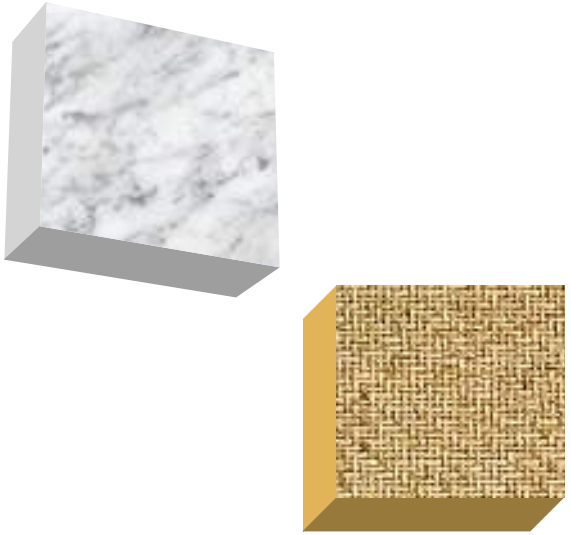
Realismo Visual



Texture mapping

E outras tecnicas

UFF – 2019

(p. 256-284)

Texturas: Texture mapping – permite dar a uma face uma aparência bem complexa!

Catmull em 1974, em sua tese de doutorado, foi o primeiro a adicionar detalhes de textura na superfície de modelos 3D por mapeamentos

Fazer um **texture map** em uma superfície é como aplicar uma **folha de papel auto adesivo nela para lhe dar um aspecto** semelhante **a textura do** desenho deste papel !

Edwin Earl "Ed" Catmull (1945, ...)

Americano formado em **ciência da computação e atualmente presidente da Pixar e Disney Animation.**

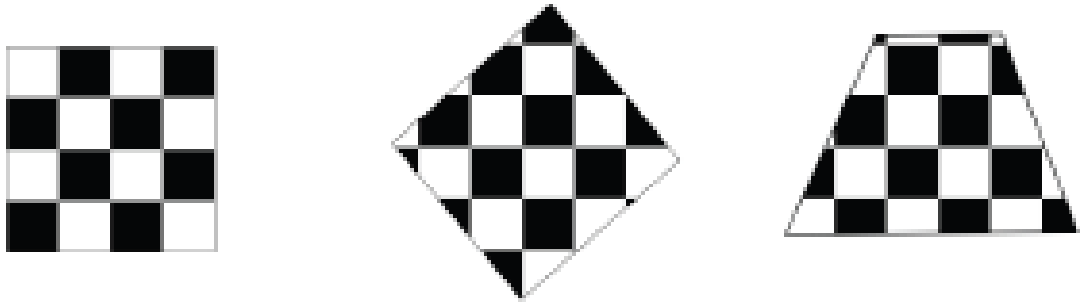
Tem feito diversas contribuições a CG.

Em 2001, recebeu um **Oscar** "for significant advancements to the field of motion picture rendering".

Em 2006, foi premiado com a **IEEE von Neumann medal** pelas suas contribuições na ***modelagem em CG, animação e rendering***.

Texture mapping pode ser:
Paramétrico ou não-Paramétrico:

Pode ser fixo (em tamanho ou orientação) ou se deformar com o objeto

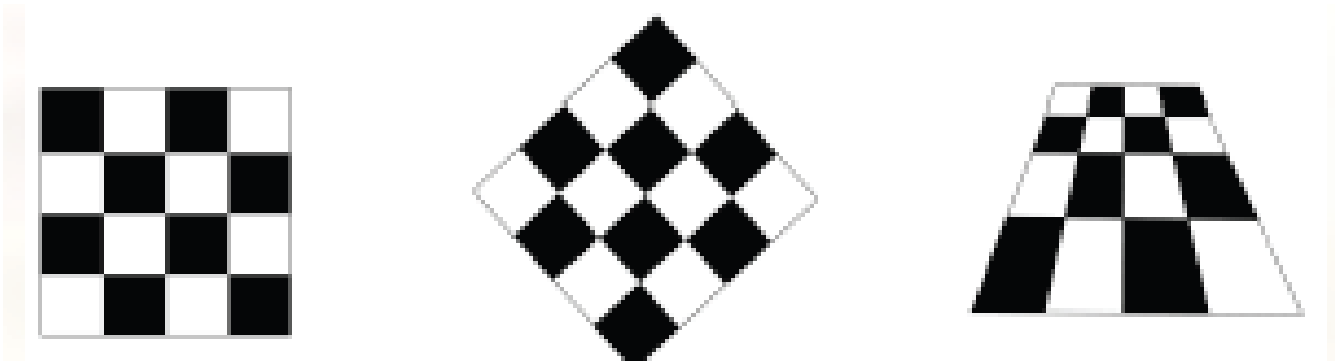


Não-Paramétrico: a textura parece sempre igual, como que se Fosse usada uma FORMA , ou se a face fosse um CORTADOR SOBRE UM TECIDO OU MATERIAL que representasse a textura.

Paramétrico

se deforma com o objeto

Parece que A TEXTURA É do material DO OBJETO!
TEM A MESMA deformação QUE ELE NA CENA OU NA ANIMAÇÃO!



Aplicar um **texture map** é

Associar às coordenadas da face , as coordenadas do mapa de textura.

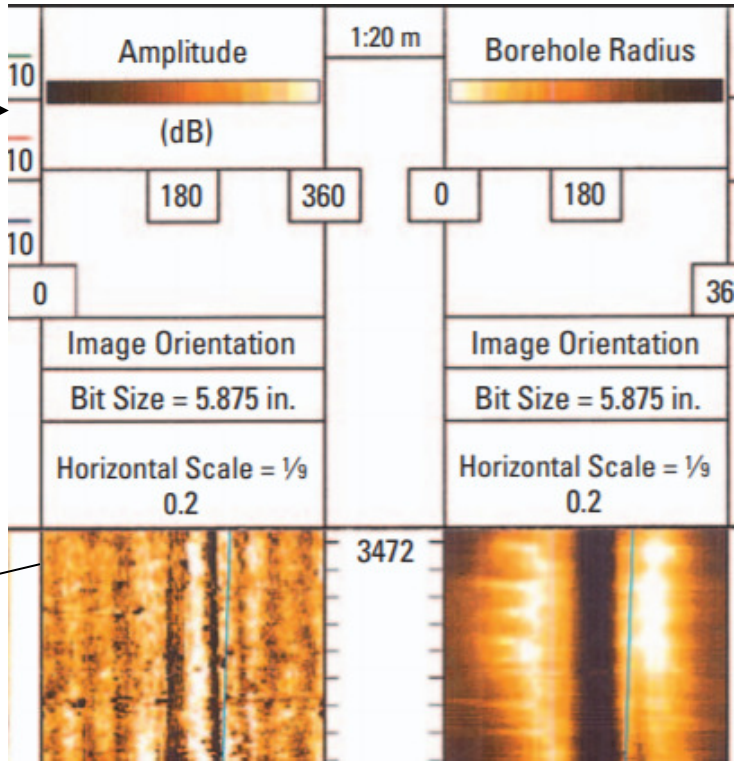
Que pode ser

1D : (false colors)

2D (coordenadas UV)

3D (sculptures)

ou até nD.



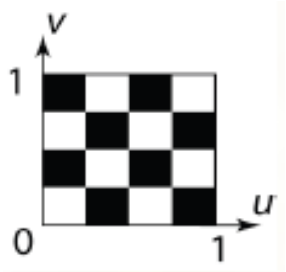
280, 300,100, 60, 1

Um mapa 3D

usadopara tar realismo na
Criação de objetos por
Torneamento ou
tecnicas de escultura

ESPAÇO DE TEXTURA (u,v)

As texturas são definidas em um sistema cartesiano normalizado [0,1]x[0,1]



E depois usadas para

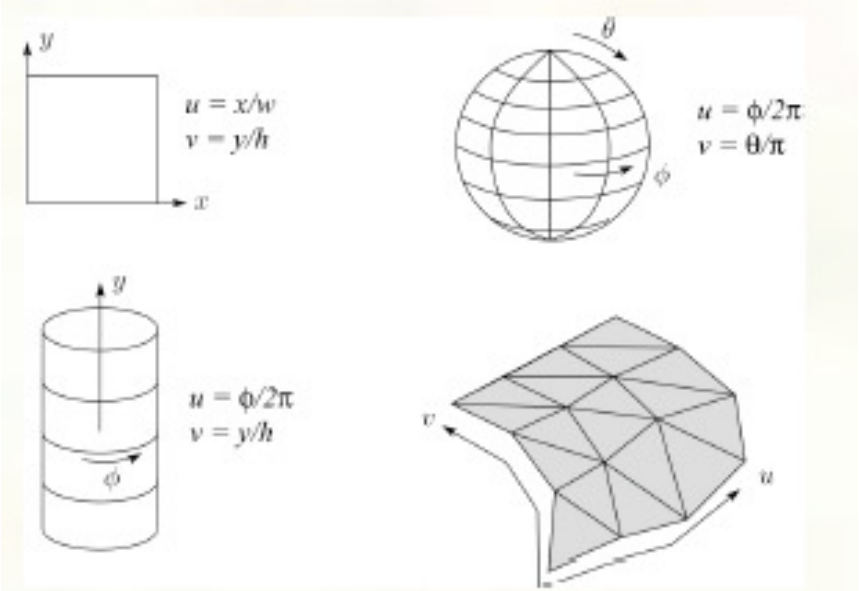
"encapar" nossos objetos,

De modo que, quando

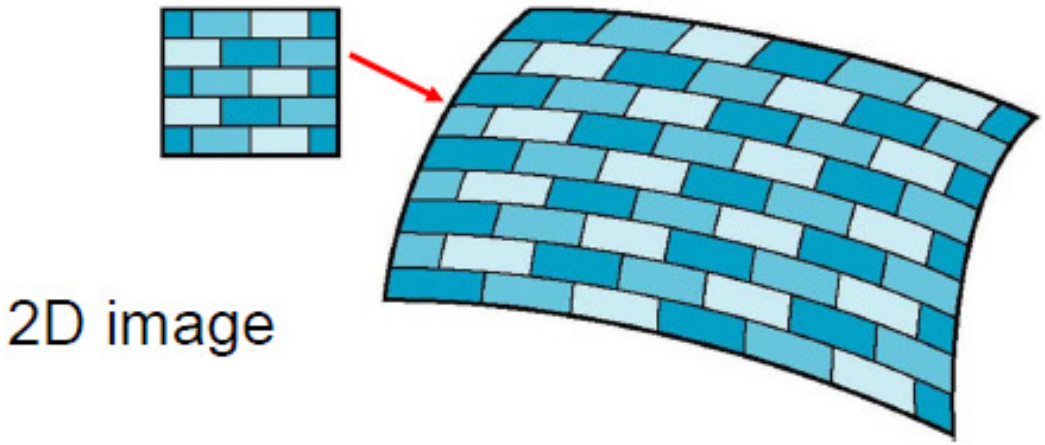
A superfície de mover ,

a textura

vá junto!



