Algoritmos para Cálculo de Sombras

Marcelo Walter - UFPE

Revisto Maio/2009

Quão Complexo é o Universo? Muito :-) e em termos de CG?

Fotos de Eneida Serrano Interiores - RS

Sumário

- Motivação
- Algoritmos para sombras "*duras*"
- Algoritmos para sombras "*moles*"

©Walt Disney

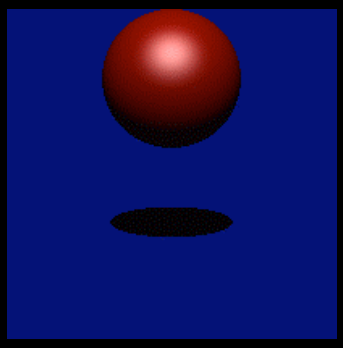
Será que está correta? Importa?

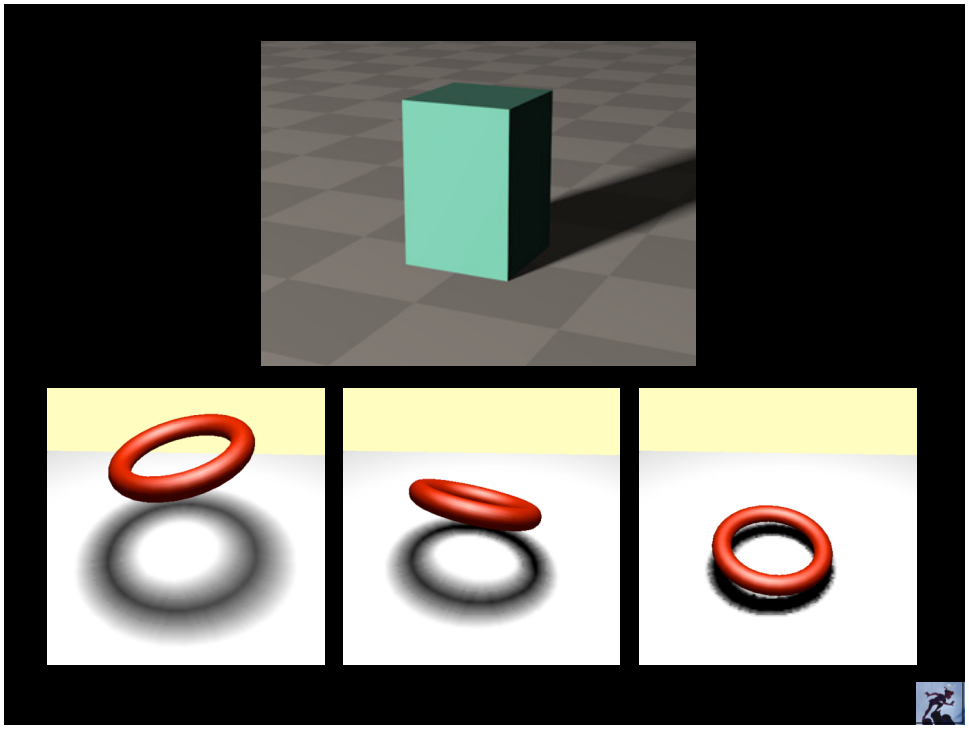
Motivação

- Sombras contribuem significativamente para aumento do realismo de cenas
- Informação visual sobre o posicionamento relativo dos objetos
- Efeito global que é caro de ser calculado
- Quais soluções são adequadas para aplicações interativas como jogos?

Sombras: Algoritmos Globais x Locais

- Para algoritmos globais a sombra já faz parte da solução
- Veremos aqui como acrescentar sombras aos algoritmos locais

