Computação Gráfica I

Professor:

Anselmo Montenegro
www.ic.uff.br/~anselmo

Conteúdo:

- Mapeamentos

Mapeamentos : *introdução*

- Os modelos de tonalização apresentados anteriormente consideram que as superfícies possuem reflectância uniforme.
- Isto funciona bem para materias como paredes pintadas, papel, mas não é suficiente para representar objetos tais como quadros, madeira, etc.
- Uma possível solução consiste em armazenar a reflectância da superfície como uma função.

Mapeamentos : *introdução*

- Uma solução mais utilizada, entretanto, consiste em utilizar uma imagem e mapeá-la sobre a superfície.
- A função ou imagem é denominada mapa de textura e o processo é conhecido como mapeamento de textura.
- A idéia de mapeamento de textura foi proposta em 1975 por Catmull.

Mapeamentos : *introdução*

- 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.

<u>Mapeamentos</u> : *mapeamentos de objetos gráficos*

- Dados dois objetos $O_1=(U,f)$ e $O_2=(V,g)$, um mapeamento de O_1 em O_2 é uma transformação $T:V\rightarrow U$.

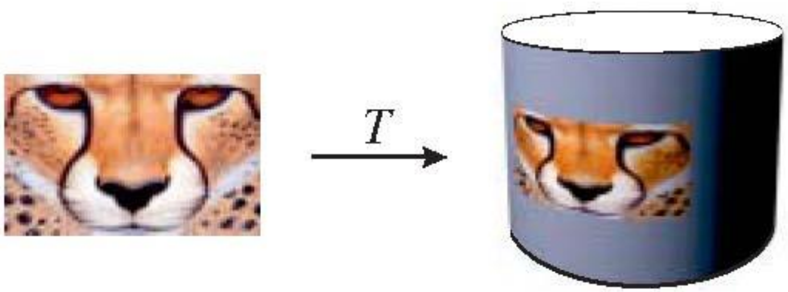


Figura 1. Mapeamento de textura.

$$\begin{tikzcd}
\mathbf{V} \arrow[d, "\mathbf{T}"'] \arrow[dr, "\mathbf{g}"] & \\
\mathbf{U} \arrow[r, "\mathbf{f}"'] \arrow[ur, phantom, "\mathbf{g^*}"] & \mathbf{R^n}
\end{tikzcd}$$

Mapeamentos : *mapeamentos de objetos gráficos*

- O mapeamento ocorre como uma mudança de coordenadas
- Queremos levar as coordenadas de textura de U para V
- Logo, a transformação é de V para U

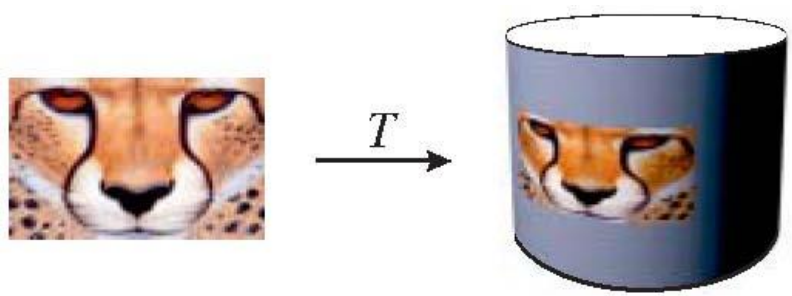


Figura 1. Mapeamento de textura.

$$\begin{tikzcd} \mathbf{V} \arrow[d, "\mathbf{T}"'] \arrow[dr, "\mathbf{g}"] \arrow[dr, phantom, "\mathbf{g}^*"'] & \\ \mathbf{U} \arrow[r, "\mathbf{f}"] & \mathbf{R}^n \end{tikzcd}$$

<u>Mapeamentos</u> : *mapeamentos de objetos gráficos*

- O mapeamento ocorre como uma mudança de coordenadas
- Queremos levar as coordenadas de textura de U para V
- Logo, a transformação é de V para U

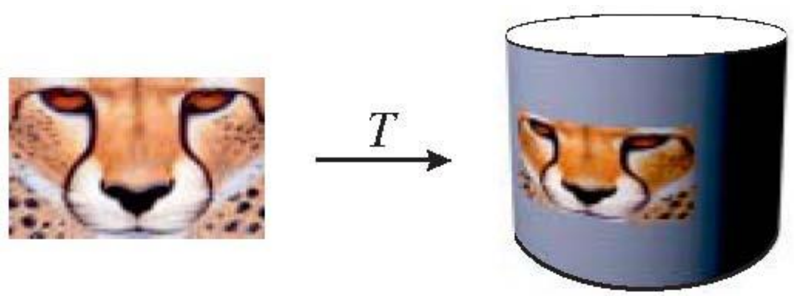


Figura 1. Mapeamento de textura.

$$\begin{tikzcd} V \arrow[d, "T"'] \arrow[dr, "g"] & \\ U \arrow[r, "f"'] \arrow[ur, phantom, "g^*"] & R^n \end{tikzcd}$$

<u>Mapeamentos</u> : *mapeamentos de objetos gráficos*

- O objeto $O_1=(U,f)$ é denominado ***objeto fonte*** e o objeto $O_2=(V,g)$ é denominado ***objeto alvo***

- O mapeamento T define uma nova função de atributos g* no objeto mapeado O_2, onde $g^*=f \circ T$

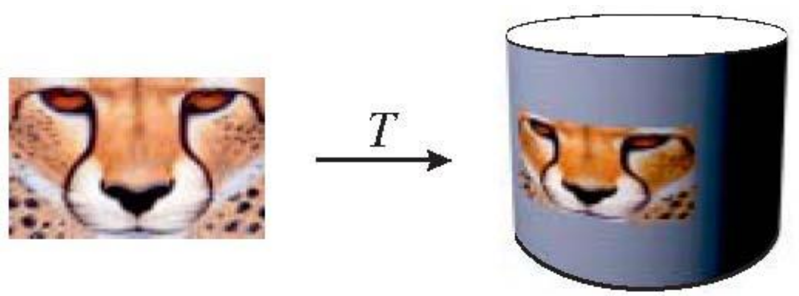


Figura 1. Mapeamento de textura.

$$\begin{tikzcd} V \arrow[d, "T"'] \arrow[dr, "g"] \arrow[d, phantom, shift left=6ex, "g^*"] & \\ U \arrow[r, "f"] & R^n \end{tikzcd}$$

Mapeamentos : *mapeamentos de objetos gráficos*

- O novo atributo do objeto O_2 em um ponto $p \in V$ é o atributo do objeto O_1 no ponto $T(p)$ ou seja $f(T(p))$.

- Em geral, exigimos que T seja bijetiva, já que isto faz com que pontos distintos de V não sejam mapeados no mesmo atributo.

- Isto também é útil no cálculo do mapeamento.

Mapeamentos : *mapeamentos de objetos gráficos*

- A função de atributos g* pode ser combinada com outras funções de atributo do objeto O_2.

- Essa variedade de combinação, juntamente com as diversas formas de escolher o mapeamento T dão origem a uma grande variedade de aplicações.

Mapeamentos : *dimensão de mapeamentos*

- A dimensão do mapeamento é igual a dimensão do objeto gráfico fonte O_1.
- Portanto é igual a dimensão do suporte geométrico U de O_1.
- Casos mais comuns:
  - Mapeamentos unidimensionais – usados para mudar atributos de uma curva
  - Mapeamento bidimensional – onde O_1 é uma imagem
  - Mapeamento tridimensional – onde O_1 é uma imagem 3D

Mapeamentos : *dimensão de mapeamentos*

- É também comum utilizar mapeamentos *nD*, onde a dimensão *n-1* é usada para definir a textura a ser mapeada e a dimensão extra é usada como tempo para mudar os atributos da animação.

- Bob espoja aqui

<u>Mapeamentos</u> : *mapeamento bidimensional*

- Mapeamento em que o suporte geométrico do objeto fonte é 2D.
- Caso típico: mapeamentos de textura bidimensionais
- Exemplo: O_2 é um cilindro é O_1 uma imagem

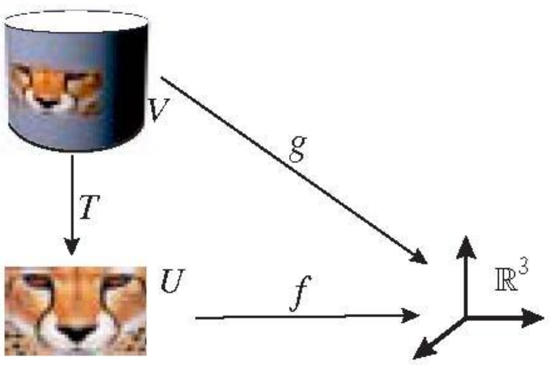


Figura 2. Mapeamento 2*D*.

Mapeamentos : *mapeamento bidimensional*

- Na figura abaixo, a componente de reflexão difusa do pedaço da superfície texturizada foi obtida da função de atributos $g^{*}=f \circ T$
- Esse novo atributo tem o efeito visual de transferir a imagem para superfície, como em um processo de decalque

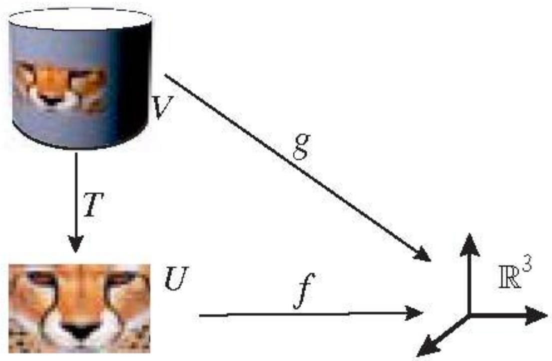


Figura 2. Mapeamento 2D.