Sistemas de coordenadas envolvidos:



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

Forward mapping

Forwards x backwards mapping

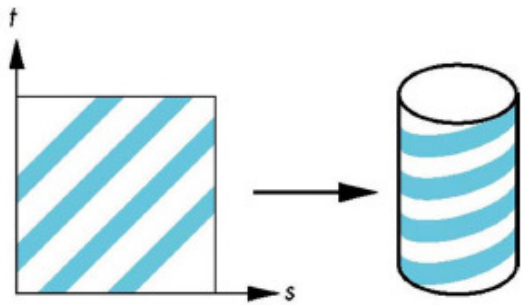
Mapeamento em 2 partes

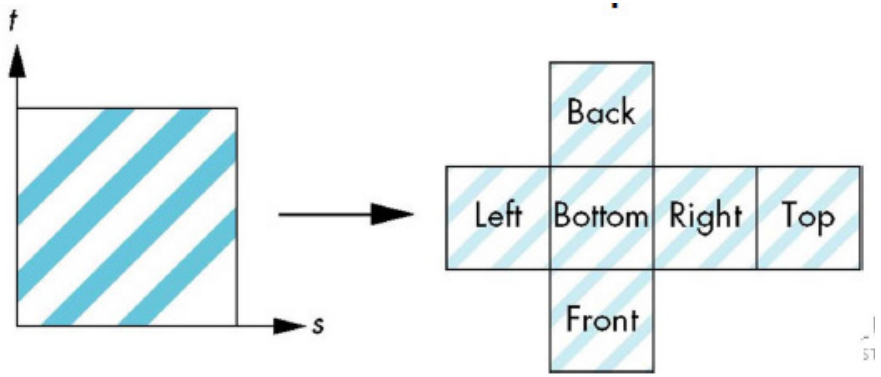
1- textura em uma forma intermediaria mais simples (cubo – projeções ortográficas, cilindro de altura h e raio r, esfera de raio r)

$$x = r \cos 2\pi u$$
$$y = r \sin 2\pi u$$
$$z = v/h$$

$$s = u$$
$$t = v$$

$$x = r \cos 2\pi u$$
$$y = r \sin 2\pi u \cos 2\pi v$$
$$z = r \sin 2\pi u \sin 2\pi v$$

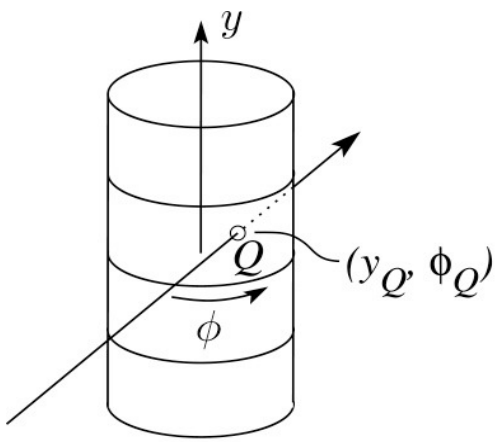




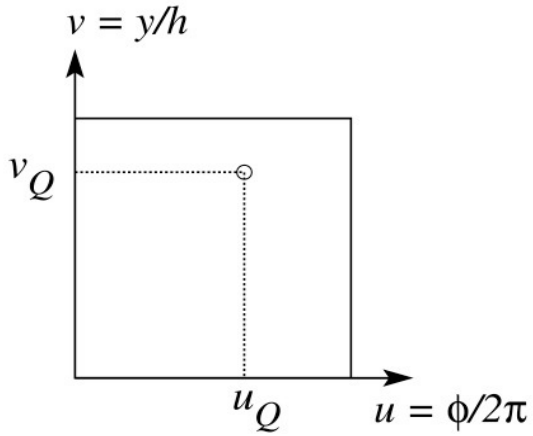
O mapa de textura é uma imagem

Essa imagem deve ser convertida para [0,1]x[0,1] e depois para as coordenadas onde será mapeada

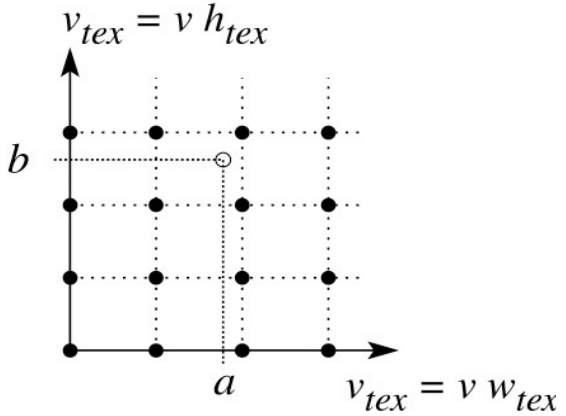
(u_{tex},v_{tex}) entre os valores: ([0.. w_{tex}], [0.. ht_{ex}])



Ray intersection

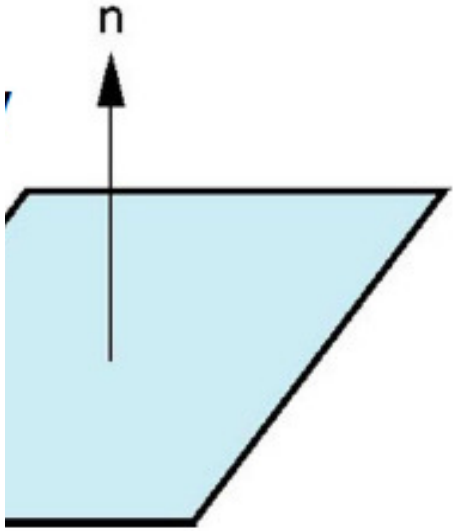


Mapping to
abstract texture coords



Mapping to
texture pixel coords

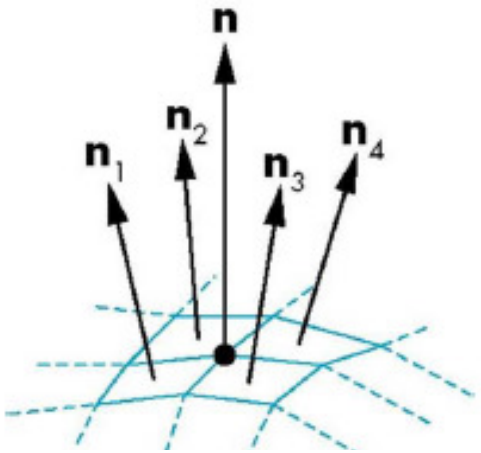
Normais



Superfícies planas:

Equação do plano:

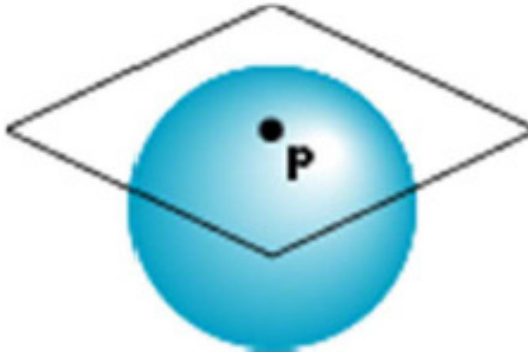
$ax+by+cz+d = 0$



Normal = (a,b,c)

Os por 3 pontos do plano: p0, p1, p2

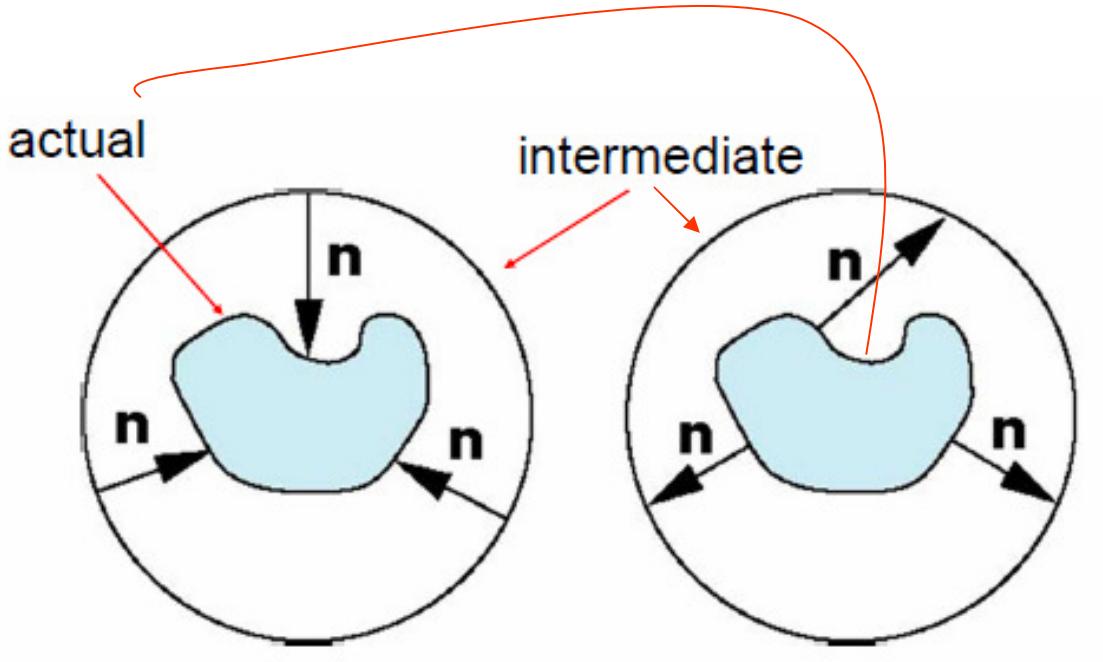
$$\mathbf{n} = (p_2\text{-}p_0) \times (p_1\text{-}p_0)$$



Círculos: raio do ponto ao centro

Mapeamento em 2 partes

2- da superfície mais simples para o objeto real, usando as normais (da intermediária para o objeto e do objeto para a intermediária)



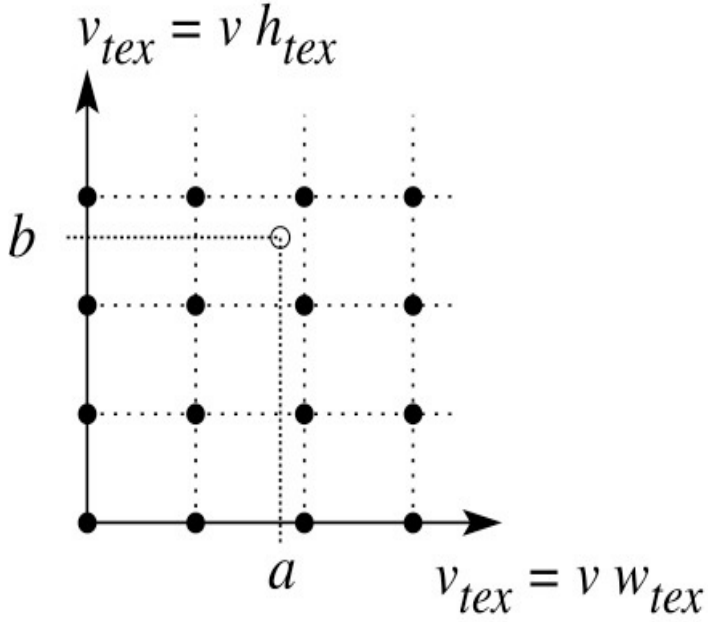
Mapeamento direto e inverso

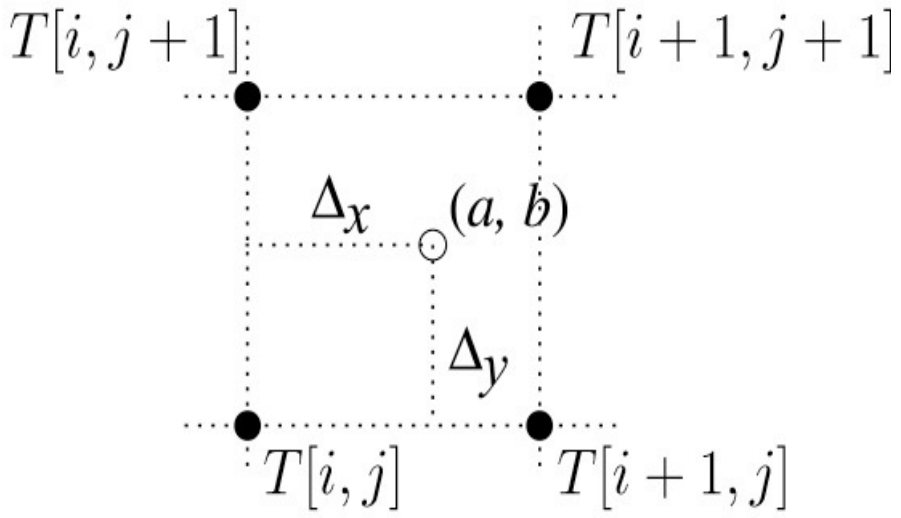
Dado um ponto da tela (pixel) queremos saber a que ponto do objeto ele corresponde (inverso),

e

dado um ponto do objeto, queremos saber a que ponto da textura ele corresponde (direto)