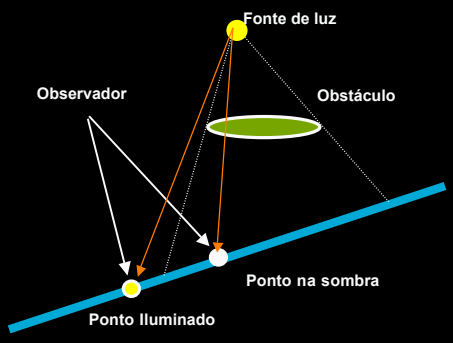




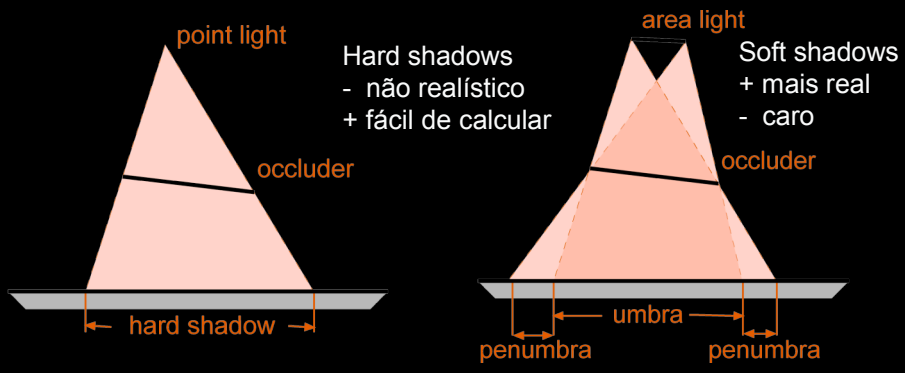

N
NE

To be or not to be a Shadow...

- Um ponto de uma superfície está na sombra, em relação a uma fonte de luz, se há um obstáculo entre esta luz e o ponto

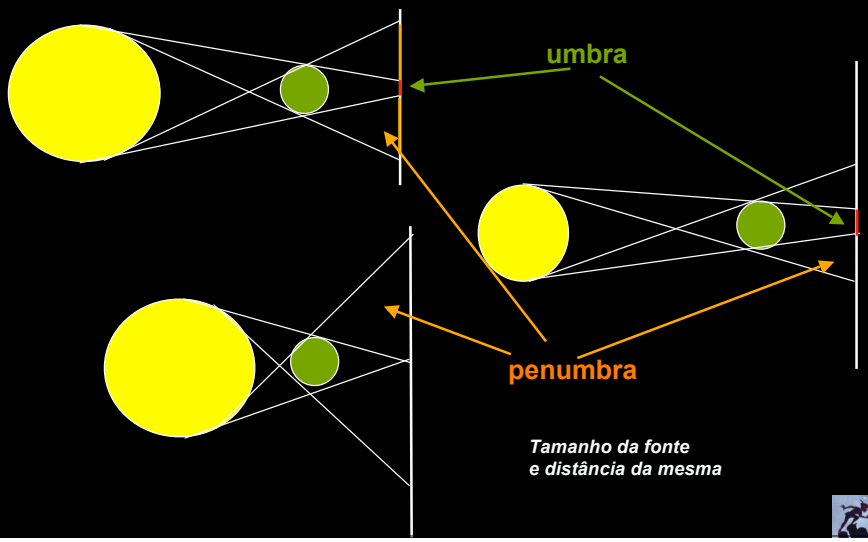


Sombra "dura" versus sombra "mole"



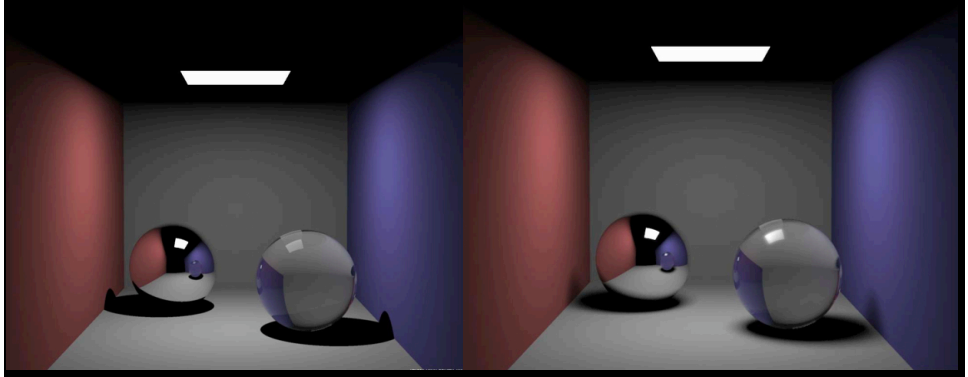
Umbra: ponto totalmente bloqueado
Penumbra: parcialmente bloqueado

