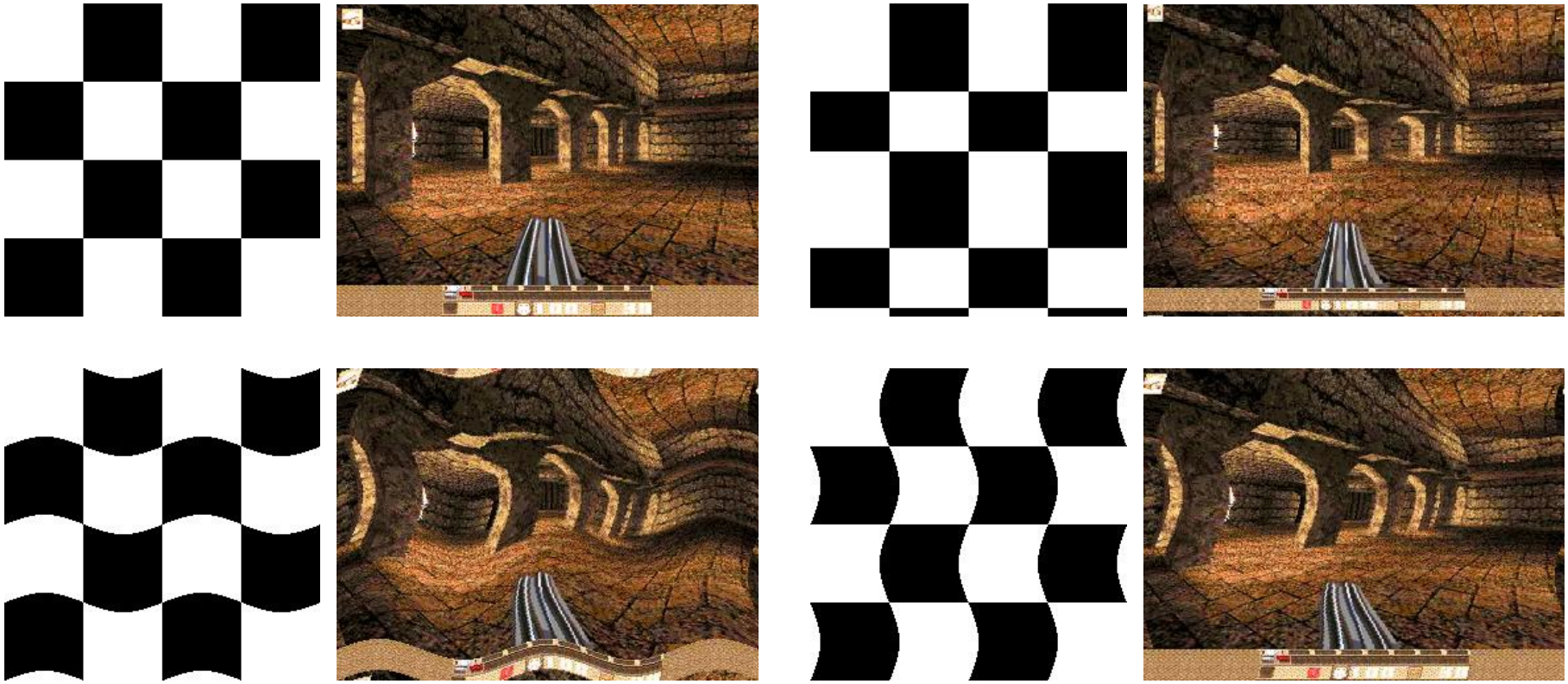
- Podemos criar qualquer função para que, dados os valores de *s* e *t*, produza um valor de textura.



Texturas procedurais



Introdução de distorções



Distorções

Aplicando texturas

- Texturas são aplicadas na execução do fragment shader, utilizando um *sampler*
- *Samplers* retornam a cor da textura a partir de um *texture object*

```
varying vec4 color;    // color from rasterizer
varying vec2 texCoord; // texture coordinate from rasterizer
uniform sampler2D texture; // texture object from application

void main()  {
    gl_FragColor = color * texture2D( texture, texCoord );
}
```

Vertex shader

- Normalmente, o vertex shader deverá propagar as coordenadas de textura
- Porém, precisa cuidar das outras tarefas usuais
  - Calcular a posição do vértice
  - Calcular a cor, caso necessário

```
attribute vec4 vPosition; //vertex position in object coordinates
attribute vec4 vColor;  //vertex color from application
attribute vec2 vTexCoord; //texture coordinate from application

varying vec4 color; //output color to be interpolated
varying vec2 texCoord; //output tex coordinate to be interpolated
```

Parâmetros e faces de um cubo

```
var texCoord = [
    vec2(0, 0),
    vec2(0, 1),
    vec2(1, 1),
    vec2(1, 0)
];

function quad(a, b, c, d) {

     pointsArray.push(vertices[a]);
     colorsArray.push(vertexColors[a]);
     texCoordsArray.push(texCoord[0]);

     pointsArray.push(vertices[b]);
     colorsArray.push(vertexColors[a]);
     texCoordsArray.push(texCoord[1]);

  // etc
```

Texture object

```
function configureTexture( image ) {
    var texture = gl.createTexture();
    gl.bindTexture( gl.TEXTURE_2D, texture );
    gl.pixelStorei(gl.UNPACK_FLIP_Y_WEBGL, true);
    gl.texImage2D( gl.TEXTURE_2D, 0, gl.RGB,
                  gl.RGB, gl.UNSIGNED_BYTE, image );
    gl.generateMipmap( gl.TEXTURE_2D );
    gl.texParameteri( gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER,
                      gl.NEAREST_MIPMAP_LINEAR );
    gl.texParameteri( gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER,
                      gl.NEAREST );
    gl.activeTexture(gl.TEXTURE0);
    gl.uniform1i(gl.getUniformLocation(program, "texture"), 0);
}
```

Ligando com os shaders

```
var vTexCoord = gl.getAttribLocation( program, "vTexCoord" );

gl.enableVertexAttribArray( vTexCoord );
gl.vertexAttribPointer( vTexCoord, 2, gl.FLOAT, false, 0, 0);

    // Set the value of the fragment shader texture sampler variable
    //   ("texture") to the the appropriate texture unit. In this case,
    //   zero for GL_TEXTURE0 which was previously set by calling
    //   gl.activeTexture().

    gl.uniform1i( glGetUniformLocation(program, "texture"), 0 );
```

Tarefa de casa

- Leitura livro-texto
  - Shirley and Marschner. Fundamentals of Computer Graphics, CRC Press, 3rd Ed. 2010
  - Capítulo 11