Mapeamentos : *mapeamento tridimensional*

- A idéia do mapeamento 3D é similar do mapeamento 2D:

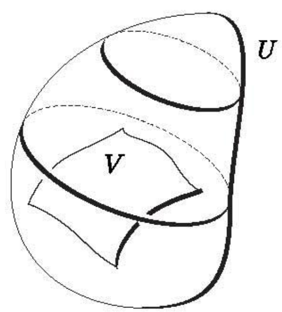
  encontrar uma função definida no suporte geométrico do objeto alvo deforma a determinar uma nova função de atributos desse objeto

- Um caso simples é aquele em que:
  - Objeto alvo - $O_2(V,g)$, $V \subset R_3$
  - Objeto fonte - $O_1(U,f)$, $U \subset R_3$, $f:U \rightarrow R_3$
  - $V \subset U$

Mapeamentos : *mapeamento tridimensional*

- Nesse caso a função de mapeamento é trivial: é a transformação de inclusão $T:V \rightarrow U$, $T(p)=p$

- A função de atributos g^* é simplesmente a função f do objeto fonte restrita a V, *isto é,* $g^*:f\ |V \rightarrow R^3$



(a) Mapeamento 3D

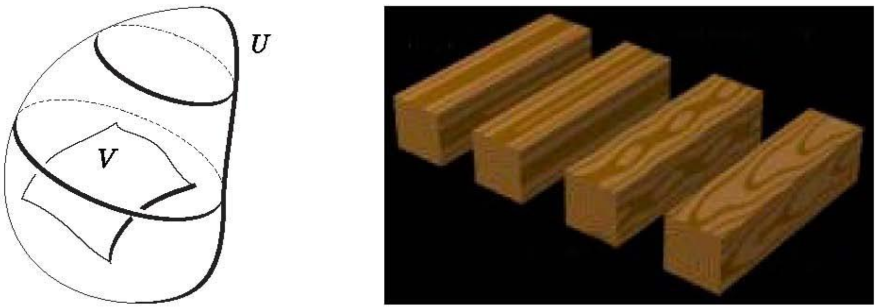
(b) Textura de madeira 3D

Figura 3.

<u>Mapeamentos</u> : *mapeamento tridimensional*

- No mapeamento 3D a imagem volumétrica pode ser interpretada como um função que mede a densidade do espaço ambiente

- Como no mapeamento 2D utilizamos a nova função de atributo para alterar a componente de reflexão difusa do objeto mapeado



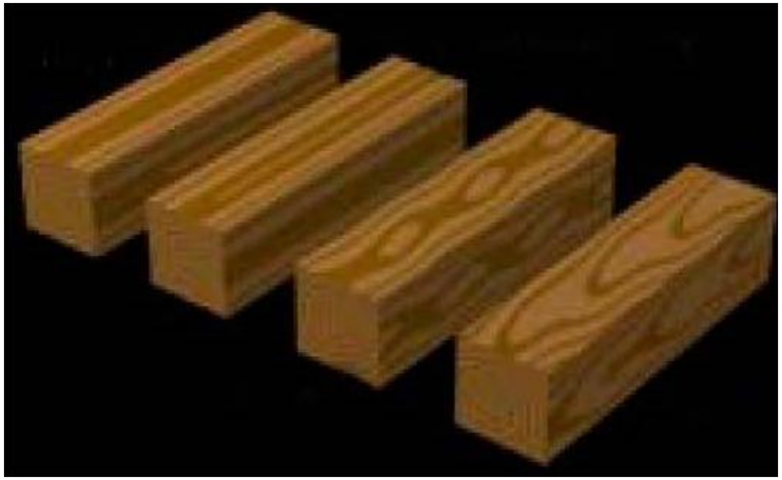
(a) Mapeamento $3D$ (b) Textura de madeira $3D$

Figura 3.

<u>Mapeamentos</u> : *mapeamento tridimensional*

- Na figura abaixo, foram gerados diferentes padrões de textura de madeira variando-se os parâmetros da função de textura

- Como a textura é volumétrica, cortando-se a madeira, a textura vai aparecer como parte intrínseca do objeto



Mapeamentos : *criação de texturas*

- No exemplo de mapeamento 2D a textura foi obtida de uma foto real.
- No exemplo de mapeamento 3D a textura de madeira foi gerada de forma sintética, proceduralmente.
- Este um dos problemas do mapeamento 3D: o mapeamento é simples, mas a construção da imagem volumétrica não é trivial

Mapeamentos : *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

Mapeamentos : *criação de texturas*

- Escaneamento de imagens reais:
  - É bastante simples.
  - Pode gerar problemas quando a escala em que a textura foi capturada difere da escala da superfície a ser mapeada.
  - Uma solução consiste em colar pequenos pedaços de textura na escala adequada.
  - Isso pode gerar problemas de criação de uma textura periódica e com descontinuidades nas bordas dos ladrilhos.

Mapeamentos : *criação de texturas*

- Definição de forma algorítmica
  - Baseia-se na idéia de que um objeto gráfico pode ser representado por um algoritmo em alguma máquina virtual
    (Turing, por exemplo)

    Objeto = algoritmo(entrada, parâmetros)

  - Funciona bem para qualquer dimensão
  - A entrada é normalmente um conjunto de pontos e os parâmetros permitem um controle do objeto gerado
  - A semântica do objeto é obtida na sua reconstrução a partir do algoritmo aplicado ao conjunto de pontos

<u>Mapeamentos</u> : *criação de texturas*

- Existem duas abordagens para a definição algorítimica:
  - Procedural :
    - usada para gerar objetos de grande complexidade geométrica como nuvens, fogo,etc.
    - São flexíveis e fáceis de implementar.
    - Requerem uma boa intuição para controlar a semântica do objeto.
  - Modelagem física:
    - São métodos procedurais que utilizam modelos físicos para construir o algoritmo.
    - Muito difícil encontrar tais modelos.
    - São computacionalmente caros.

Mapeamentos : *função ruído*

- Objetos naturais apresentam grande irregularidade em sua geometria
- É por este motivo que é difícil a obtenção de bons resultados utilizando métodos determinísticos
- A irregularidade se traduz em um certo grau de aleatoriedade na geometria
- Além disso, existem altas e baixas freqüências em diferentes escalas
- Tais características estão relacionadas aos denominados fractais

Mapeamentos : *função ruído*

- A modelagem de objetos com tais características envolve três fatores:
  - Freqüência – determina a oscilação da irregularidade
  - Amplitude – determina a magnitude da irregularidade
  - Escala – determina nossa percepção da irregularidade

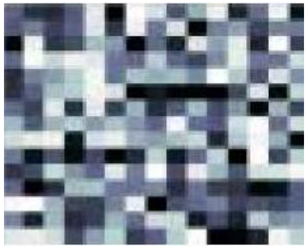
Mapeamentos : *função ruído*

- Geofrey Gardner foi o pioneiro ao tentar explorar esses fatores para gerar texturas relacionadas com objetos naturais
- Gardner utilizou a transformada de Fourier discreta para obter texturas bidimensionais de nuvens variando freqüência e amplitude (síntese espectral)
- Uma das limitações da síntese espectral é que a Transformada de Fourier não permite variação de escala
- Pode-se solucionar tal problema utilizando-se Wavelets

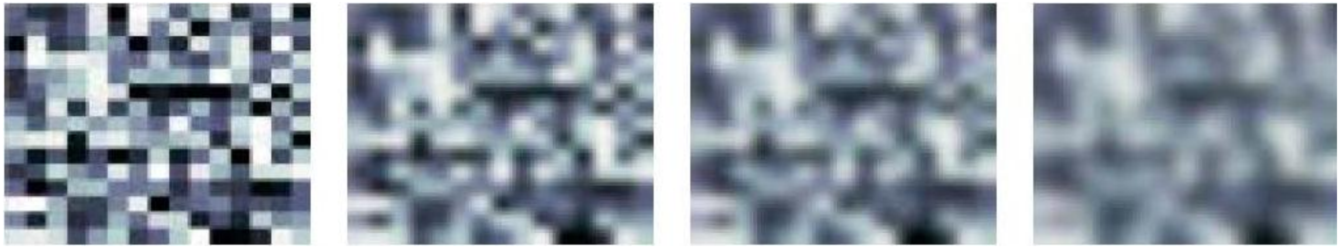
<u>Mapeamentos</u> : *função ruído por convolução*

- É um método ingênuo para construção de texturas no plano com variação de freqüência, amplitude e escala:

1. Construir um campo aleatório random(i,j) que associa um valor aleatório a cada vértice do reticulado

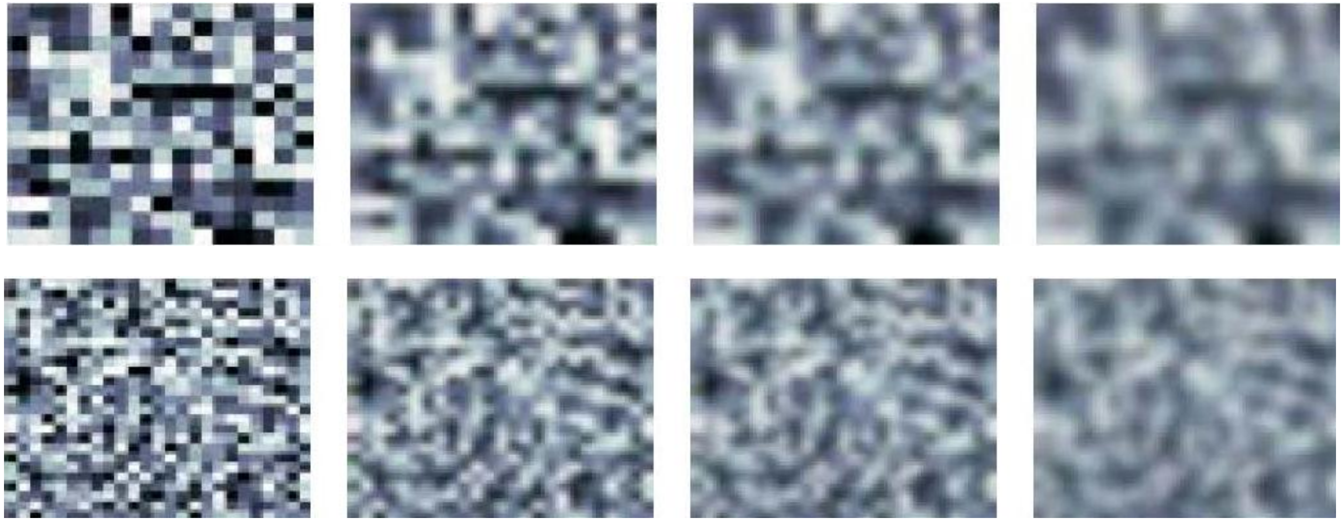


2. Aplicar sucessivas filtragens passa baixa para obter uma sequência de texturas em diferentes escalas



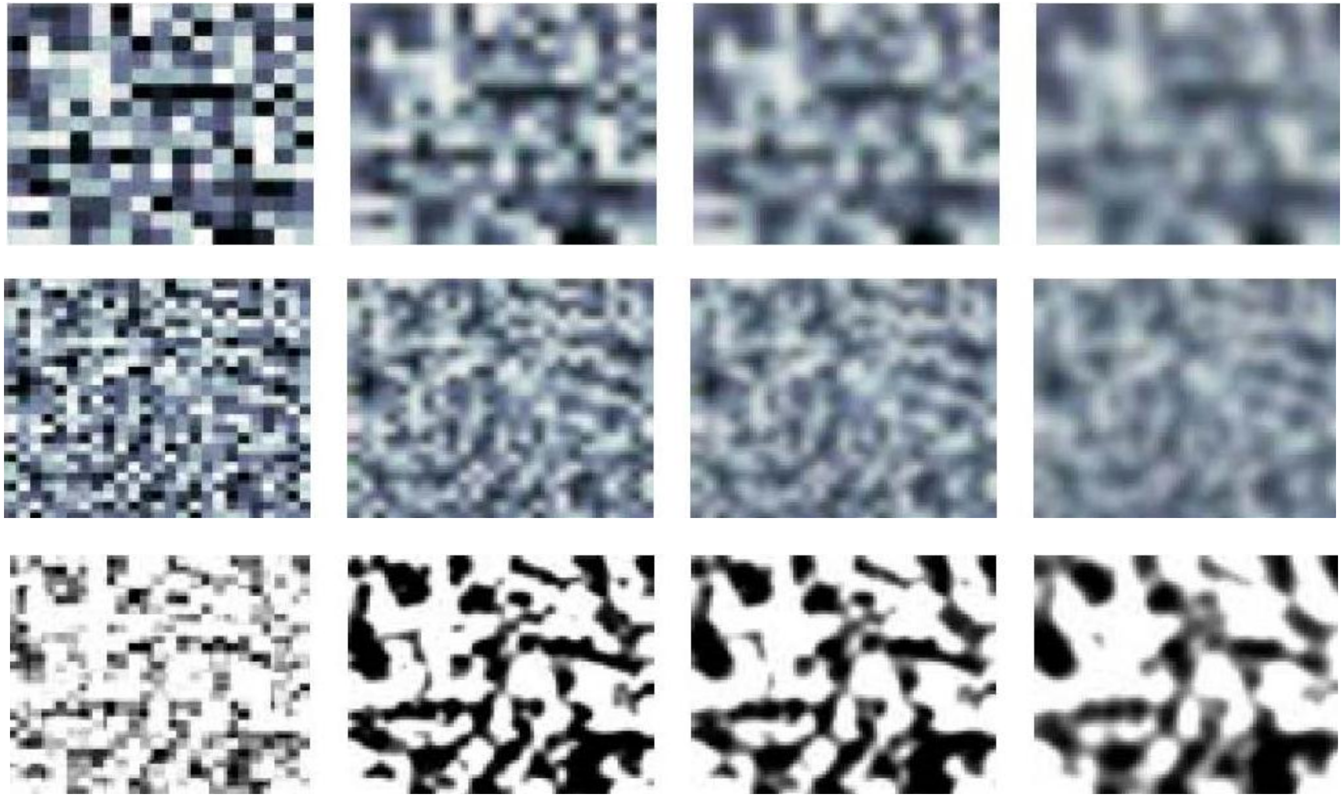
<u>Mapeamentos</u> : *função ruído por convolução*

3. Gerar uma novas imagens de mesmo tamanho, mas com diferentes resoluções e intensidades dos pixels usando 1, e repetir 2



<u>Mapeamentos</u> : *função ruído por convolução*

4. Para capturar a informação de resolução e escala em uma mesma imagem somamos as imagens duas a duas (nas figuras as amplitudes foram mantidas)



Mapeamentos : *função ruído por convolução*

- Problemas:
  - Não existe uma parametrização simples para controlar as características da textura gerada.
  - O processo e filtragem é caro computacionalmente
- Apesar dos problemas a idéia está na base das funções de ruído utilizadas atualmente.
- As funções de ruído utilizadas atualmente foram introduzidas por Ken Perlin e D. Peachey independentemente

Mapeamentos : *função ruído de Perlin*

- Características do ruído da função de Perlin
  - Está relacionado ao espaço ambiente.
  - Tem memória (não cria uma textura diferente ao re-processar o ruído).
  - Está presente em diversas escalas.
  - É parametrizável.

Mapeamentos : *função ruído de Perlin*

- Etapas para construção do ruído de Perlin:
  - Definir um reticulado no R^n.
  - Definir um campo pseudo-aleatório no reticulado.
  - Reconstruir o campo pseudo-aleatório.
  - A partir da função ruído construir uma função denominada turbulência em diferentes escalas, freqüências e amplitudes.

Mapeamentos : *função ruído de Perlin – o reticulado*

- O reticulado de dimensão m é produto cartesiano $J^m = J_n{}^m = J_n x J_n x \ldots J_n$, onde J_n é uma partição do intervalo $[0,n]$ pelos inteiros $0,1,2,\ldots,n$.

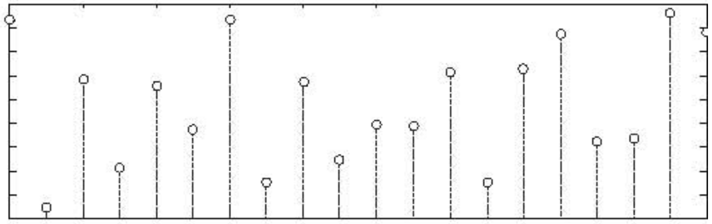
- Um vértice qualquer do reticulado J^m é indicado por v_j , onde j é um multi-índice $(i_1,i_2,\ldots,i_m)$.

Mapeamentos : *função ruído de Perlin – o campo aleatório*

- O campo aleatório pode ser de definido de várias formas.
- Citamos as 3 abaixo:
  - Escalar
  - Gradiente
  - Escalar-Gradiente

Mapeamentos : *função ruído de Perlin – o campo aleatório*

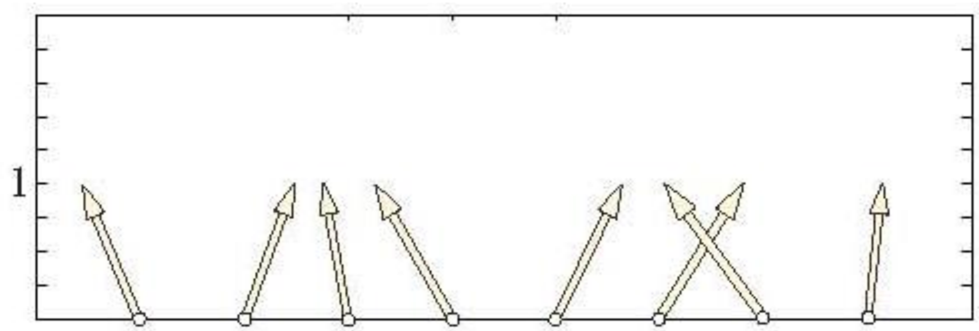
- O campo aleatório escalar: define um campo aleatório $N{:}J^m \rightarrow R$ associando a cada vértice v_j um número aleatório $N(v_j) \in [-1,1]$.



(a) Campo escalar aleatório.

<u>Mapeamentos</u> : *função ruído de Perlin – o campo aleatório*

- O campo aleatório gradiente: define um campo aleatório $g{:}J^m \to R^{m+1}$, onde $g(v_j) = (N(v_j),1) \in R^{m+1}$ e $N(v_j){:}\ J^m \to S^{m-1}$ *é um campo pseudo aleatório tomando valores na esfera unitária* $S^{m-1} \subset R^m$



(b) Campo gradiente aleatório.

Mapeamentos : *função ruído de Perlin – o campo aleatório*

- O campo aleatório escalar-gradiente: associa a cada vj do reticulado um valor escalar aleatório $N_1(v_j) \in [-1,1]$ e um vetor aleatório $(N_2(v_j),1) \in R^{m+1}$

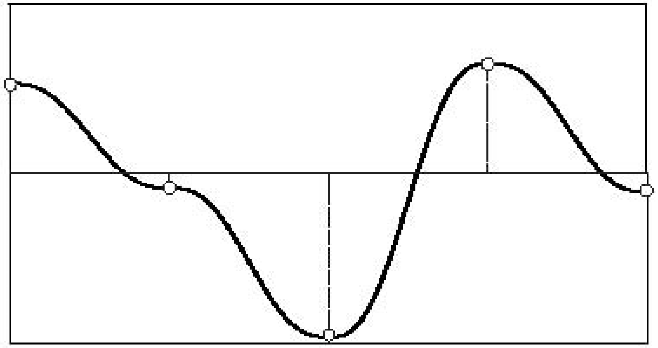
- Observações: O campo gradiente toma valores no conjunto S^{m-1} x {1}. Quando m=1, S^{m-1}= {1,-1} *e portanto* g=(-1,1) *ou* g=(1,1).

- Podemos substituir a esfera S^{m-1} *pelo disco* $D^m = \{x \in R^m ; |x| \leq 1\}$

Mapeamentos : *função ruído de Perlin – reconstrução*

- O objetivo da reconstrução é definir uma função *Noise*:$[0,n]^m \rightarrow R$, de classe $C^{k,}$ $k \geq 0$, a partir do campo aleatório definido em J^m.

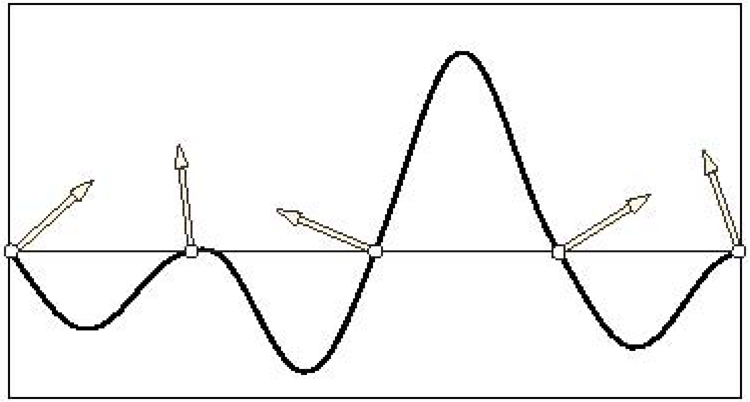
- No caso do campo escalar aleatório *Noise*(x) é obtida interpolando os valores v_j do campo conforme a figura abaixo, para o caso unidimensional, m=1.