Variações em função da geometria



Tamanho da fonte
e distância da mesma

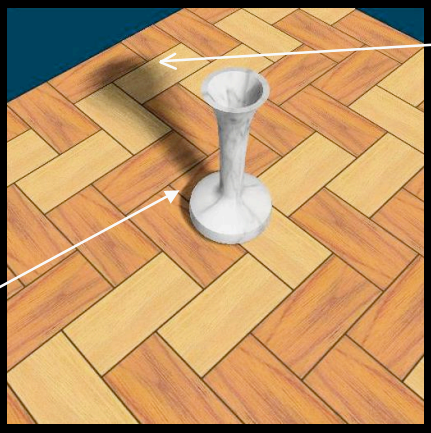
Exemplo



hard

soft

Mas, uma sombra mole não é uma sombra dura "*esfumaçada*"??



Algoritmos para *Hard Shadows*

- Sombras aproximadas
- Sombras projetadas [Blinn88]
- Shadow Textures
- Volumes de sombra [Crow77]
- *Shadow maps* [Williams78]

Sombras Aproximadas

- Geometria simples, acrescenta um polígono na cena

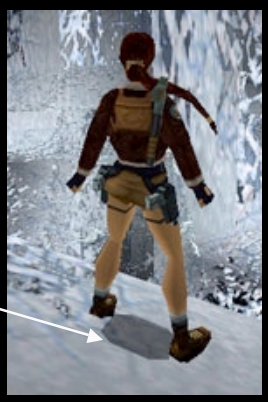


Image from TombRaider. ©Eidos Interactive.

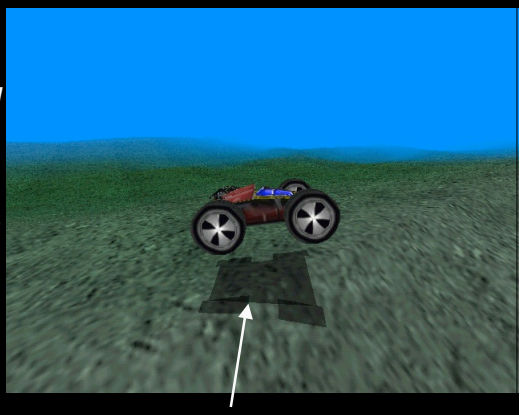
- Sem efeito global
  …mas melhor do que não ter nenhuma sombra ☺

Sombras Aproximadas

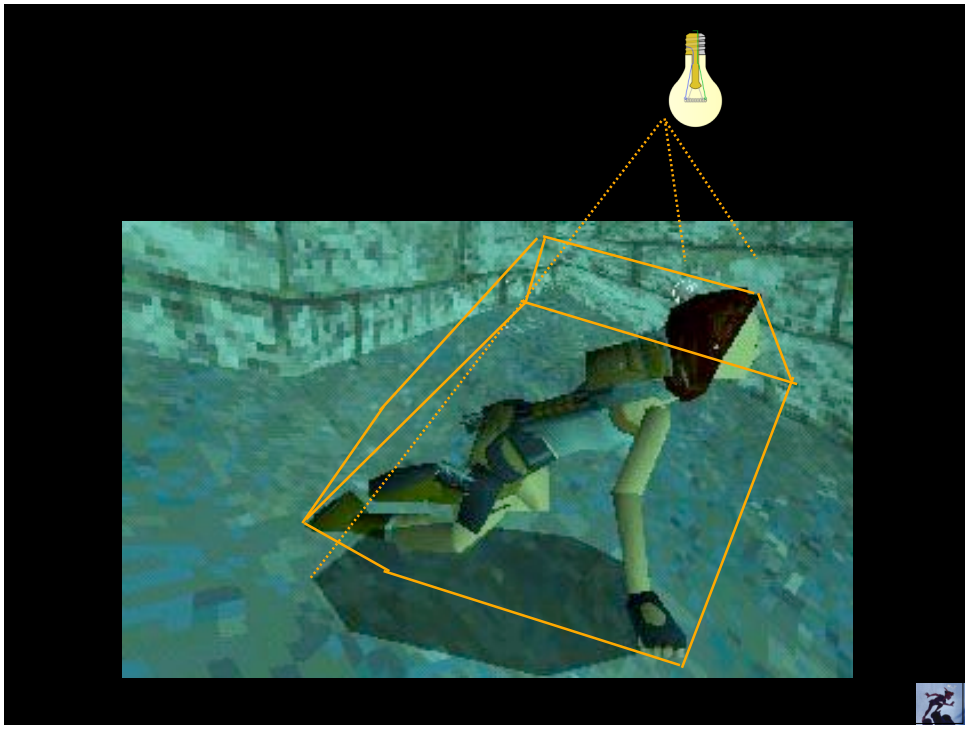
- Sombra tem forma e tamanho constante (normalmente um círculo com as bordas “borradas”)
- Para encontrar o polígono que recebe a sombra, disparar um raio que vai da fonte de luz e passa pelo centróide do objeto