E finalmente re amostrada pelo uso de interpolação bi-linear quando necessário





$T(a,b) = T[i + \Delta x, j + \Delta y]$
$= (1- \Delta x)(1- \Delta y)\ T[i, j]+ \Delta x\ (1- \Delta y)\ T[i+1, j]$
$+(1- \Delta x)\ \Delta y\ T[i, j+1]+ \Delta x\ \Delta y\ T[i+1, j+1]$

Os mapas podem fazer mais que apenas mudar os tons

Podem mudar a **geometria da superfície** em que serão mapeados, por exemplo:

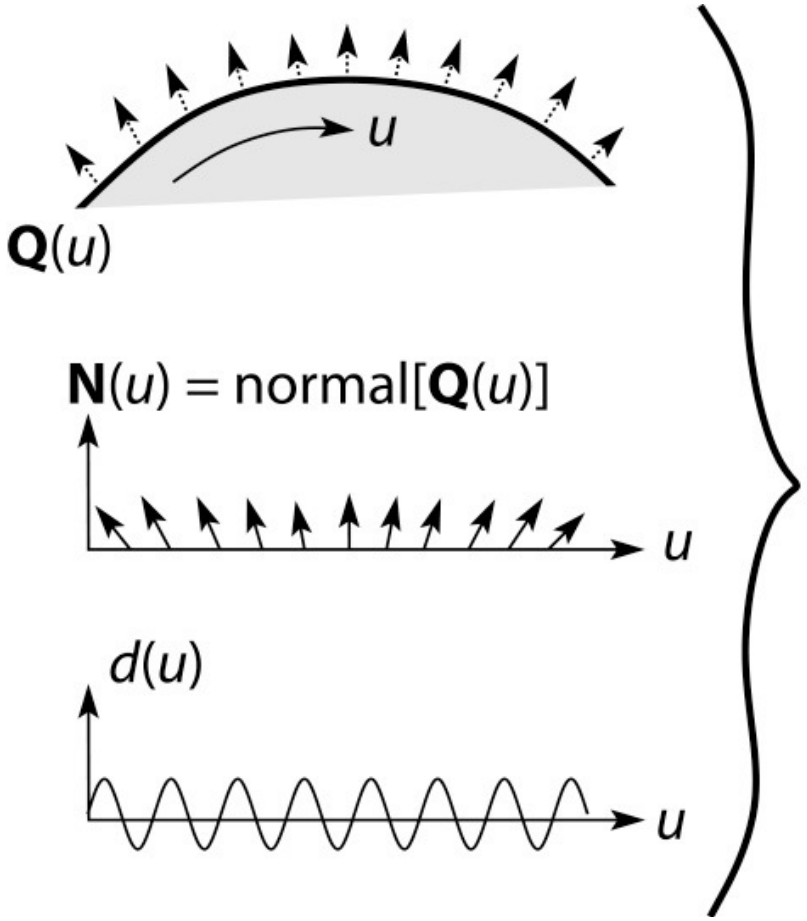
Se a superfície poder ser descrita como função de um parâmetro u: Q(u)

Sua normal, será geralmente também uma função: N(u)

Assim muda a **geometria da superfície** :

Displacement mapping (emboss)

Teremos assim uma nova superfície:



$$\tilde{\mathbf{Q}}(u) = \mathbf{Q}(u) + d(u)\mathbf{N}(u)$$

Bump map

Continuando com essa ideia pode-se pensar em modificar a normal da superfície (depois de fazer os tratamentos de hiddens , apenas na hora de produzir seu shading).

$$\tilde{\mathbf{N}} = \text{normal}[\tilde{\mathbf{Q}}(u)]$$

Mais de um mapeamento

Pode ser combinados, por exemplo:

cores e normais;

cores e deslocamentos;

Etc.

Multitexturing ocorre quando mais de um mapeamento é aplicado na face ao mesmo tempo.

Alterações na imagem final ao ser adicionada

Da fase de rendending, como:

Textura para cor difusa

textura do bump map

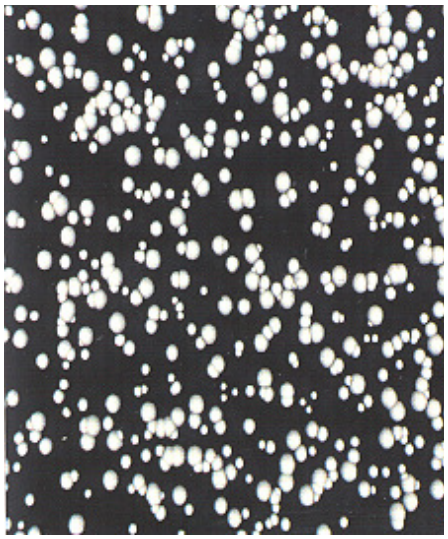
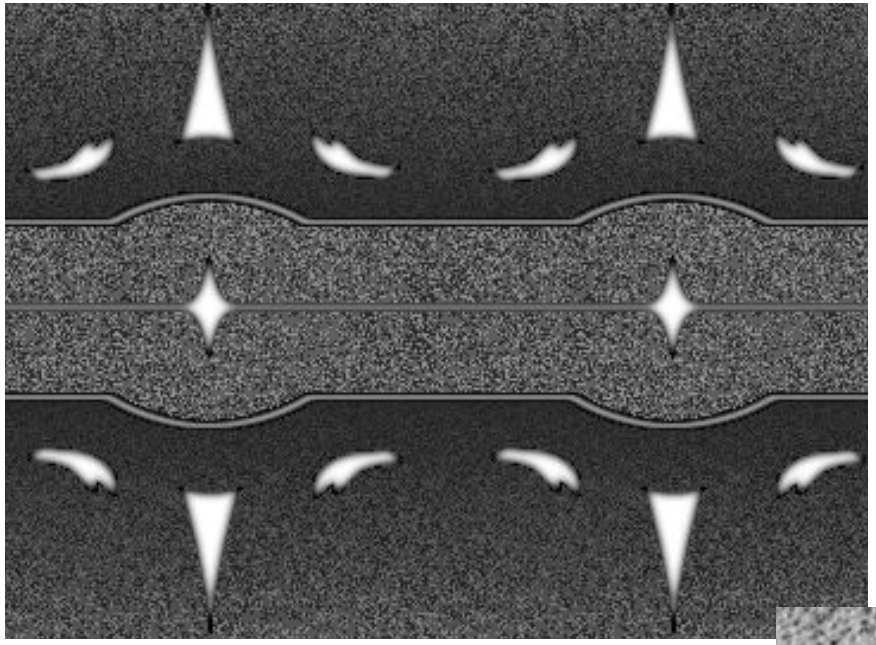


imagem final



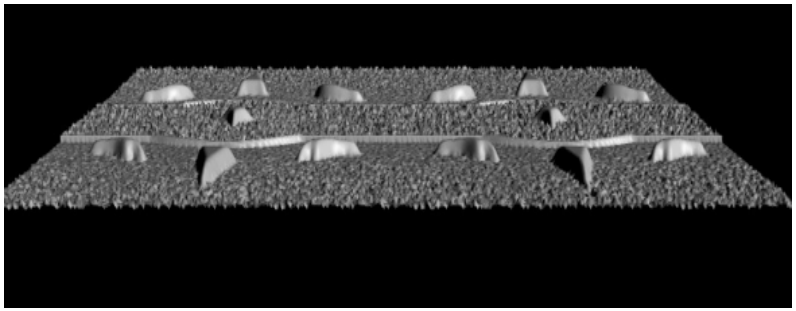
Mesma textura em fase plana

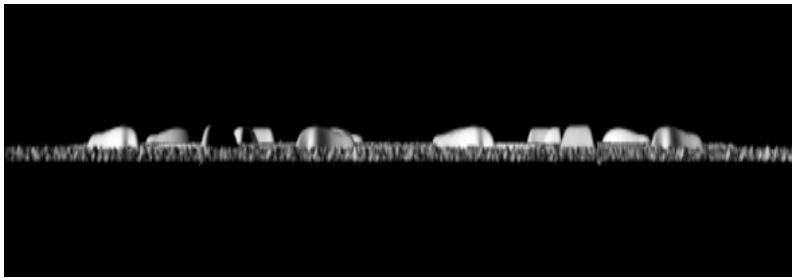
Como:

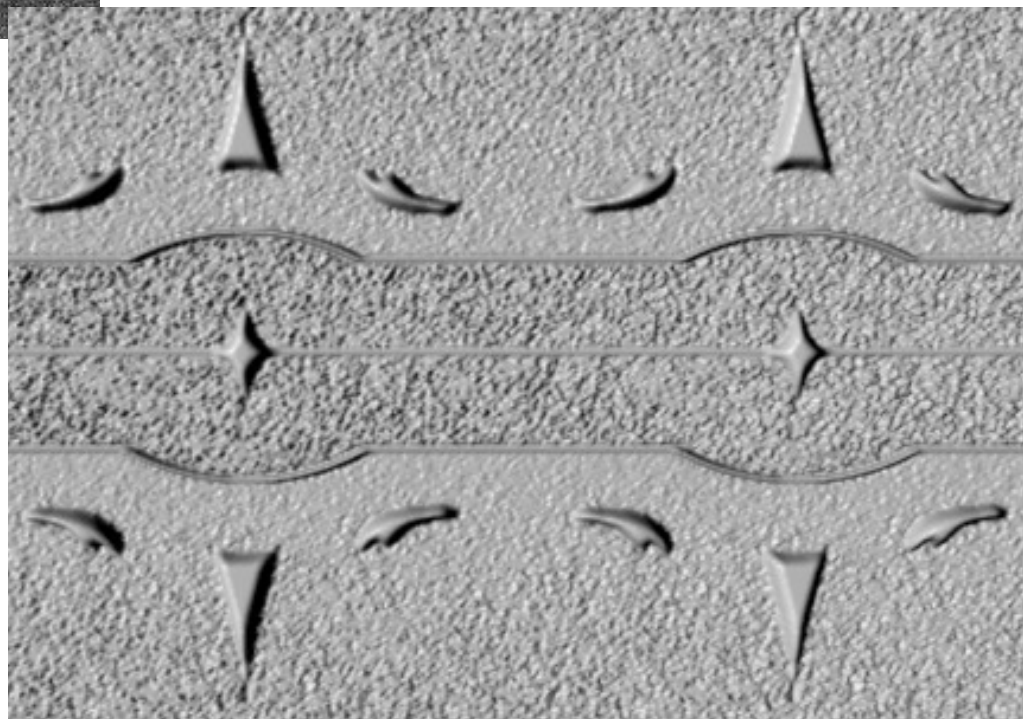
displacement map

+

bump map







Sobre uma fase cilíndrica

Displacement map

Displacement map + bumpmap

Environment ou reflection map

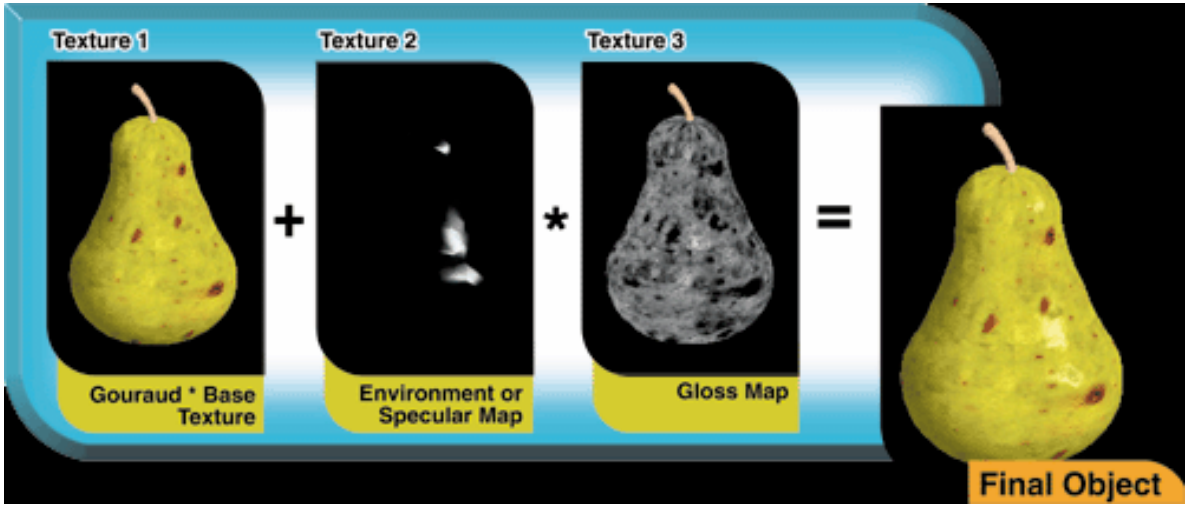
Usa para modelar o ambiente em uma superfície, como espelho.

Funciona bem com apenas um objeto na cena,

Dando uma ótima idéia de reflexão sem usar nenhum raio ou pode ser unido ao raytracing

Mapas podem ser combinados em diversos níveis

Para produzir um grau de realismo na cena de maneira simplificada



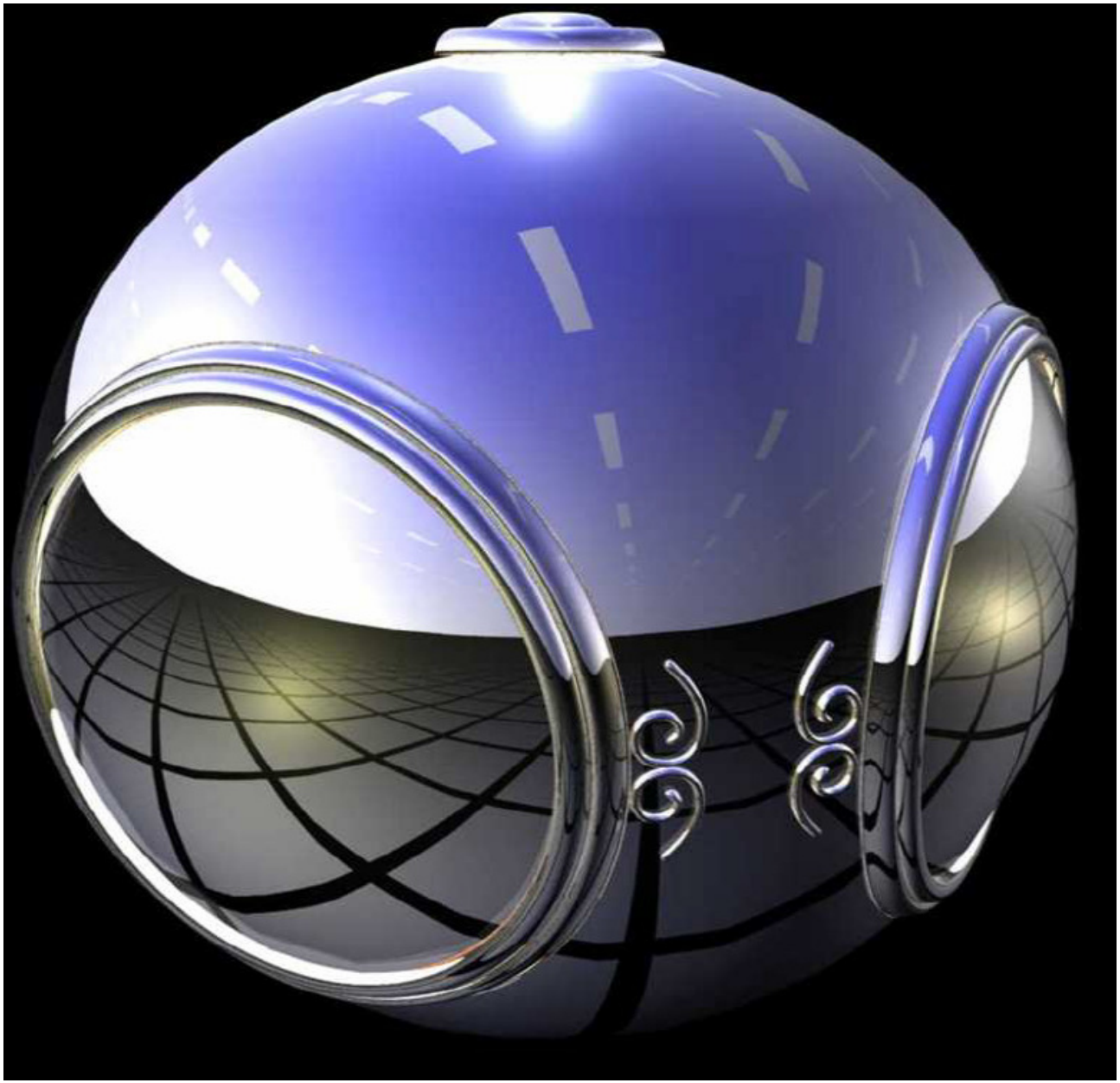
https://noppa.oulu.fi/noppa/kurssi/521493s/luennot/521493S_3-d_graphics_vi.pdf

Shading

Modelo Phong

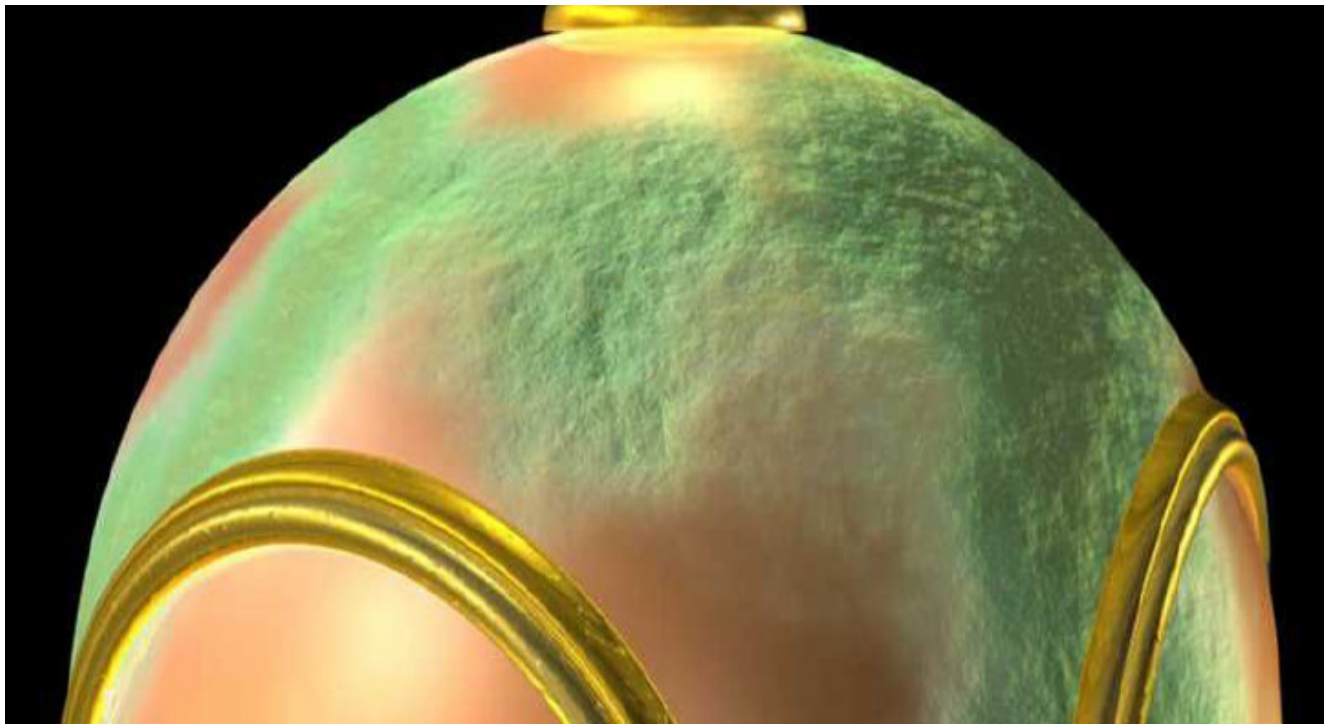
Texture map

Environment Mapping



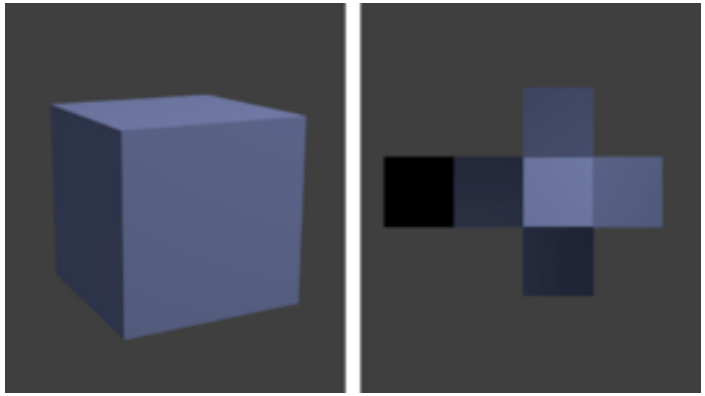
detalhes

Bump Mapping



Lightmap

Mapeamento que contém a intensidade luminosa das faces. Útil em objetos que permanecem estáticos em games. Geralmente flat , sem incluir a idéia da direção da iluminação . Presentes na maioria dos plug-ins 3D



http://www.computerhistory.org/

The Utah Teapot

Criado por Martin Newell na University of Utah em 1975.

Tem sido usado como modelo 3D por 40 anos para verificar modelos de iluminação , cor, realismo, etc.

(uma das estruturas de dados com malhas disponivel no OpenGL)

Procedural texture generation method

As texturas podem ser geradas por programação (procedures) e não apenas por captura de texturas já existentes

Geradores de padrões fractais são muitos úteis para isso!