(a) Função ruído escalar.

<u>Mapeamentos</u> : *função ruído de Perlin – reconstrução*

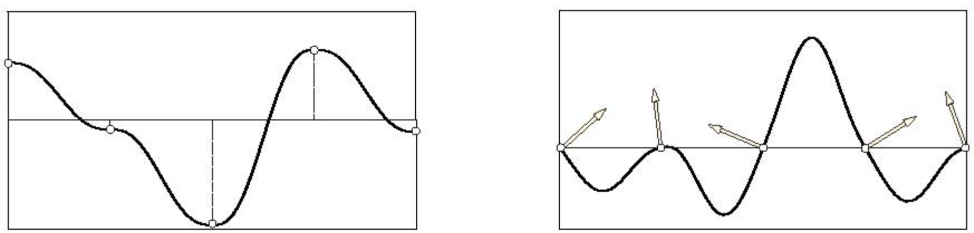
- No caso do campo aleatório gradiente, *Noise*(*x*)=0 para x= v_j e o vetor aleatório $g(v_j)$ deve ser normal ao gráfico (x, Noise(x)) para em x∈R^m com x= v_j



(b) Função ruído gradiente.

Mapeamentos : *função ruído de Perlin – reconstrução*

- A função ruído gradiente tem freqüências mais altas que a função ruído escalar já que a restrição de que Noise(x)=0 em $x=v_j$ faz com que a função oscile mais



(a) Função ruído escalar.

(b) Função ruído gradiente.

Figura 6. Reconstrução da função ruído.

- O fato da função possuir zeros nos vértices pode gerar padrões periódicos.

<u>Mapeamentos </u>: *função ruído de Perlin – reconstrução*

- Isto pode ser solucionado utilizando-se a função ruído escalar-gradiente

- A função ruído interpola os valores no campo escalar e seu gráfico é perpendicular ao campo gradiente.

Mapeamentos : *função ruído de Perlin – turbulência*

- Variando a escala, ou seja mudando simultaneamente a freqüência e a amplitude da função ruído fundamental Noise(x) podemos obter uma família de ruídos fundamentais

- Fixando um número real *p* e um inteiro *i* definimos a família de ruídos fundamentais

$$\mathit{Noise}_i(x) = \mathit{Noise}(2^i x)/p^i$$

- Aumentando *i*, o fator 2^i causa uma redução de escala e aumento das freqüências

- Além disso, ao aumentar *i* diminuímos a amplitude da função,para $p>1$

- Isto significa que o sinal oscila mais com amplitude menor (maior irregularidade)

Mapeamentos : *função ruído de Perlin – turbulência*

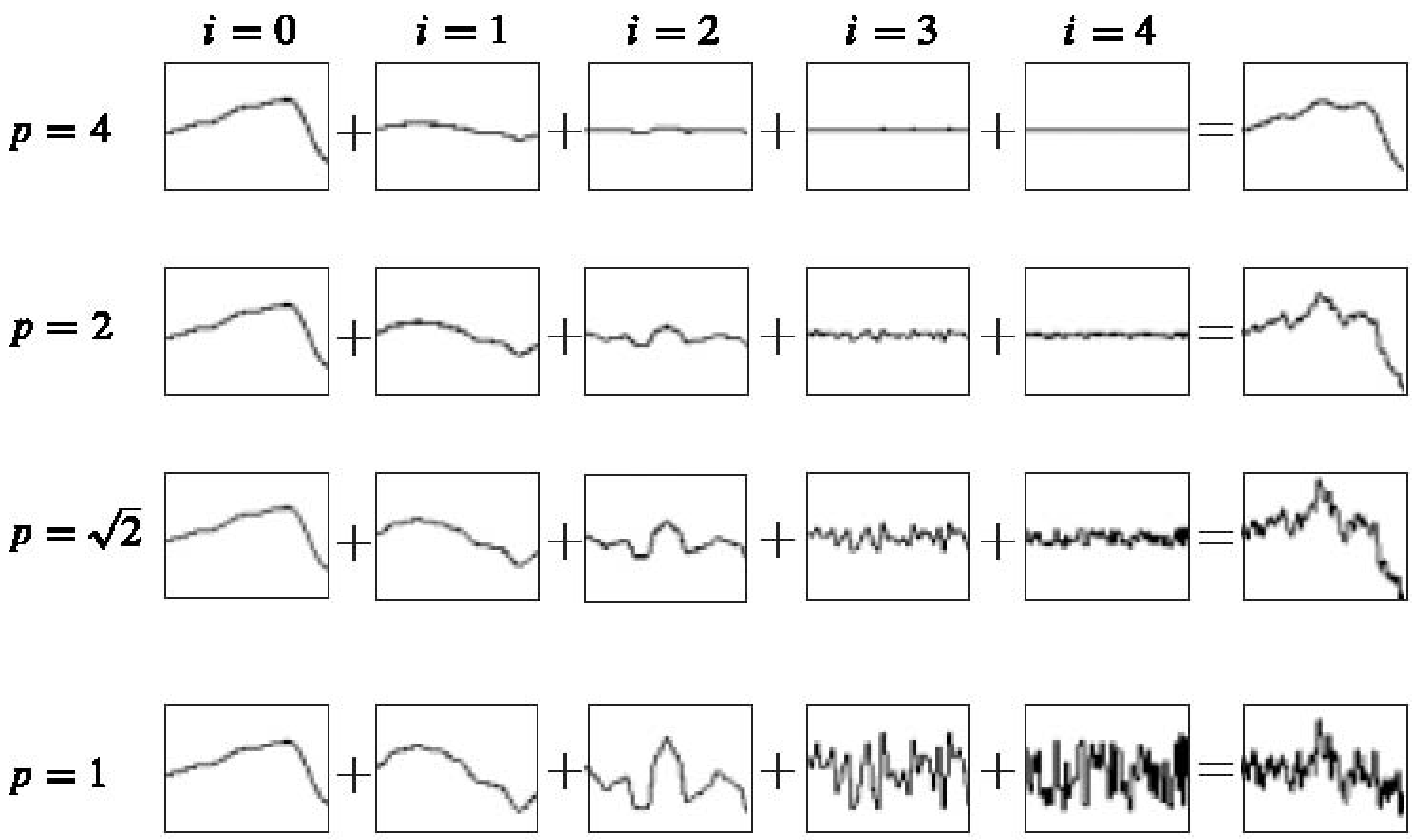


Figura 7. Funções ruído de Perlin com diferentes amplitudes.

<u>Mapeamentos</u> : *função ruído de Perlin – reconstrução de ruído escalar*

- Caso unidimensional

$$T(x) = \sum_{i=0}^{N-1} Noise_i(x) = \sum_{i=0}^{N-1} \frac{Noise_i(2^i x)}{p^i}$$

<u>Mapeamentos</u> : *função ruído de Perlin – exemplos*

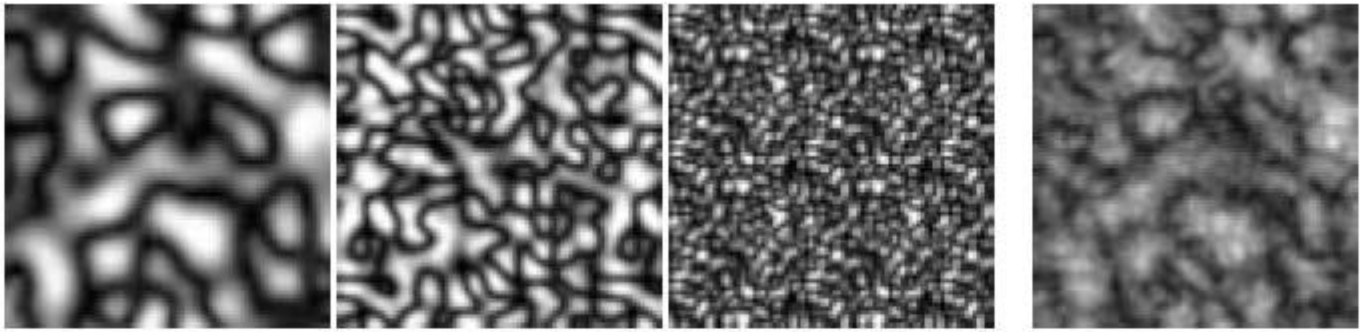


Figura 18. Ruído obtido pela soma de ruídos.