Geometria Projetada (melhorando um pouquinho...)

- Englobar o objeto em um (ou mais) AABB (*axis aligned bounding box*)
- Projetar os vértices do AABB

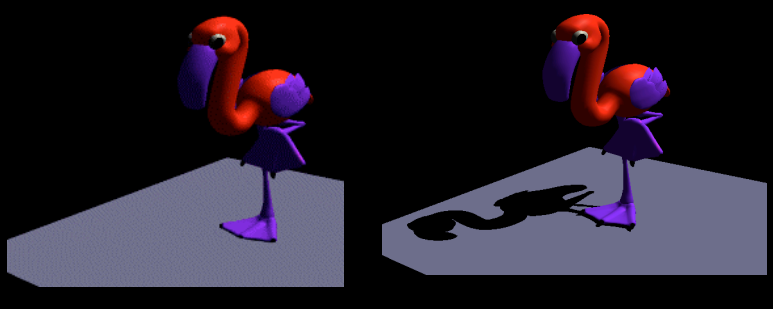


5 bounding boxes

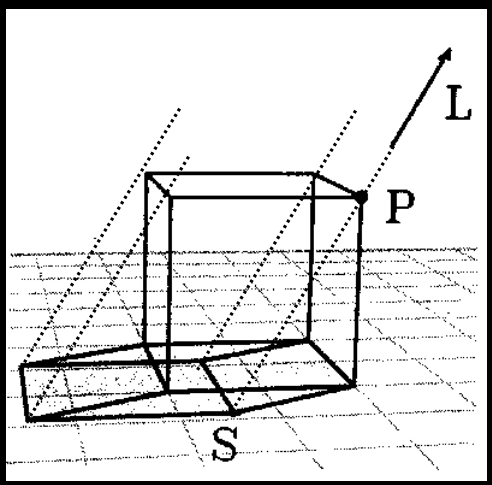


Geometria Projetada (melhorando ainda mais...)

- [Blinn88] *Me and my fake shadow*
  - Sombras para polígonos grandes (plano do chão, paredes,...)



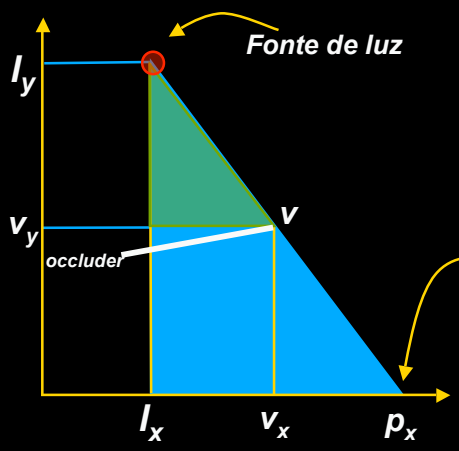
- Projetar cada vértice da geometria contra um plano (parede ou chão)