Level of details (mip maps)

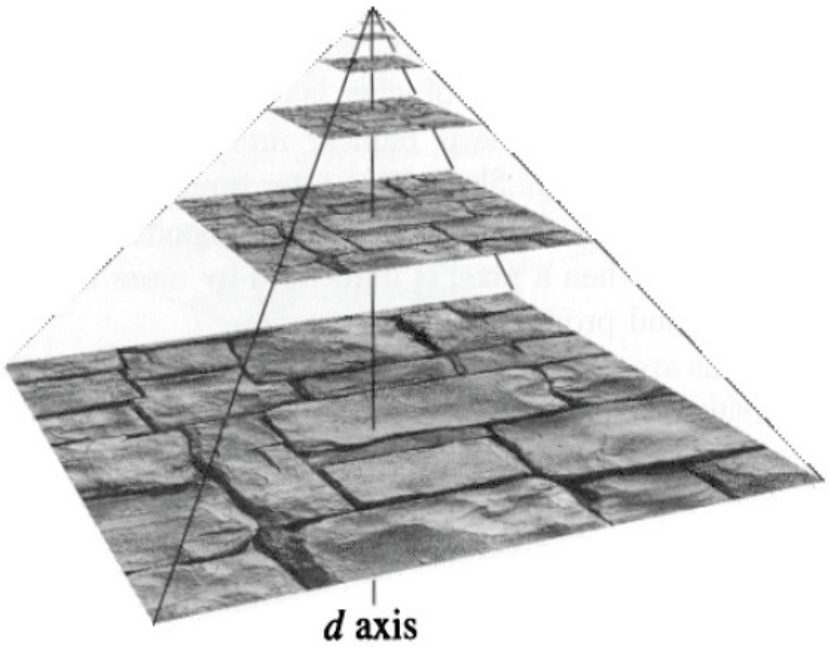
Alterado detalhes da textura com a distância ao observador

Também pode ser simulado com filtros

Que diminuem a resolução

"MIP" acronym of the phrase *multum in parvo* =

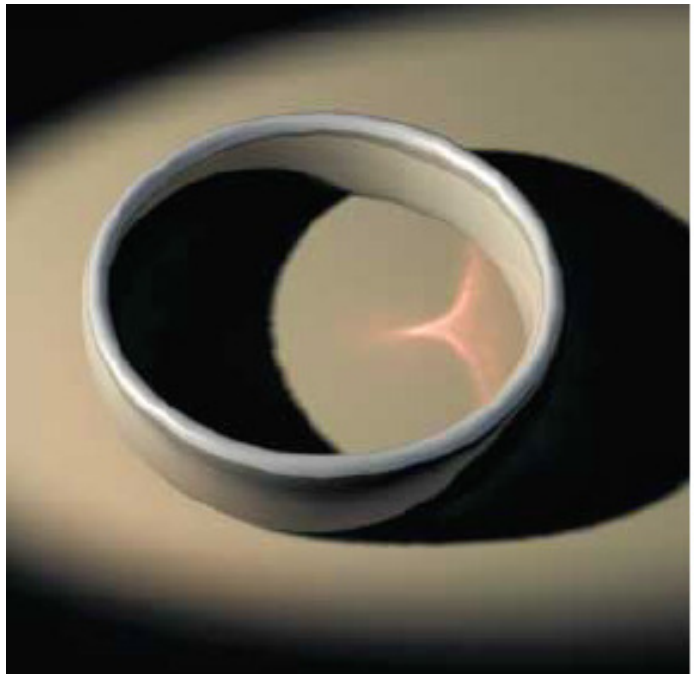
"much in little"

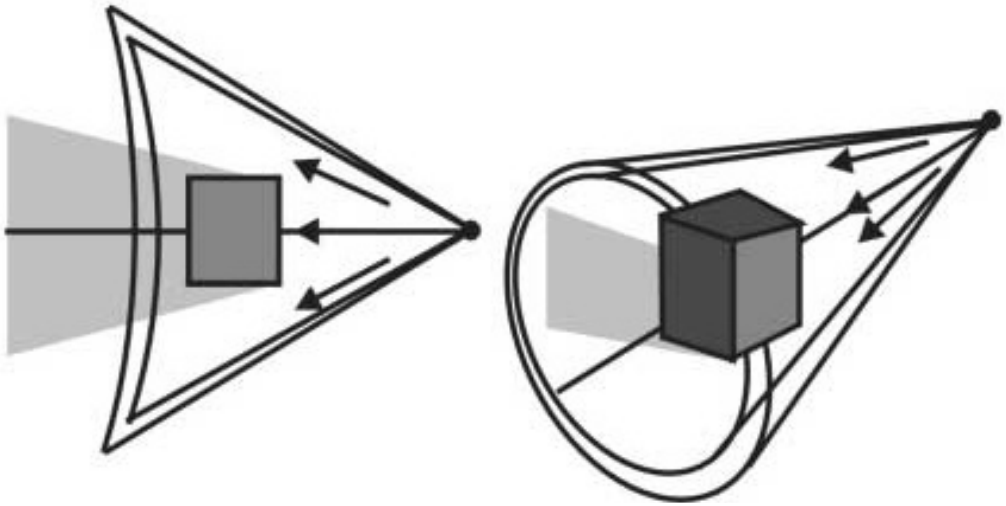


Sombras, refração, reflexão e efeitos específicos

Sombras,

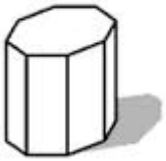
podem ser consideradas por diversos métodos, de simples projeções , passando por texturas até os métodos globais (seção 7.3.6 do livro texto tem boa revisão do assunto) !

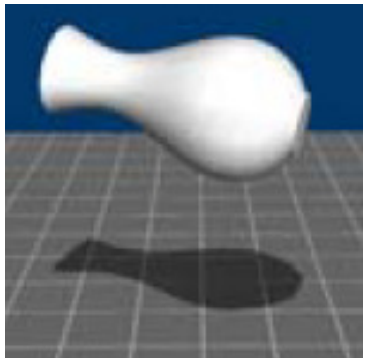




Sombras planas e projetadas:

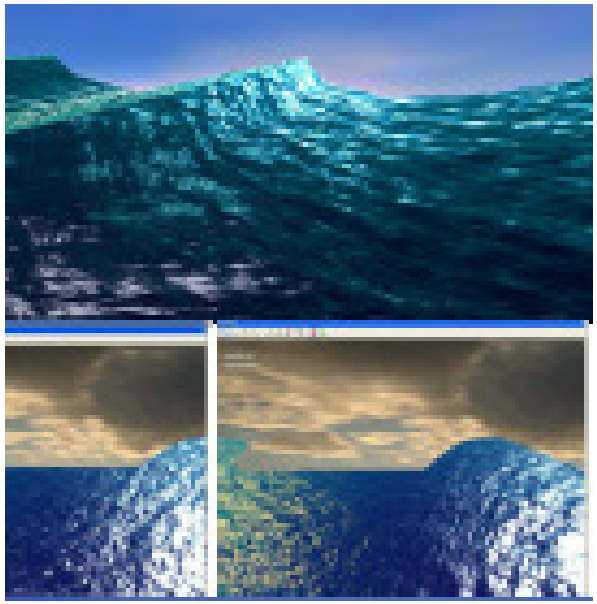
Sombra=umbra e penumbra





Cautics

São padrões de luz (refletidas e refratadas) que parecem concentrar a luz em alguns pontos. Ocorrem em vidros, água, modelagens de ondas, piscinas e outras situações que concentramos raios luminosos



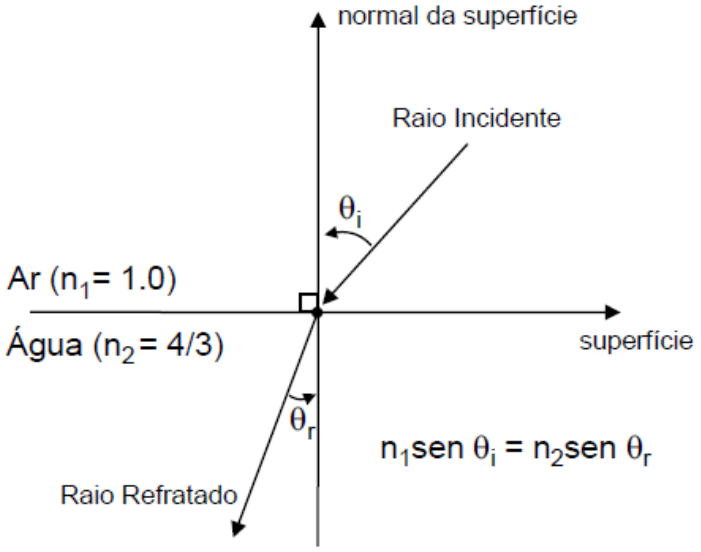
Refração

Quando o feixe de luz penetra em alguns materiais sua trajetória muda de ângulo de acordo com a diferença de densidade dos meios.

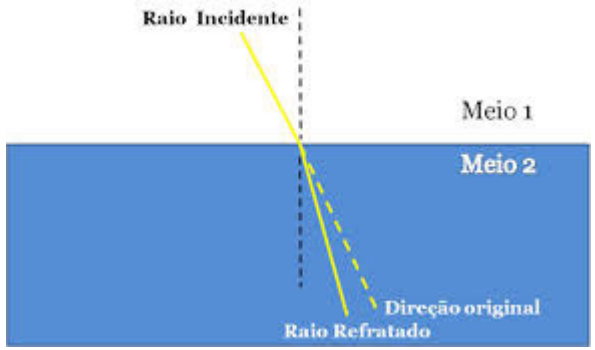
Lei da refração ou de Snell:

$$\frac{\text{sen}\,\theta_i}{\text{sen}\,\theta_r} = n_{21}$$

n_{21} é uma constante, chamada índice de refração ou IR



Exemplo de alguns IR:



Material	IR
Ar (em temperatura e pressão padrão ou STP)	1,0003
Água	1,33
Álcool etílico	1,36
Vidro	1,66
Plástico	1,51
Vidro Denso	1,52
Sal	1,53
Quartzo	1,46
Cristal	1,58
Diamante	2,42

Transparência

$$I = t\, I_1 + (1-t)\, I_2,\ 0 \leq t \leq 1$$

onde, I_1 é a superfície visível, I_2 é a superfície imediatamente atrás da superfície visível, e t é o fator de transparência para I_1. Se I_2 também é transparente, o algoritmo é aplicado recursivamente até encontrar uma superfície opaca ou o fundo da cena.

Modelos de iluminação globais

Ao contrario dos modelos locais que consideram a superfície a luz e o observador, os globais consideram todos os objetos da cena, precisam ter toda a base de dados dos objetos

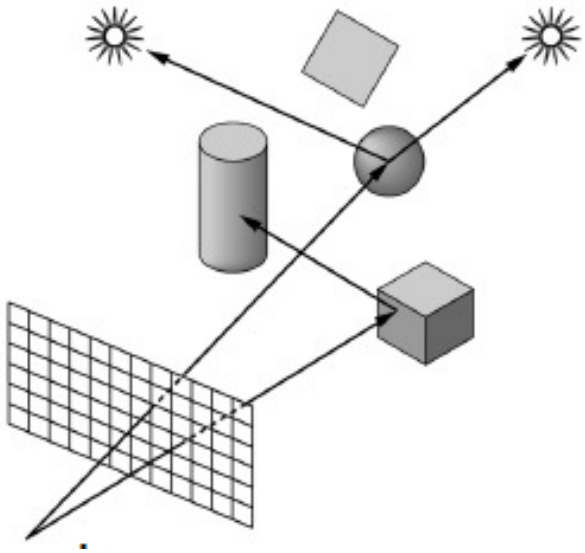
Principais: Raytracing e radiosidade

Não produzem os mesmo efeitos nem são adequados pra as mesmas coisas!