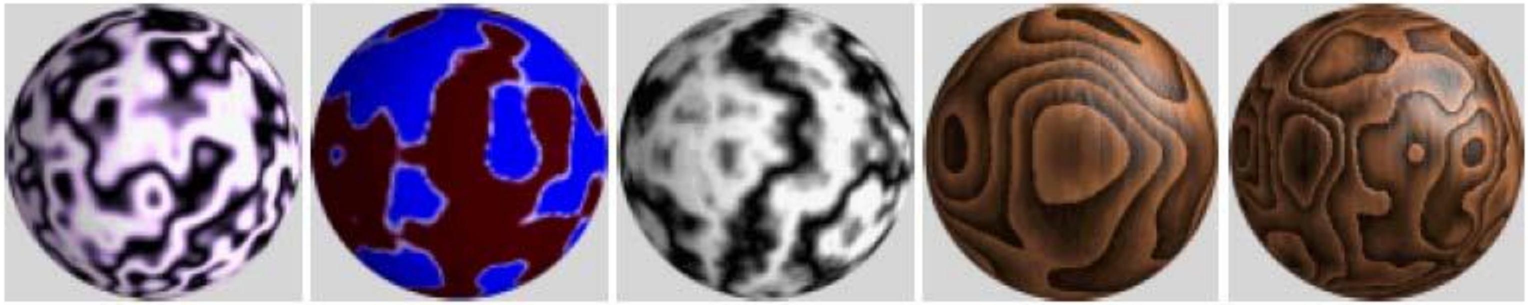
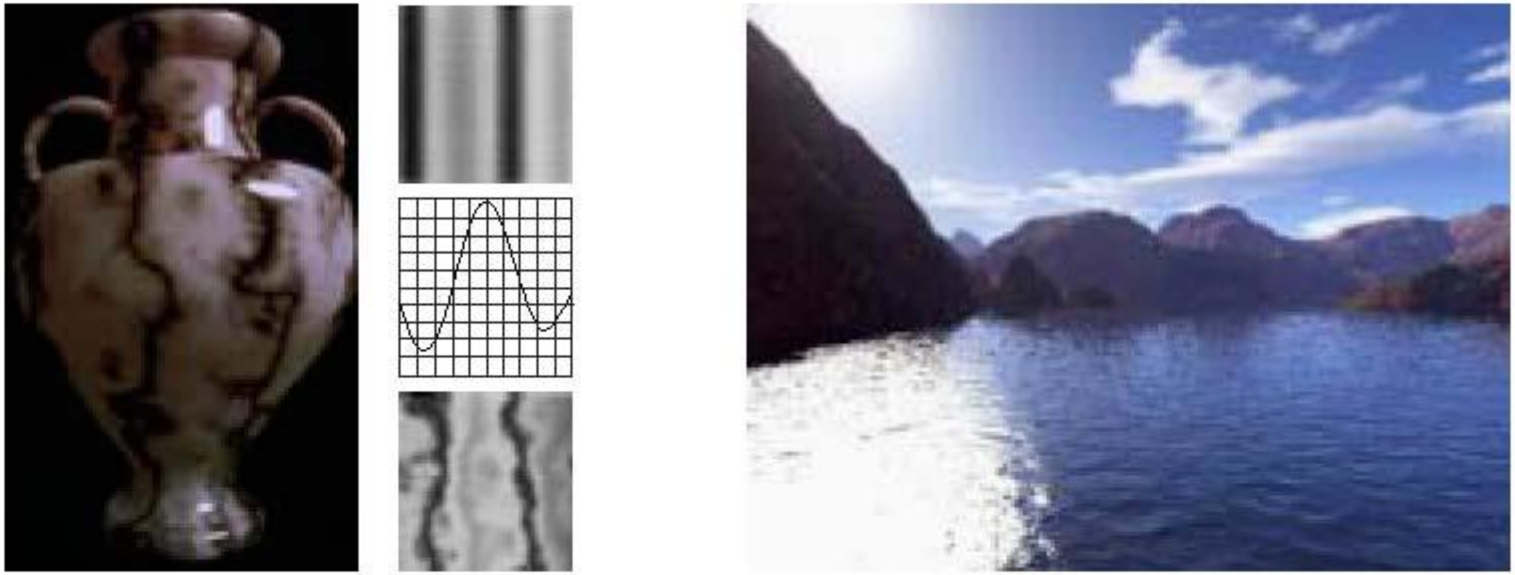


Figura 19. Esferas com texturas $3D$ definidas com a função ruído de Perlin.

<u>Mapeamentos</u> : *função ruído de Perlin – exemplos*



(a) (b) (c)

Figura 20. Textura $3D$ de mármore.

<u>Mapeamentos</u> : *métodos de mapeamento 2D*

- Os médos mais utilizados para definir uma transformação de mapeamento são:
  - Mapeamento por parametrização
  - Mapeamento por projeção
  - Transformações do plano
  - Mapeamento por superfície auxiliar

Mapeamentos : *métodos de mapeamento 2D – por parametrização*

- Tomemos uma parametrização φ:$U \rightarrow V$, da superfície no suporte geométrico da imagem

- O mapeamento T:$V \rightarrow U$ *é dado pela função inversa* $T = \varphi^{-1}:V \rightarrow U$

- *Portanto, a função de atributos* $g^* = f \circ T = \varphi^{-1} \circ f: U \rightarrow R^3$

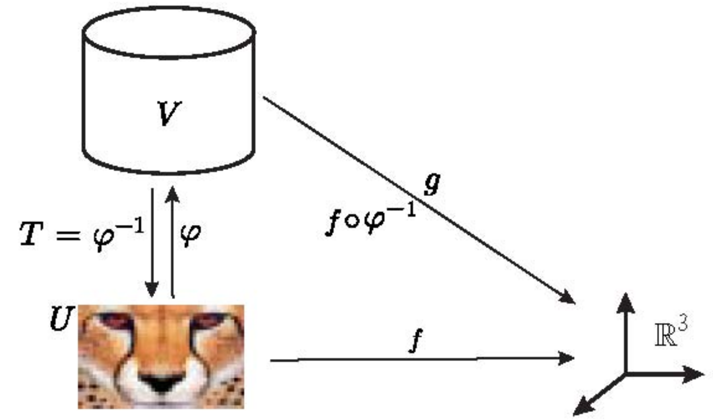
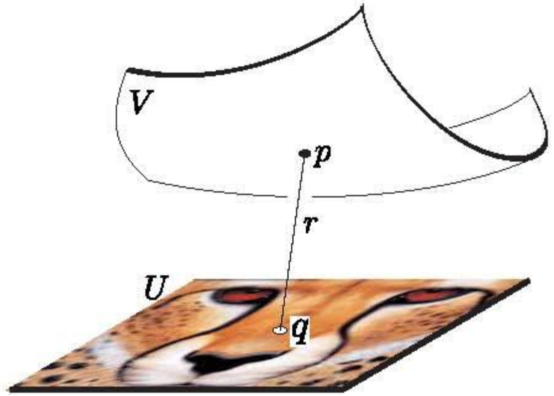


Figura 21. Mapemento por parametrização.

Mapeamentos : *métodos de mapeamento 2D – decal mapping*

- Para cada ponto $p \in V$ de $O_2(V,g)$, tomamos a semi-reta r com origem em p e cuja direção é normal a V.

- Se r intersecta o suporte U de $O_1(U,f)$ num ponto q, então $T(p) = q$



- Em outras palavras, o mapeamento T é obtido projetando cada ponto $p \in V$ num ponto do suporte U da imagem, ao longo da reta que passa por p e é normal a V em p.

<u>Mapeamentos</u> : *métodos de mapeamento 2D – transformação do plano*

- Seja *U* uma região retangular do plano, e considere uma imagem $g{:}U \subset R^2 \rightarrow R^3$.

- Se existe uma transformação $T{:}U \rightarrow V$ do retângulo *U* sobre outra região *V* do plano então podemos usar *T* para mapear a imagem *g* em *V.*

- *Para cada ponto* $p \in V$, a cor de *p* é dada pela cor *de* $T^{-1}(p)$

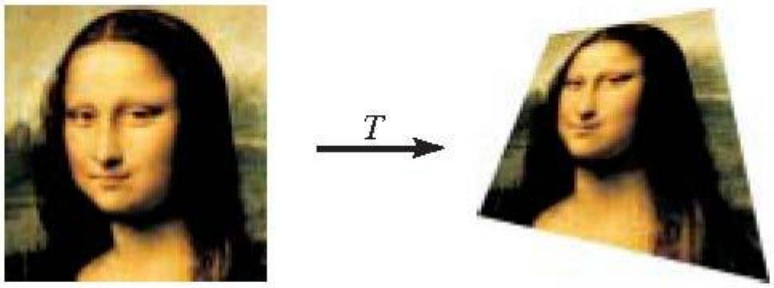


Figura 23. Mapeamento projetivo.

Mapeamentos : *mapeamento de textura 2D – por superfície auxiliar*

- Nem sempre temos uma boa parametrização da superfície do objeto a ser mapeado, o que dificulta o mapeamento por parametrização.

- O decal mapping, por outro lado, exige que o conjunto de retas normais a superfície seja bem comportado (não tenha auto-interseções)

- O mapeamento por superfície auxiliar evita os dois problemas combinando as duas técnicas

<u>Mapeamentos</u> : *cálculo de mapeamento 2D*

- Na prática, a maioria dos mapeamentos se reduz ao mapeamento de uma imagem em uma região do plano
- Temos então uma transformação bijetiva *f* de uma imagem $U \subset R^2 \rightarrow R^3$ em uma região do plano *V*
- As coordenadas de *U* são indicadas por (*x*,*y*) e as coordenadas de *V* por (*u*,*v*)

Mapeamentos : *cálculo de mapeamento 2D*

- Exemplo simples: escalamento linear f(x)=2x

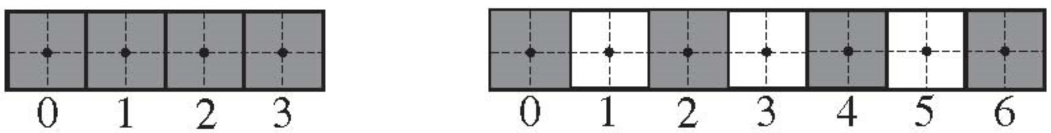


Figura 24. Escalamento por um fator 2.

- A transformação *f* é uma expansão e no caso discreto surgem buracos não pintados que precisam ser reconstruídos

- A transformação f^{-1} é uma contração e vários pixels vão se acumular num mesmo pixel

<u>Mapeamentos</u> : *cálculo de mapeamento 2D*

- Fatos gerais:

  - Expansões causam diminuição de freqüências e aparecem buracos.

  - Contrações causam aumento de freqüências e vários pixels são colapsados em um único pixel.

  - O segundo caso requer o uso de boa técnica de filtragem.

  - Problemas surgem ao lidarmos com o problema no domínio discreto

  - As propriedades de expansão e contração tem papeis fundamentais nos processo de mapeamento

Mapeamentos : *cálculo de mapeamento 2D – expansão e contração*

- A deformação de uma aplicação é a medida pela distorção que ela causa na distância entre dois pontos

- Uma aplicação $f{:}U \rightarrow V$ é uma:

  - **expansão** se $|f(x)-f(y)|>c|x-y|$, para $x,y \in U$, $c<1$
  - **contração** se $|f(x)-f(y)|<c|x-y|$, para $x,y \in U$, $c<1$
  - **isometria** se $|f(x)-f(y)|=|x-y|$

Mapeamentos : *cálculo de mapeamento 2D – expansão e contração*

- Uma estimativa local da natureza da deformação pode ser obtida considerando a derivada f’ da transformação que é dada por sua matriz Jacobiana J(f).