Luz Direcional

Exemplo: fonte pontual

- Plano *xz* com *y=0*



$$\frac{l_y - v_y}{v_x - l_x} = \frac{v_y}{p_x - v_x}$$

$$p_x = \frac{l_y v_x - l_x v_y}{l_y - v_y}$$

- Transformação como matriz 4 x 4

$$\begin{pmatrix} p_x \\ p_y \\ p_z \\ p_w \end{pmatrix} = \begin{pmatrix} l_y & -l_x & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -l_z & l_y & 0 \\ 0 & -1 & 0 & l_y \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 1 \end{pmatrix}$$

- Seja um plano E genérico dado por

Normal ao plano

$$\vec{n} \bullet \vec{x} + d = 0$$

- Os vértices v projetados no plano E acima são calculados por:

Fonte de luz

$$\vec{p} = \vec{l} - \frac{d + \vec{n} \bullet \vec{l}}{\vec{n} \bullet \left(\vec{v} - \vec{l}\right)} (\vec{v} - \vec{l})$$

- Expressando a eq anterior como matriz temos

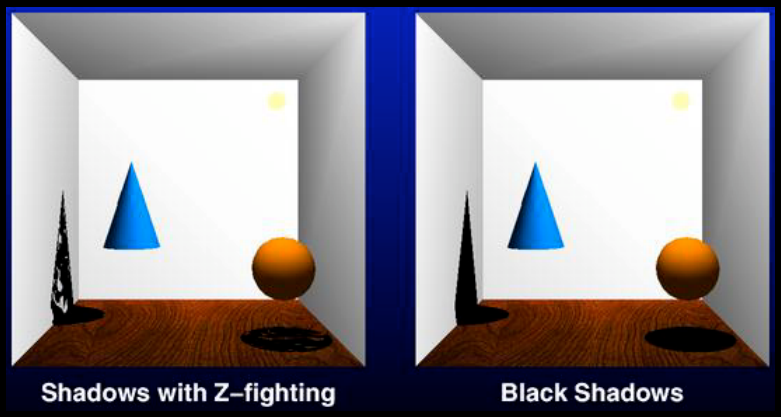
$$\begin{pmatrix} n \cdot l + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & n \cdot l + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & n \cdot l + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & n \cdot l \end{pmatrix}$$

Página 252
RealTime Rendering

Geometria Projetada Algoritmo Básico

- Renderiza a cena normalmente
- Para cada polígono *receiver*
  - Calcula a matriz de projeção *M*
  - Multiplica a matriz atual por *M*
  - Renderiza o polígono (ou mais escuro ou preto)

Problemas?



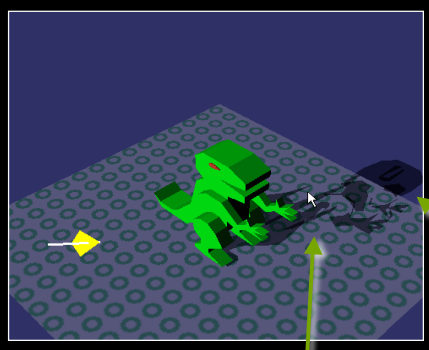
- Z-fighting

Problemas?

- Z-Fighting
  - Utilizar *bias* quando renderizar o polígono de sombra (`glPolygonOffset`)
  - Desenhar o polígono de sombra com o teste de zbuffer desligado (após o desenho do plano do chão)

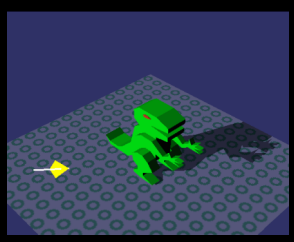
Geometria Projetada

Bad



Z fighting

Good



Não é limitada
ao plano do “chão”

Stencil Buffer

- Área de memória, mesma resolução do frame-buffer
- Normalmente um byte por pixel, com valores inteiros (0-255)
- Auxílio para algumas operações, como limitar superfície da sombra

Stencil Buffer

- `void glStencilFunc( GLenum func, GLint ref, GLuint mask );`
- Func
  Specifies the test function. Eight tokens are valid: GL_NEVER, GL_LESS, GL_LEQUAL, GL_GREATER, GL_GEQUAL, GL_EQUAL, GL_NOTEQUAL, and GL_ALWAYS. The initial value is GL_ALWAYS
- Ref
  Specifies the reference value for the stencil test. ref is clamped to the range [0,2n-1], where n is the number of bitplanes in the stencil buffer. The initial value is 0
- Mask
  Specifies a mask that is ANDed with both the reference value and the stored stencil value when the test is done. The initial value is all 1's.