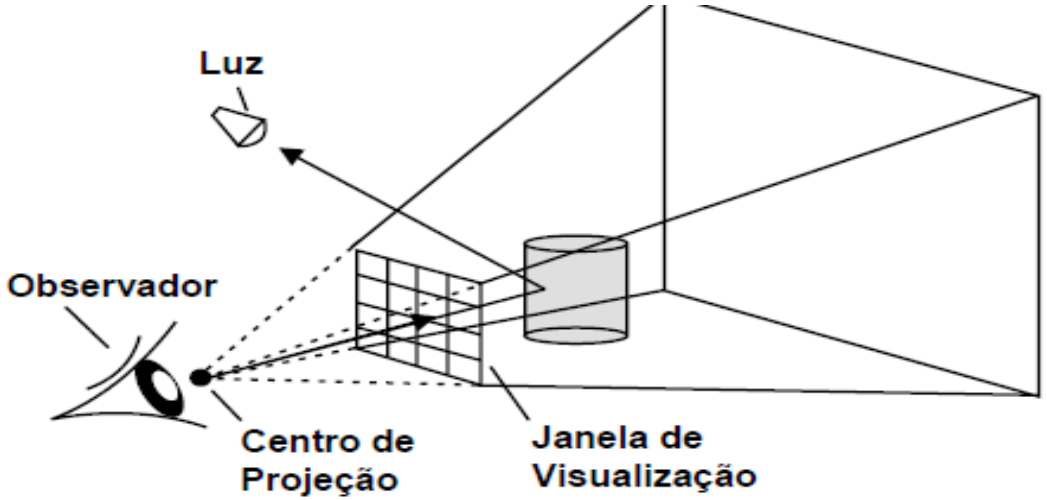
Lentos para real time!

Raytracing

Bom para:
reflexões,
transparências,
objetos fáceis de
calcular interseções
(superfícies,
planas, esférica,
E cilíndricas)



Ray tracing



é uma técnica para gerar uma imagem, seguindo o caminho da luz através de pixels em um plano de imagem e simulando os efeitos de seus encontros com objetos.

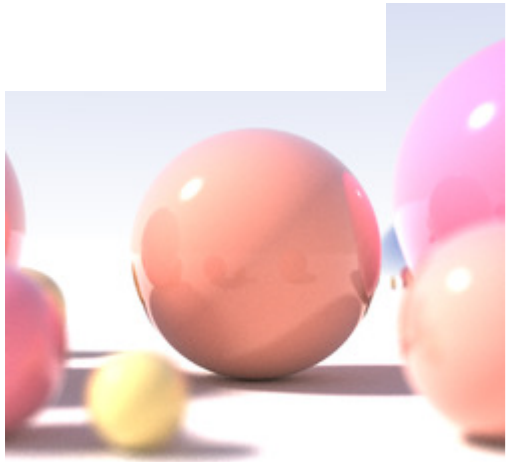
é capaz de produzir um elevado realismo visual, geralmente maior do que o dos métodos de processamento locais típicos, mas em um maior custo computacional.

Isso faz com ray tracing mais adequado para aplicações em que a imagem pode ser renderizada lentamente, como em imagens de cinema e televisão, efeitos visuais, e pouco adequada para aplicações em tempo real, como jogos, onde a velocidade é fundamental.

é capaz de simular uma variedade de efeitos ópticos, como os fenômenos de dispersão, reflexão e refracção.

O algoritmo de Ray tracing considera os seguintes pontos:

- Os raios são disparados de forma sistemática, de modo que cada um deles corresponda a um pixel na tela.
- Após o disparo, o raio percorre o espaço podendo atingir um objeto ou sair da cena.
- Se atingir algum objeto, o ponto de intersecção é calculado. As contribuições das fontes de luz para cada ponto, levando em conta a sombra de outros objetos, também são calculadas.
- Se o objeto for opaco, a soma dessas contribuições será a intensidade luminosa total naquele ponto.
- Caso contrário, as contribuições devidas aos reflexos e refrações, serão também computadas. O pixel correspondente pode, então, ser exibido.
- Se não houver interseção, o pixel terá a cor de fundo.



Algoritmo clássico

Para cada pixel da tela:

1. Trace um "raio" a partir do observador até a cena a ser representada através de um pixel da tela;
2. Determine qual o primeiro objeto a interceptar esse raio;
3. Calcule a cor ambiente da superfície do objeto no ponto de interseção baseado nas características do objeto e na luz ambiente;
4. Se a superfície do objeto for reflexiva, calcule um novo raio a partir do ponto de interseção e na "direção de reflexão";
5. Se a superfície do objeto for transparente, calcule um novo raio a partir do ponto de interseção.
6. Considere a cor de todos os objetos interceptados pelo raio até sair da cena ou atingir uma fonte de luz, e use esse valor para determinar a cor do pixel e se há sombras.

Espelhos:

O ray tracing deve considerar os raios refletidos toda vez que o coeficiente de reflexão de uma superfície for diferente de zero. O coeficiente de reflexão varia entre 0 e 1, determinando que quantidade de energia do raio de luz deve ser considerada como absorvida pelo objeto em questão, compondo uma soma ponderada das componentes de cor para o pixel na tela. Um espelho possui um coeficiente de reflexão próximo de 1, ou seja, nessa superfície todos os raios incidentes devem ser refletidos com o mesmo ângulo de incidência em relação à direção da reta normal à superfície. Além do coeficiente de reflexão, as superfícies também apresentam um coeficiente de refração que expressa a maneira pela qual a luz passa através de um meio para outro.

Cálculo de interseções:

A tarefa principal do ray tracing consiste no cálculo da interseção de um raio com o objeto. Para essa tarefa, utiliza-se normalmente a representação paramétrica de um vetor ou reta. Cada ponto (x, y, z) ao longo de um raio com origem no ponto (x_0, y_0, z_0) e direção do ponto (x_0, y_0, z_0) para o ponto (x_1, y_1, z_1) é definido em função do parâmetro t, (com valores no intervalo [0,1] pelas equações paramétricas da reta):

$x = x_0 + t(x_1 - x_0);$ $\quad x = x_0 + t\Delta x;\ \Delta x = x_1 - x_0$

$y = y_0 + t(y_1 - y_0);$ $\quad y = y_0 + t\Delta y;\ \Delta y = y_1 - y_0$

$z = z_0 + t(z_1 - z_0);$ $\quad z = z_0 + t\Delta z;\ \Delta z = z_1 - z_0$

(x_0, y_0, z_0) for considerado o centro de projeção, ou o olho do observador

(x_1, y_1, z_1) for o centro de um pixel na "janela"

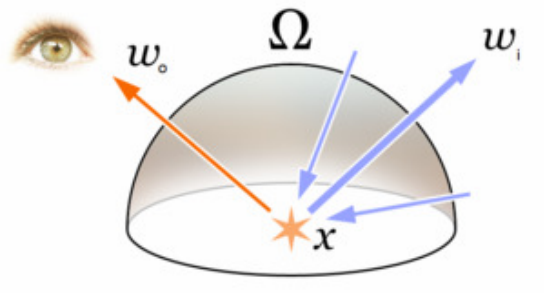
t varia de 0 a 1 entre esses pontos.
valores de t maiores que 1 correspondem a pontos depois da janela

Radiosidade

considera a solução da integral de rendering (equilíbrio da radiância em um ponto ou a conservação da energia) para modelar a iluminação.

O nível de realismo da modelagem é muito maior.

Considera a função bidirecional de distribuição da reflectancia- **bidirectional reflectance distribution function (BRDF)**.

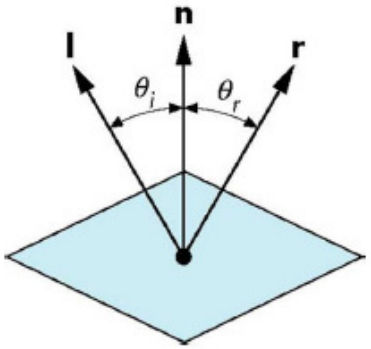


Os anteriores todos consideram

Que os 3 vetores estão no mesmo plano

(reflexão ideal)

$$\mathbf{r} = 2\,(\mathbf{l} \cdot \mathbf{n})\,\mathbf{n} - \mathbf{l}$$



Radiosidade é:

- uma aplicação do método de elementos finitos para resolver a equação de renderização para cenas com superfícies que refletem a luz de forma difusa.

-um algoritmo de iluminação global: a iluminação não vem apenas a partir das fontes de luz, mas todas as superfícies de cena interagindo uns com os outros.

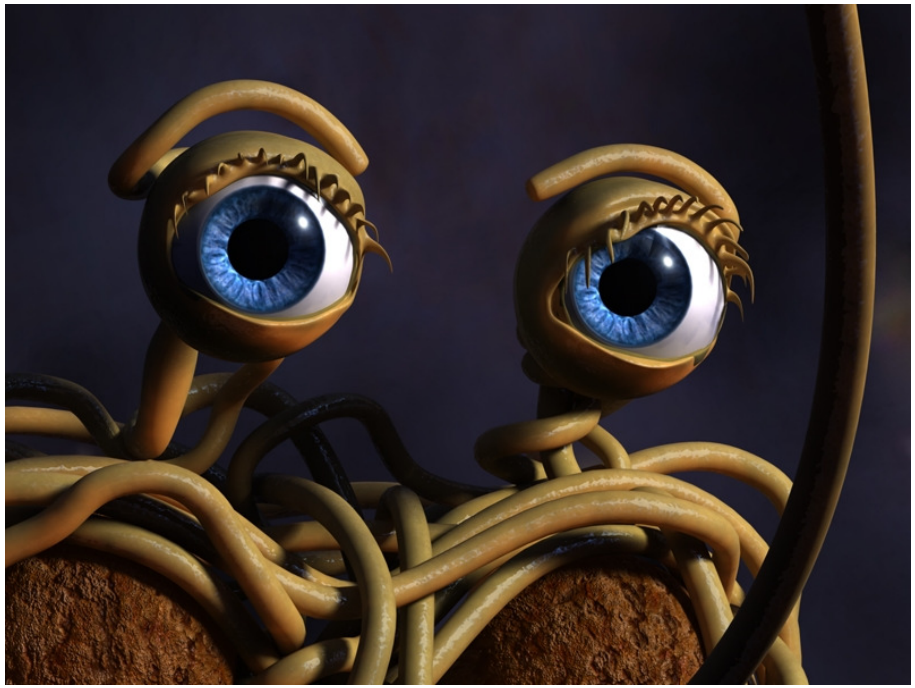
-**independente do ponto de vista**, o que aumenta o volume dos cálculos envolvidos, mas torna-os **úteis para todos os pontos de vista.**

- inicialmente uma aplicação desenvolvidos na área de transferência de calor, posteriormente adaptada para a aplicação de computação gráfica (1984 na Universidade de Cornell).