- Se f(x,y) = (u(x,y),v(x,y)) então

$$J(f) = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{bmatrix}$$

Mapeamentos : *cálculo de mapeamento 2D – expansão e contração*

- Podemos determinar se o tipo de deformação local avaliando a norma da matriz Jacobiana

$$|J(f)| = \max\left\{ \left| \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right) \right|, \left| \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right) \right| \right\}$$

- Uma deformação pode ser uma expansão em uma direção e uma contração em outra

<u>Mapeamentos</u> : *cálculo de mapeamento 2D – expansão e contração*

- O fato da deformação poder ser de um tipo ou de outro ao longo do domínio nos indica da necessidade da utilização de filtros que variam espacialmente.

- Uma deformação pode também variar conforme a direção o que nos leva aos filtros anisotrópicos.

Mapeamentos : *cálculo de mapeamento 2D – domínio contínuo*

- No domínio contínuo, um pixel é considerado um retângulo do reticulado.
- Desse modo, sua imagem pela transformação f é um quadrilátero curvilíneo

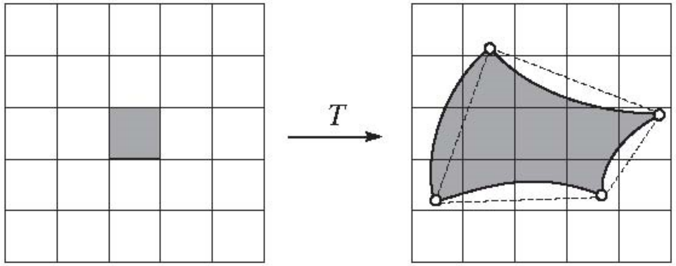


Figura 26. Imagem de um pixel no domínio contínuo.

Mapeamentos : *cálculo de mapeamento 2D – domínio contínuo*

- Podemos aproximar as curvas nas arestas por segmentos de reta.

- Assim, a imagem do pixel fica determinada pela imagem dos seus quatro vértices

<u>Mapeamentos</u> : *cálculo de mapeamento 2D – domínio contínuo*

- Para obter a imagem no domínio contínuo devemos reconstruí-la a partir dos valores nos vértices do reticulado da representação matricial através de interpolação.

- Uma vez no domínio contínuo aplicamos a transformação da imagem e em seguida re-amostramos para obter a imagem transformada em versão discreta.

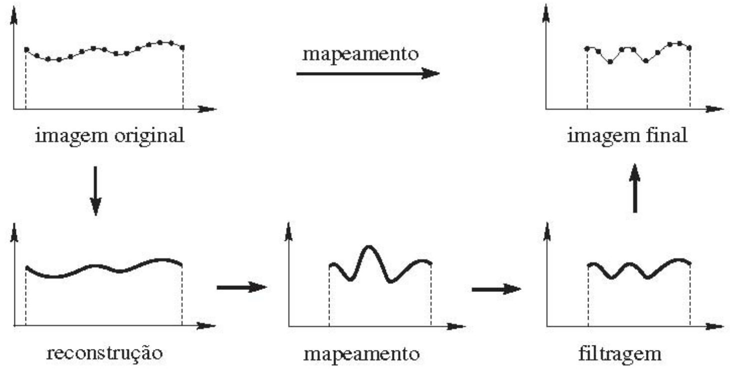
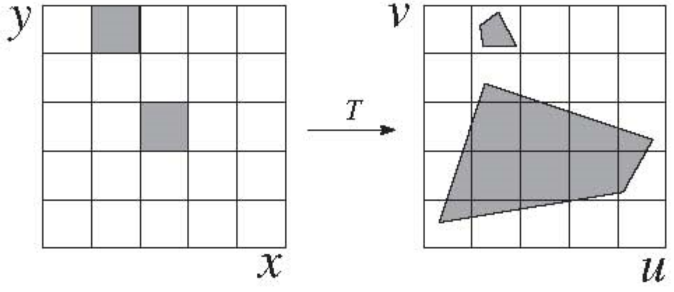


Figura 27. Etapas da transformação de uma imagem.

Mapeamentos : *cálculo do mapeamento 2D – mapeamento direto*

- Neste método aplicamos a transformação a cada pixel da imagem fonte e pintamos os pixels correspondentes na imagem destino

- A figura abaixo ilustra esse processo no caso de uma expansão e uma contração local



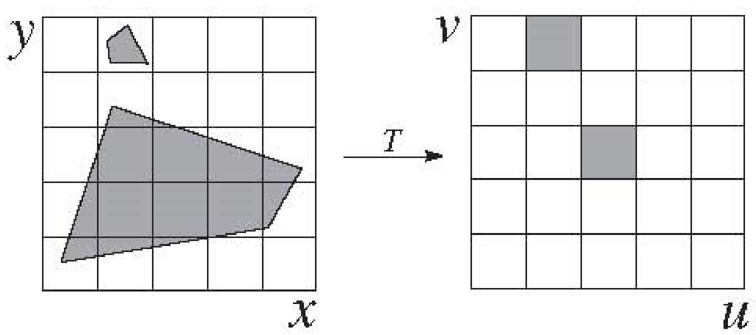
(a) Mapeamento direto.

Mapeamentos : *cálculo do mapeamento 2D – mapeamento direto*

- No caso da expansão, rasterizamos o polígono e pintamos os pixels levando em conta a área ocupada pelo polígono em cada pixel.
- No caso da contração, devemos juntar todos os fragmentos de quadriláteros para reconstruir a cor em cada pixel.
- No domínio contínuo não existe o problema de buracos
- Entretanto, ainda existem problemas na contração que é o caso em que temos maiores chances de *aliasing.*

Mapeamentos : *cálculo do mapeamento 2D – mapeamento inverso*

- No mapeamento inverso parte-se da imagem destino.
- Para cada pixel $p \in V$ calcula-se a imagem inversa $T^{-1}(p)$.
- A imagem inversa $T^{-1}(p)$ é processada para calcular a cor que será atribuída a *p.*
- Observe que se T é uma expansão então T^{-1} é uma contração e vice-versa.



(b) Mapeamento inverso.

Mapeamentos : *cálculo do mapeamento 2D – mapeamento inverso*

- No caso de expansão tomamos o baricentro do quadrilátero obtido por $T^{-1}(p)$ e atribuímos a p a cor do pixel mais próximos do baricentro

- No caso da expansão, em que há diminuição de freqüências não é necessário uma abordagem mais elaborada.

- Na contração rasterizamos o quadrilátero obtido por $T^{-1}(p)$ de modo a obter os pixels que são mapeados em p e fazemos uma filtragem para obter o valor da cor em *p*.

<u>Mapeamentos </u>: *cálculo do mapeamento 2D – domínio discreto*

- O processo de fazer o cálculo do mapeamento no domínio contínuo, apesar de robusto, é muito caro computacionalmente.

- Existem métodos mais baratos que trabalham no domínio discreto e que seguem a seguinte estratégia:

  - Usar mapeamento inverso.
  - Filtragem em paralelo com o processo de transformação

Mapeamentos : *cálculo do mapeamento 2D – domínio discreto*

- O tipo de filtragem depende das características da deformação.
- No caso de expansão (magnification) pode se utilizar um filtro vizinho mais próximo (nearest neighbor) o que em geral gera um aspecto pixelado.
- A filtragem bilinear produz resultados mais aceitaveis ainda que a um custo de borramento do resultado final.

<u>Mapeamentos</u> : *cálculo do mapeamento 2D – domínio discreto*

- No caso de contração (minification) o problema é mais grave já que há um aumento das freqüências.
- Pode-se utilizar ambos os filtros anteriores o que entretanto resulta em um resultado pobre em relação a redução de aliasing.
- Uma solução mais adequada requer a pré-filtragem e armazenamento em um estrutura apropriada.
- Está é a base do funcionamento da técnica denominada mipmapping.

<u>Mapeamentos</u> : *cálculo do mapeamento 2D – mipmapping*

- Mip significa (*multum in parvo* – muitos em pouco).

- A idéia consiste em gerar uma seqüência de texturas filtradas e subamostradas a partir da original

- A primeira textura da seqüência (nível 0) é a textura original.

- Dada uma textura de nível *i*, obtém-se uma nova textura de nível *i+1*, com um quarto da resolução do nível *i*, onde cada *texel* é a média dos quatro vizinhos do nível anterior.

Mapeamentos : *cálculo do mapeamento 2D – mipmapping*

(a) Pré-filtragem no mipmap.

(b) Pirâmide do mipmap.

Figura 29.

Mapeamentos : *cálculo do mapeamento 2D – mipmapping*

- O acesso a estrutura requer a escolha do nível de detalhe apropriado.

- Isso requer a determinação da área coberta pelo pixel no espaço da textura.

- Existem diferentes abordagens para determinação do nível de detalhe d (λ na OpenGL).

Mapeamentos : *cálculo do mapeamento 2D – mipmapping*

- Uma delas consiste em determinar o comprimento da maior aresta do quadrilátero (pixel transformado) no espaço da textura.

- Uma segunda consiste em tomar o maior valor absoluto das diferenciais parcias $\partial u/\partial x$, $\partial u/\partial y$, $\partial v/\partial x$, $\partial v/\partial y$

- f(x,y)=(u(x,y),v(x,y)) corresponde a transformação das coordenadas de tela do pixel nas coordenadas da textura.

- Logo, as diferencias medem a taxa de variação na coordenada de textura em função da variação com respeito aos eixos da tela.

- Por último, podemos utilizar a norma de $J(f)$.