Problemas?

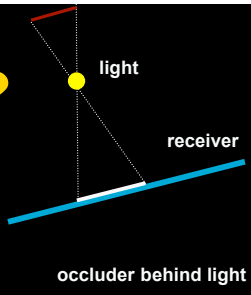
- Limitando o polígono *receiver*
  - Utilizar *stencil buffer* (restringir o desenho somente na área do receiver)

```
1. Desenhe o receiver normalmente no
   frame buffer e stencil
2. Desliga o zbuffer, renderiza o
   polígono de sombra apenas onde o
   receiver foi desenhado (utilizando
   info do stencil)
3. Renderiza o resto da cena
   normalmente
```

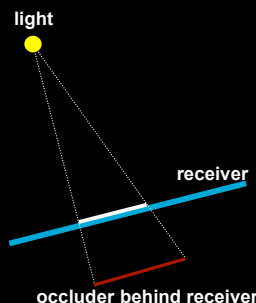
Mais problemas?

- Sombras erradas e Anti-Sombras
  - Objetos “atrás” da fonte de luz
  - Objetos atrás do *receiver*
- Solução: clipping em 3D

  Utiliza hardware gráfico para mapear a geometria e fonte de luz num frustum retangular (proj. ortográfica). Um simples teste em z detecta os casos problemáticos *[Heckbert 1997]*



occluder behind light



occluder behind receiver

Em resumo...

- Uso somente para poucos polígonos grandes
- Fácil de implementar
- Utilização do stencil buffer
- Em alguns casos renderizar a sombra como textura → **Shadow Textures**
  - *Occluders* e *receivers* estáticos por alguns frames...

Shadow Textures

- Utilizar uma imagem da sombra projetada como textura
- Também conhecidas como
  - *Shadow maps (in the game community)*
- Confusão com os *shadow maps* de Williams (1978)

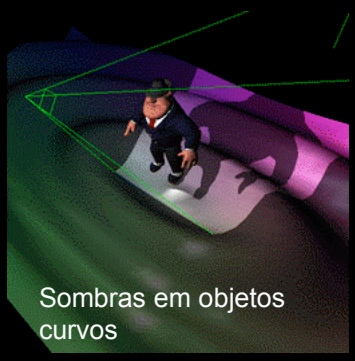
Shadow Textures

Ponto de Vista da Luz

Textura

Textura aplicada no polígono do chão

Shadow Textures
Mais exemplos



Sombras em objetos curvos

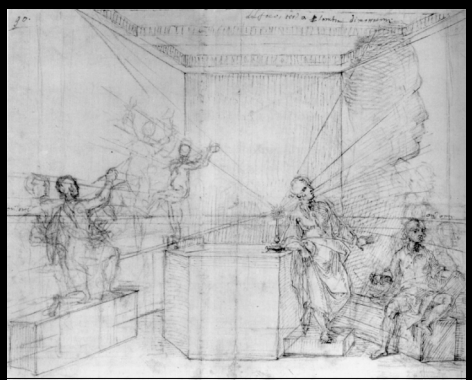
Contribuições de várias fontes de luz somadas

Shadow Textures
Problemas?

- Não utiliza informação de visibilidade da fonte de luz
  - Criar e aplicar as texturas em order *front-to-back* (considerando a fonte de luz)
  - Partes do objeto que não são visíveis pela fonte também terão a textura aplicada
- Usuário precisa identificar objetos *receivers* e objetos *occluders*
- *Receivers* podem já ter textura
  - Duas passadas
  - Multitexture (em hardware que tenha isto...)