Color bleeding

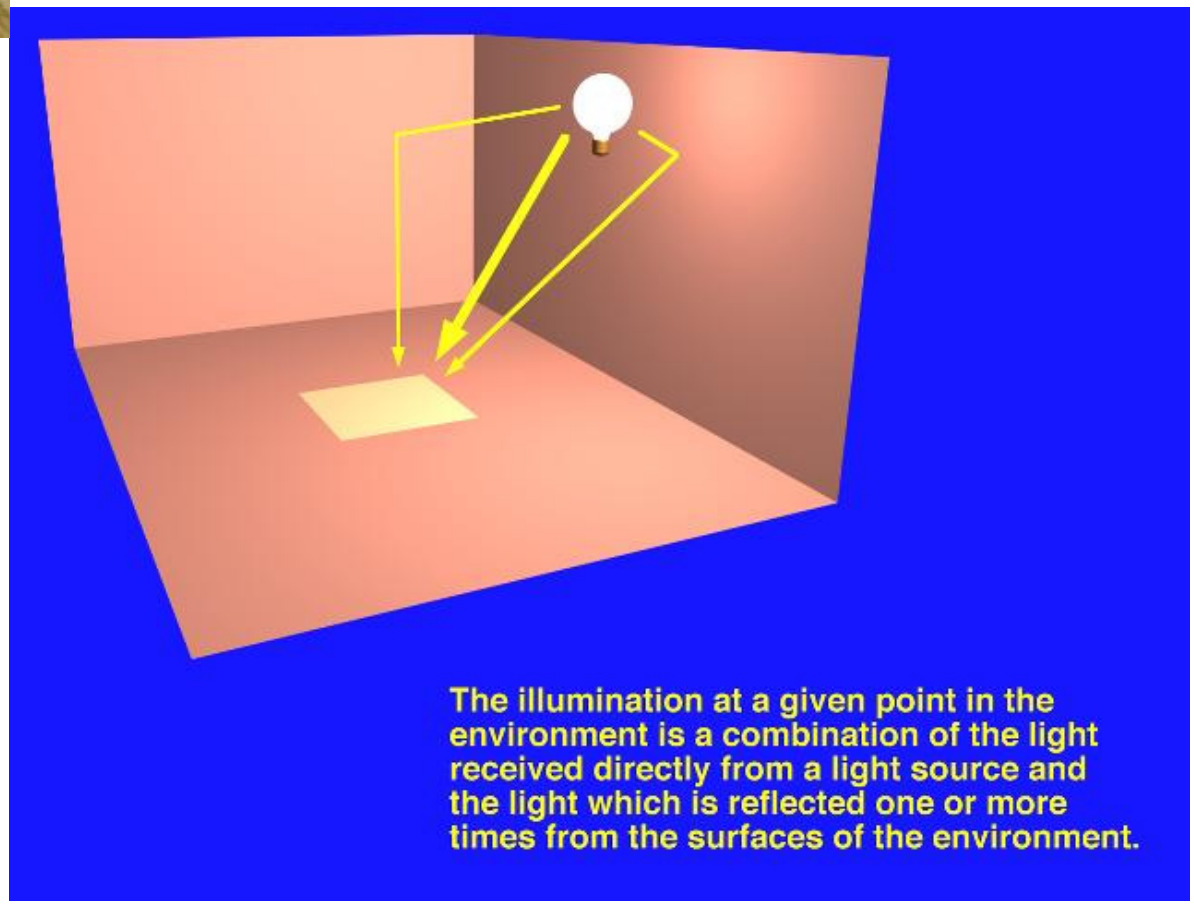
Em rendering , **color bleeding** é a ocorrência de colorização de um objeto ou superfície pela cor refletida de superfícies próximas.

Ocorre principalmente quando se usa Radiosity para a cena 3D.

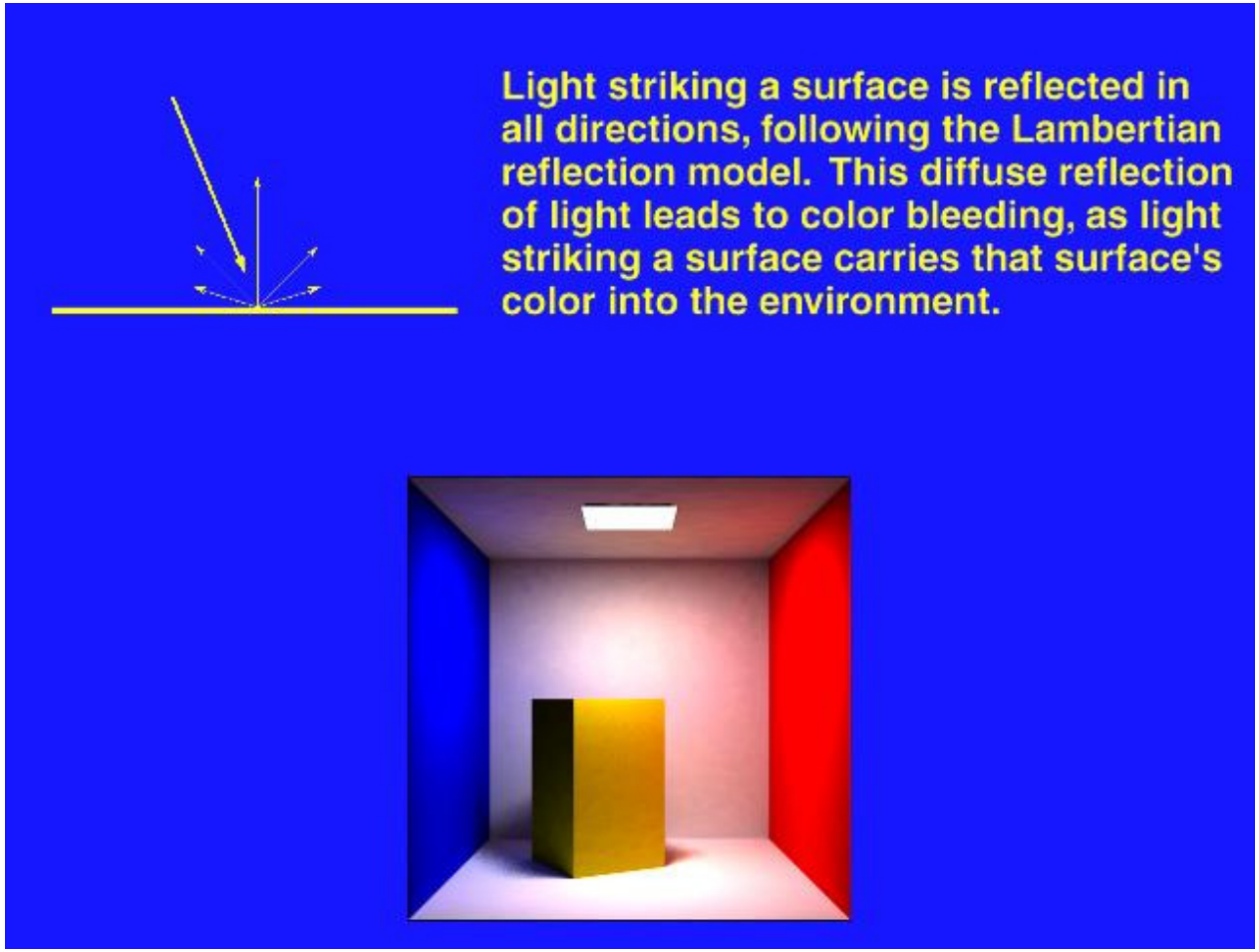


Radiosidade: discretiza o ambiente em um malha

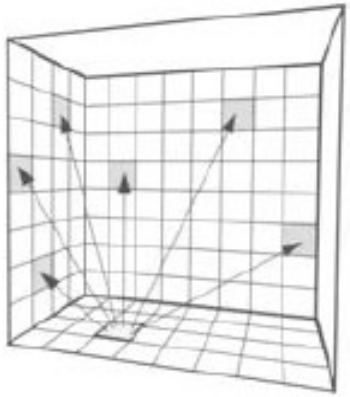
Os limites da malha devem coincidir com os limites das zonas de diferença de iluminação



Balanço ou equilíbrio de energia radiante



Radiosidade:



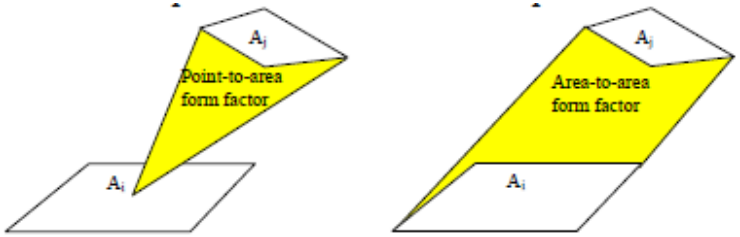
O método da radiosidade é baseado em um modelo simples de balanço de energia. Na sua origem, o cálculo da radiosidade empregado em Transmissão de Calor não é mais do que a aplicação da lei da conservação da energia a cada uma das superfícies de um recinto ou cena, e pressupõe a existência de equilíbrio térmico. Em cada superfície de um modelo, a quantidade de energia emitida é a soma entre a energia que a superfície emite internamente mais a quantidade de energia refletida. A quantidade de energia refletida pode ser caracterizada pelo produto entre a quantidade de energia incidente na superfície e a constante de reflexão da superfície.

$$B_j = \rho_j H_j + E_j$$

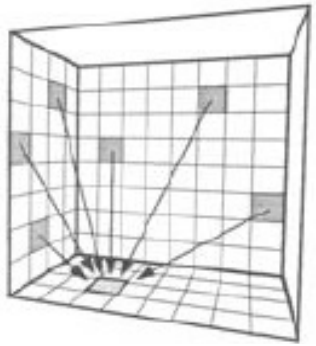
onde B_j é a radiosidade da superfície j, ρ_j sua reflectividade, H_j a energia incidente nesta superfície e E_j a energia emitida pela superfície j

Radiosidade

A radiosidade de uma superfície é a energia dissipada. Isso é usado para determinar a intensidade luminosa da superfície. A quantidade de energia emitida por uma superfície deve ser especificada como um parâmetro do modelo, como nos métodos tradicionais onde a localização e a intensidade das fontes de luz devem ser especificadas. A reflectividade da superfície também deve ser especificada no modelo, como nos métodos de iluminação tradicional. A única incógnita da equação é a quantidade de luz incidente na superfície. Esta pode ser encontrado somando-se todas as outras superfícies à quantidade de energia refletida que contribui com a iluminação dessa superfície:



$$H_j = \sum_{i=1}^{n} B_j F_{ij}$$



onde H_j é a energia incidente na superfície j, B_j a radiosidade de cada superfície i da cena e F_{ij} uma constante i j.

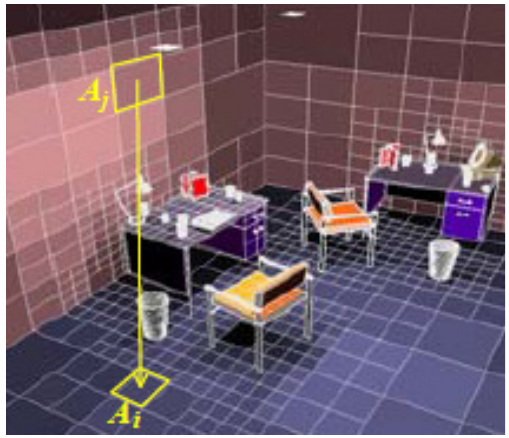
A constante dessa equação é definida como a fração de energia que sai da superfície i e chega na superfície j, e é, portanto, um número entre 0 e 1. Essa constante pode ser calculada por métodos analíticos, ou através de semelhança geométrica.