Mapeamentos : *cálculo do mapeamento 2D – mipmapping*

- Uma vez obtido d associado a tranformação $T^{-1}(p)=f(p)=q$, podemos calcular o valor do pixel $p \in V$ acessando a estrutura da seguinte forma:

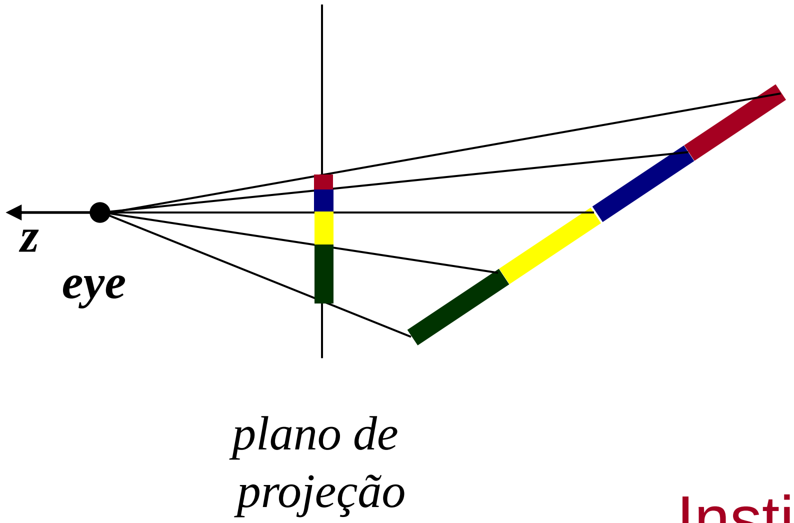
  - O nível j é dado por floor(d).
  - Calculamos os pixels q_j e q_{j+1} correspondentes a q nas imagens de nível j e j+1.
  - Calculamos as cores $C(q_j)$ e $C(q_{j+1})$ fazendo interpolação bilinear dos pixels vizinhos, em cada nível.
  - O valor final do pixel $q=T^{-1}$(p) é obtido interpolando $C(q_j)$ e $C(q_j+1)$.

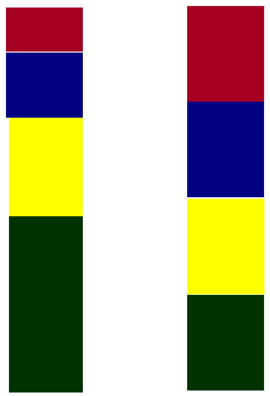
<u>Mapeamentos</u> : *Interpolação perspectiva*

- Infelizmente a interpolação linear não é correta para interpolarmos atributos como cor ou coordenadas de textura entre os vértices durante a rasterização .

- A figura abaixo à esquerda mostra um exemplo simples onde uma linha qualquer, dividida uniformemente em 4 cores, é projetada segundo uma projeção cônica.

- A figura abaixo à direita mostra em destaque como estas cores se projetam ao lado de uma interpolação linear para ressaltar as diferenças.





<u>Mapeamentos</u> : *Interpolação perspectiva*

- Na geração de fragmentos a partir dos vértices temos que levar em conta efeitos como este para obtermos realismo visual.

- Quando olhamos os dormentes ao longo de uma linha de trem ou em uma longa parede de tijolos esperamos ver este efeito. Se ele não estiver presente a imagem não parece real.

- Os algoritmos de rastreio interpolam entre vértices ao longo de retas no espaço projetado.

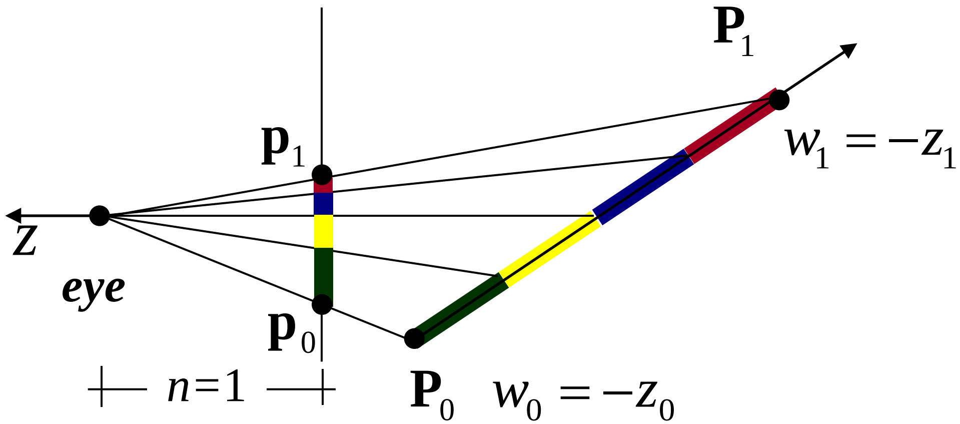
- Ou seja, capturam pontos igualmente espaçados neste espaço.

Mapeamentos : *Interpolação perspectiva*

- Para atribuirmos cor ou coordenadas de textura para estes pontos precisamos correlacionar as parametrizações do espaço projetado e do espaço cartesiano.

- A idéia básica é que para atribuirmos uma cor ou outro atributo a um ponto no espaço projetado nós precisamos saber o valor do atributo interpolando no espaço cartesiano onde a variação é linear.

Mapeamentos : *interpolação perspectiva*



$$\mathbf{P}(s) = (1-s)\mathbf{P}_0 + s\mathbf{P}_1$$

$$w(s) = (1-s)w_0 + sw_1$$

$$\mathbf{p}(t) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1$$

$$\mathbf{p}(s) = \frac{\mathbf{P}(s)}{w(s)} = \frac{(1-s)\mathbf{P}_0 + s\mathbf{P}_1}{(1-s)w_0 + sw_1}$$

$$\mathbf{p}(s) \equiv \mathbf{p}(t) \qquad (1-t)\frac{\mathbf{P}_0}{w_0} + t\frac{\mathbf{P}_1}{w_1} = \frac{(1-s)\mathbf{P}_0 + s\mathbf{P}_1}{(1-s)w_0 + sw_1}$$

$$t = \frac{sw_1}{(1-s)w_0 + sw_1}$$

$$s = \frac{tw_0}{(1-t)w_1 + tw_0}$$

Mapeamentos : *interpolação perspectiva de atributos*

$$I = (1-s)I_o + sI_1$$

$$s = \frac{tw_0}{(1-t)w_1 + tw_0}$$

$$I = (1-\frac{tw_0}{(1-t)w_1 + tw_0})I_o + \frac{tw_0}{(1-t)w_1 + tw_0}I_1$$

$$I = \frac{(1-t)w_1 I_0 + tw_0 I_1}{(1-t)w_1 + tw_0}$$

$$\boxed{I = \frac{(1-t)\left(I_0/w_0\right) + t\left(I_1/w_1\right)}{(1-t)\left(1/w_0\right) + t\left(1/w_1\right)}}$$

OpenGL Spec

$$\boxed{f = \frac{(1-t)f_a/w_a + tf_b/w_b}{(1-t)\alpha_a/w_a + t\alpha_b/w_b}}$$

Mapeamentos : *Interpolação perspectiva de atributos homogêneos*

- Os atributos de textura no OpenGL™ podem ser fornecidos em coordenadas homogêneas, que antes de serem utilizados, são divididos pela coordenada q, que corresponde ao w do espaço geométrico.

- Ou seja, se u e q são as coordenadas homogêneas de textura de um ponto a coordenada de textura cartesiana dele seria u/q.

- A interpolação linear deste atributo no espaço cartesiano também segue a mesma regra.

<u>Mapeamentos</u> : *interpolação perspectiva de atributos homogêneos*

$$\frac{u}{q} = \frac{(1-s)u_o + su_1}{(1-s)q_o + sq_1} \qquad s = \frac{tw_0}{(1-t)w_1 + tw_0}$$

$$\frac{u}{q} = \frac{(1-\frac{tw_0}{(1-t)w_1+tw_0})u_o + \frac{tw_0}{(1-t)w_1+tw_0}u_1}{(1-\frac{tw_0}{(1-t)w_1+tw_0})q_o + \frac{tw_0}{(1-t)w_1+tw_0}q_1}$$

$$\frac{u}{q} = \frac{(1-t)w_1u_0 + tw_0u_1}{(1-t)w_1q_0 + tw_0q_1}$$

$$\boxed{\frac{u}{q} = \frac{(1-t)\,u_0/w_0 + t\,u_1/w_1}{(1-t)\,q_0/w_0 + t\,q_1/w_1}}$$

OpenGL Spec

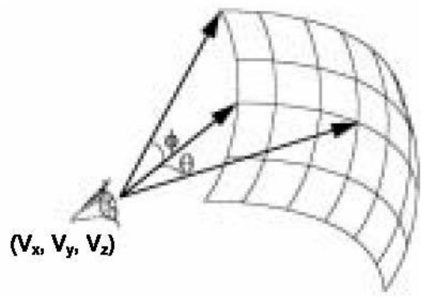
$$\boxed{f = \frac{(1-t)f_a/w_a + tf_b/w_b}{(1-t)\alpha_a/w_a + t\alpha_b/w_b}}$$

Mapeamentos : *environment mapping*

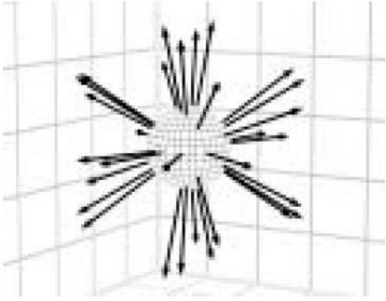
- Do ponto de vista perceptual não precisamos de modelos 3D já que percebemos o mundo através de sua projeção em nossa retina.
- Suponha que seja possível fotografa a cena de todas as posições e de todas as orientações possíveis.
- A visualização de um ponto de vista poderia ser construida simplesmente através de uma montagem fotográfica apropriada.

Mapeamentos : *environment mapping*

- As idéias anteriores podem ser formuladas matematicamente através do conceito de ***função plenóptica***.
- A função plenóptica é uma função $P(x,y,z,\theta,\varphi,t)$ que associa a cada ponto (x,y,z) e cada direção (θ,φ) o comprimento de onda λ percebido pelo observador no instante t



(a) Amostra da função plenóptica.



(b) Feixe de raios em p_0.

Figura 32.

Mapeamentos : *environment mapping*

- Vamos fazer as seguintes simplificações para melhor compreender a função plenóptica:
  - O meio não é participativo, isto é, a função plenóptica é constante ao longo de cada raio.
  - O tempo está fixado
- Sob essas hipóteses, para cada posição p0(x0,y0,z0), a função plenóptica fica completamente especificada pelo feixe de raios com origem no ponto p0.
- Portanto ela fica especificada por seus valores numa esfera de raio R>0 com centro em p0.

Mapeamentos : *environment mapping*

- As diferentes formas de amostragem da função plenóptica são denominadas genéricamente como ***environment mapping***.
- Ao invés da esfera, pode-se utilizar qualquer outra superfície S cujo ângulo sólido seja igual a 4π e tal que cada raio do feixe com centro em p0 intersecte S em apenas um ponto.
- Uma superfície que satisfaz as condições acima é denominada superfície plenóptica.
- Exemplos de superfícies plenópticas: esfera, cubo, cilindro.
- Observe que o cilindro não tem ângulo sólido igual a 4π (entretanto, este é seu limite quando a altura cresce para o valor infinito).

Mapeamentos : *environment mapping*

- Com base na noção de função plenóptica podemos fotografar um ambiente, de forma apropriada, e fazer um mapeamento de textura das imagens obtidas numa das superfícies mencionadas.
- Exemplos:
  - Mapeamento de ambiente cilíndrico
  - Panoramas virtuais

Mapeamentos : *mapeamento de ambiente cilíndrico*

- O cilindro tem uma vantagem em relação as outras funções plenópticas porque a parametrização por coordenadas cilíndricas é uma isometria.

- Isto significa que se uma imagem possui largura igual ao comprimento total do cilindro ($2\pi r$) então ela será mapeada sem distorções.

- Uma imagem que satisfaz tal condição é denominada panorama cilíndrica

<u>Mapeamentos</u> : *mapeamento de ambiente cilíndrico*

- A figura abaixo mostra uma imagem construída através da costura de varias imagens tiradas girando uma câmera fixada numa posição.

Figura 33. Panorama cilíndrico.

<u>Mapeamentos</u> : *mapeamento de reflexão*

- Considere um objeto *O* num determinado cenário

- Tomemos uma superfície plenóptica *M* e façamos o mapeamento de ambiente do cenário em M.

- O mapeamento de reflexão consiste em utiliza o mapeamento de ambiente de forma a simular a reflexão do ambiente no objeto O.

- O mapeamento de reflexão é uma aproximação de primeira ordem para o traçado de raios.

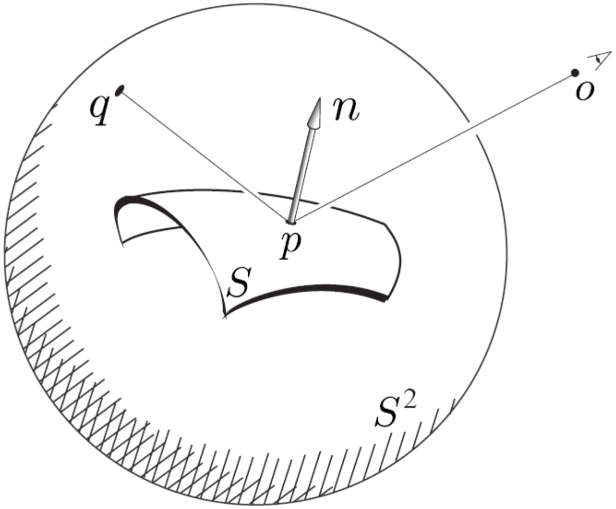
Mapeamentos : *mapeamento de reflexão*

- Uma vez criado o mapeamento de ambiente na superfície plenóptica, precisamos apenas calcular o mapeamento correto para obter o efeito de reflexão desejado.

- Esse mapeamento é calculado com base no vetor refletido pela superfície a partir do observador.

<u>Mapeamentos</u> : *mapeamento de reflexão*

- Para calcular o valor do mapeamento no ponto $p \in O$, consideramos o vetor unitário n normal a superfície O no ponto p, e o vetor unitário $v = pO$ na direção do observador.



- O vetor reflexão é dado por r = v - <v,n>n e a a equação paramétrica do raio refletido é $\gamma(t) = p + rt$, $t \geq 0$.

- O valor do mapeamento no ponto p é dado pela interseção de $\gamma(t)$ com a superfície plenóptica M .

- A mudança do atributo deve ser feita na componente especular de Phong.

<u>Mapeamentos</u> : *mapeamento de reflexão*

(a) Mapa esférico 360°.

(b) Mapeamento

Figura 41. Mapa esférico no teapot.

Mapeamentos : *mapeamento de reflexão – exemplos de aplicações*

- (a) aproximação do traçado de raios
- (b) modificação da componente especular com textura de altas e baixas freqüências para geração de aparência metálica

(a) (b)

Figura 35. Exemplos de mapeamento de reflexão.

Mapeamentos : *mapeamento cúbico*

- Se a superfície plenóptica for um cubo utilizamos seis imagens da cena, obtidas do mesmo ponto, girando a câmera de 90 graus de modo a cobrir todo o ambiente.

- Em seguida mapeamos cada imagem em uma das faces do cubo

Mapeamentos : *mapeamento cúbico*

- A direção do raio refletido, que não precisa ser normalizado, é utilizada para determinar qual face do cubo será considerada.
- A coordenada do vetor com maior magnitude seleciona a face correspondente (exemplo: o vetor (-3.2, 5.1, -8.4) seleciona a face –Z.
- As demais coordenadas são divididas pela de maior magnitude (8.4) e normalizadas para o intervalo [0,1] (exemplo: ((-3.2/8.4+1)/2,(5.1/8.4+1)/2) = (0.31,0.80).
- Se dois vértices refletem em faces diferentes é necessário uma técnica apropriada para interpolar os valores dos pixels entre eles.

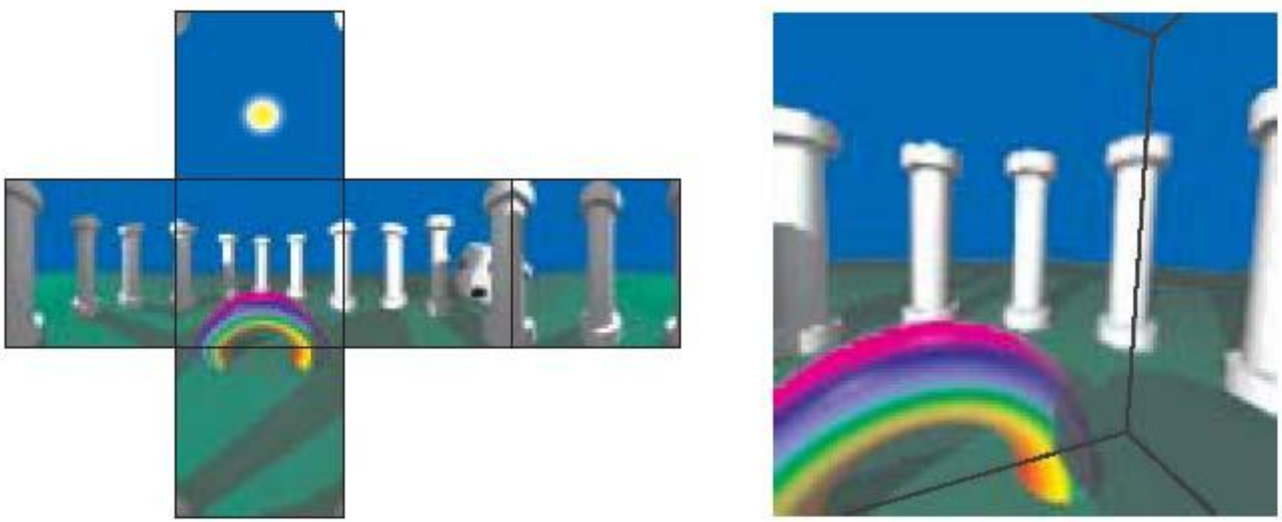
Mapeamentos : *mapeamento cúbico*

- Um solução em software consiste em subdividir o polígono problemático ao longo da aresta do cubo.

- Uma solução em hardware é colocar o vetor refletido e o cálculo das faces no acelerador gráfico de modo que a computação seja feita por pixel.

<u>Mapeamentos</u> : *mapeamento cúbico*

- Exemplo (mapeamento cúbico)



(a) Mapa do ambiente. (b) Visualização.

Figura 36. Panorama virtual com mapeamento cúbico (Gomes *et al.* , 1998).

Mapeamentos : *rugosidade*

- Nesta técnica, introduzida por Blinn(1978), a imagem mapeada é utilizada para fazer uma perturbação do vetor normal à superfície.

- 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:

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

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

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

Mapeamentos : *rugosidade*

- Nesta técnica, introduzida por Blinn(1978), a imagem mapeada é utilizada para fazer uma perturbação do vetor normal à superfície.

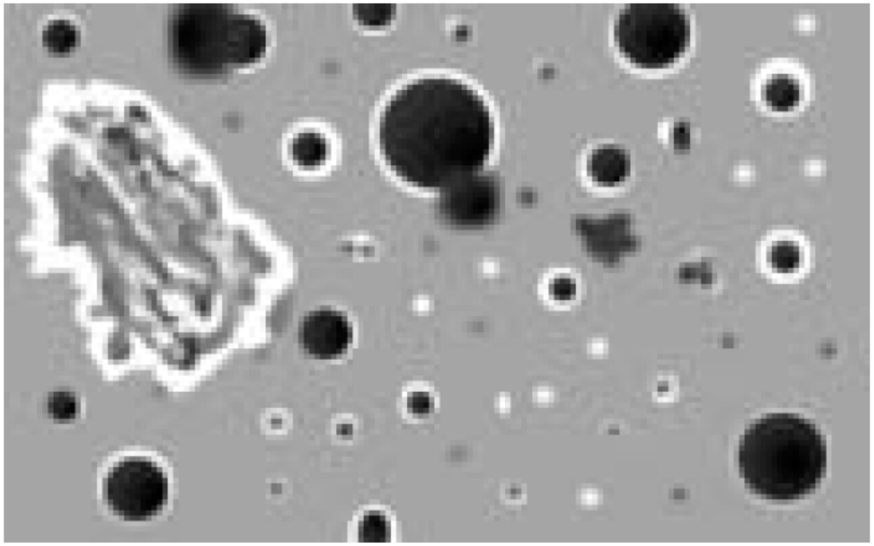
- Supondo que a imagem é dada por b(u,v), a superfície por p(u,v) e a normal N(u,v) temos que o vetor perturbado é dado por:

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

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

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

Mapeamentos : *rugosidade*



(a) Textura fonte.

(b) Mapeamento.

Figura 37. Mapeamento de rugosidade.

Mapeamentos : *rugosidade*

Figura 38. Face de uma moeda gerada com mapeamento de rugosidade.

Mapeamentos : *deslocamento*

- 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 oconsiste em deslocar realmente a superfície

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

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

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

<u>Mapeamentos</u> : *deslocamento*

(a) De rugosidade. (b) De deslocamento.

Figura 39. Mapeamentos.