A equação da radiosidade fica assim:

$$B_j = E_j + \rho_j \sum_{i=1}^{n} B_j F_{ij}$$

A consideração de todas as superfícies da cena forma uma seqüência de N equações lineares com N incógnitas, o que leva a uma solução matricial:

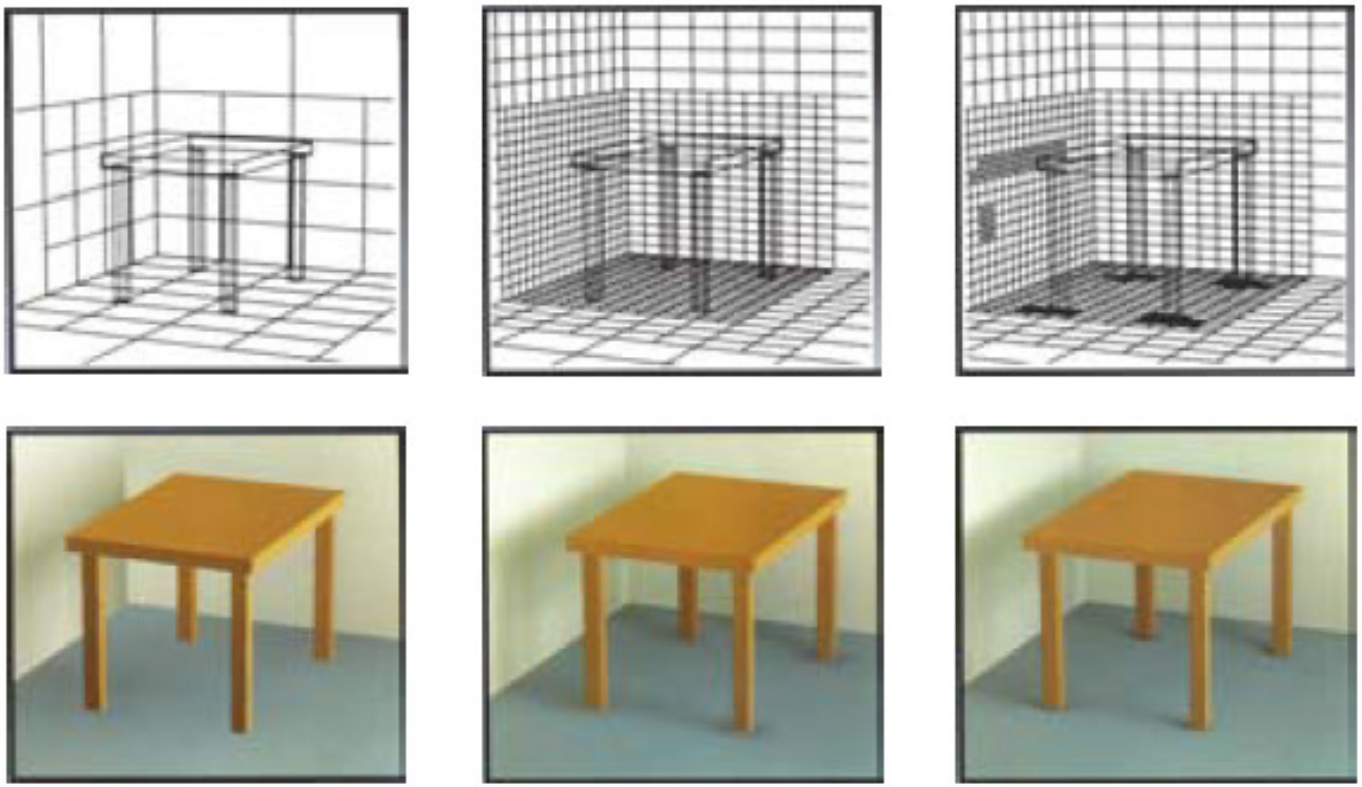
$$\begin{bmatrix} 1-\rho_1 F_{11} & 1-\rho_1 F_{12} & \Lambda & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_1 F_{22} & \Lambda & -\rho_2 F_{2n} \\ M & M & O & M \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \Lambda & -\rho_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ M \\ B_3 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ M \\ E_3 \end{bmatrix}$$



Essa matriz tem duas propriedades interessantes: é diagonalmente dominante e, portanto, converge quando usada no método iterativo de Gauss Seidel, [GOR, 84]. Métodos alternativos para o cálculo dessa matriz já foram propostos por Cohen et al. [COH, 88]. Alguns permitem uma convergência para a solução correta mais rápida que o algoritmo iterativo de Gauss Seidel.

Refinamentos progressivos

Alterando o numero de elementos da malha:



Coarse patch solution
(145 patches)

Improved solution
(1021 subpatches)

Adaptive subdivision
(1306 subpatches)

Sombreamento anisotrópico

Isotrópico x ortotrópico

Photon mapping

Algoritmo de **iluminação global** em 2 passadas (two-pass) que considera modelos de radiância para maior realismo na simulação da refração e reflexão da luz em superfícies transparentes

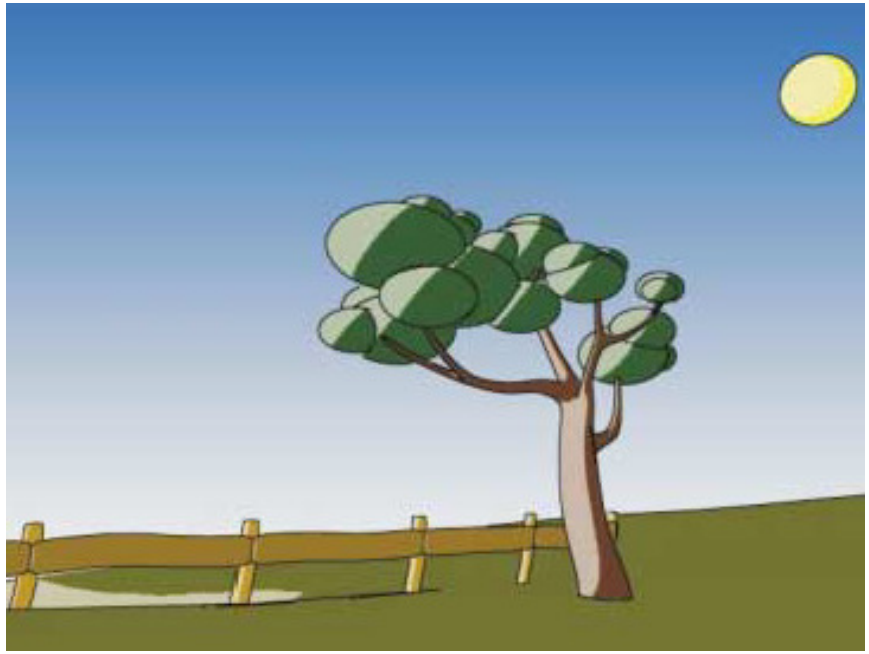
É capaz de simular a refração da luz em meios transparentes tal como o vidro ou a água, inter reflexões difusas entre objetos iluminados, a dispersão da luz sob a superfícies de materiais translúcidos, e efeitos causados por partículas, tal como o **fumaça ou a água de vapor.**

Há muito mais do que isso!

Vimos aqui apenas sobre um realismo fotográfico das imagens, mas há diversas outras formas e esse assunto esta sempre em constante evolução. Assim depois desta leve introdução continue na área! Você já tem a bagagem teórica que precisa para agora descobrir o resto sozinho!

Toon Shading

Stylistic rendering



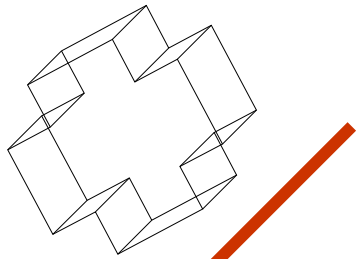
Trabalho 2 – parte 1

Rodar a figura 3D do seu grupo, em wire-frame, em torno de um ângulo e eixo qualquer que o usuário vai definir, logo depois de você mostrar a figura a ele em 3D. A definição e eixo será feita através do fornecimento das coordenadas 3D de 2 pontos deste eixo e do ângulo em graus que a figura será girada.

Depois faça a rotação parecer uma animação apagando e redesenhando o objeto em uma nova posição girada do ângulo dados pelo usuário dividido por 50 incrementalmente.

Deixe sempre visível o eixo definido pelo usuário desenhado na mesma projeção do seu objeto.

De quantos graus voce quer girar?
250



B (30,20,-10)

A (-20, -40, 15)

Trabalho 2 – parte 2

INCLUIR a opção de transladar a figura 3D do seu grupo, em wire-frame, em uma trajetória curva definida pelo usuário. Logo depois de você mostrar a figura a ele em 3D. Aparecem uma opção para translação em curva. A definição da curva será feita através do fornecimento das coordenadas 3D de 4 pontos de controle e deve ser desenhada na tela a trajetória logo depois para o usuário aprovar.

Depois faça a translação parecer uma animação apagando e redesenhando o objeto em uma nova posição sobre a curva dadas pelo usuário dividida por 100 incrementos.

Deixe sempre visível a curva definida pelo usuário desenhando-a mesma projeção do seu objeto.

Defina sua curva por 4 pontos.

I(-20, -40, 15)

(-10, 40, 5)

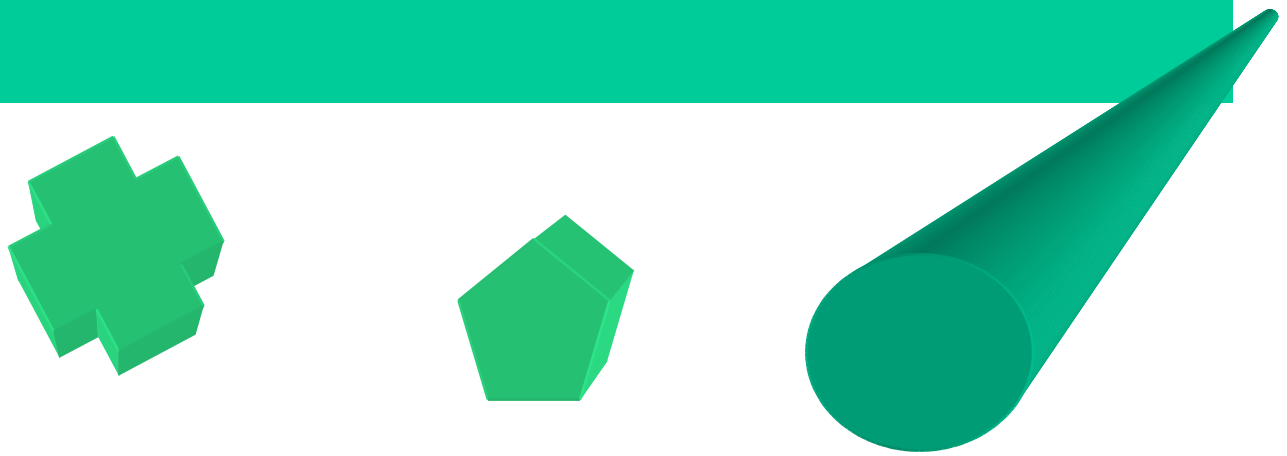
(10,-20, 0)

F (30,20,-10)

Trabalho 2 – parte 3

Ao final de qualquer das opções escolhidas pelo usuario:

1- desenhar seu objeto na cor : 50 , 250, 150 em RGB com um shading que voce considere adequado.



Trabalho 2 – parte 2

INCLUIR a opção de transladar a figura 3D do seu grupo, em wire-frame, em uma trajetória curva definida pelo usuário. Logo depois de você mostrar a figura a ele em 3D. Aparecem uma opção para translação em curva. A definição da curva será feita através do fornecimento das coordenadas 3D de 4 pontos de controle e deve ser desenhada na tela a trajetória logo depois para o usuário aprovar.

Depois faça a translação parecer uma animação apagando e redesenhando o objeto em uma nova posição sobre a curva dadas pelo usuário dividida por 100 incrementos.

Deixe sempre visível a curva definida pelo usuário desenhando-a mesma projeção do seu objeto.

Defina sua curva por 4 pontos.

I(-20, -40, 15)

(-10, 40, 5)

(10,-20, 0)

F (30,20,-10)

Bibliografia:

E. Azevedo, A. Conci, Computação Gráfica: teoria e prática, Campus ; - Rio de Janeiro, 2003

J.D.Foley,A.van Dam,S.K.Feiner,J.F.Hughes. Computer Graphics- Principles and Practice, Addison-Wesley, Reading, 1990.

H. Watt, F. Policarpo - The Computer , Addison-Wesley Pub Co (Net); 1998

http://en.wikipedia.org/wiki/Shadow_mapping

https://noppa.oulu.fi/noppa/kurssi/521493s/luennot/521493S_3-d_graphics_vi.pdf

http://graphics.stanford.edu/papers/rad/