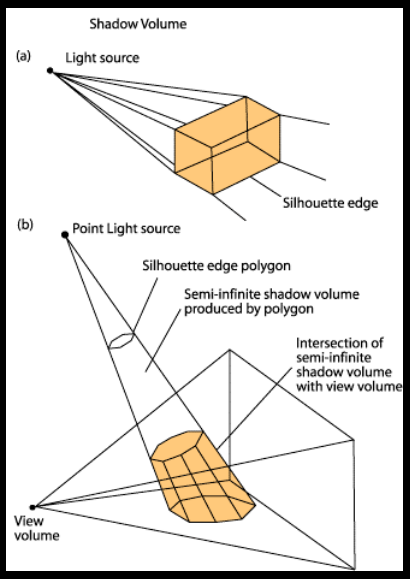
Shadow Volumes

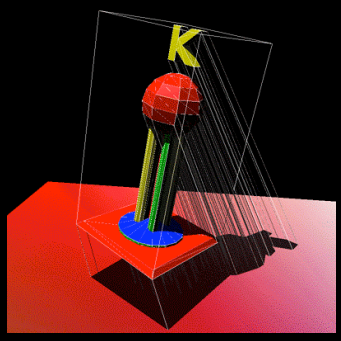
- [Crow77] *Shadow algorithms for computer graphics*
  - Cálculo de *volumes de sombra* em 3D



From the School of Leonardo Da Vinci

Shadow Volumes

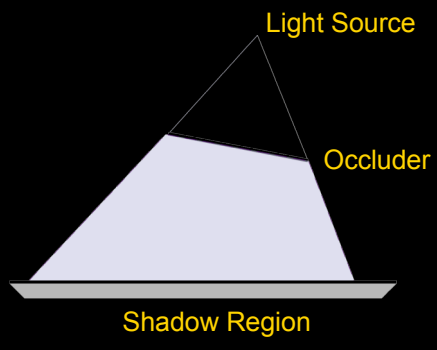




Um volume de sombra, para um objeto e uma fonte de luz, é o volume no espaço que está na sombra com relação àquela fonte de luz e objeto.

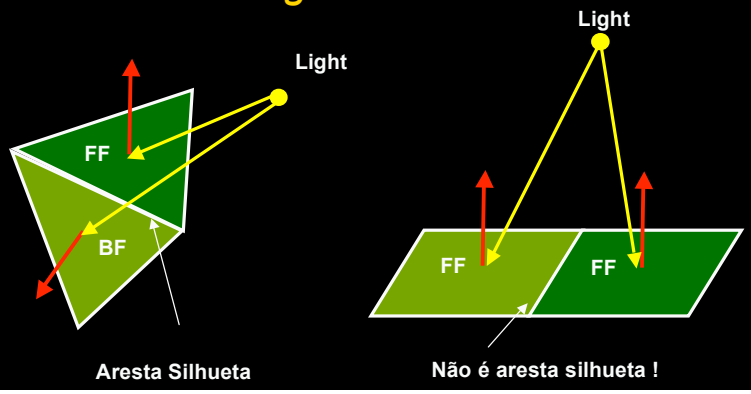
Shadow Volumes

- Calculando o volume
  - Encontrar as arestas da silhueta do objeto,cfe vistas da fonte de luz
  - Extrudar estas arestas formando polígonos tendo a fonte de luz como centro de projeção
  - Recorte destes polígonos contra o frustum



Shadow Volumes

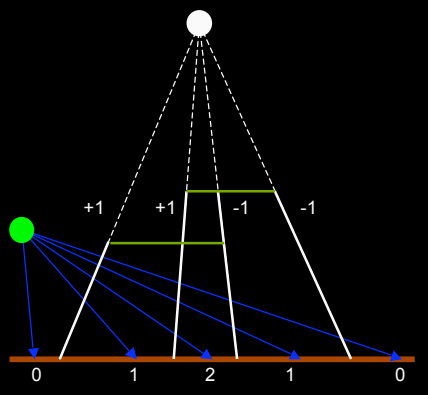
- Arestas da silhueta: uma aresta é da silhueta se ela é uma aresta compartilhada por uma face *front-facing* **e** uma *back-facing*



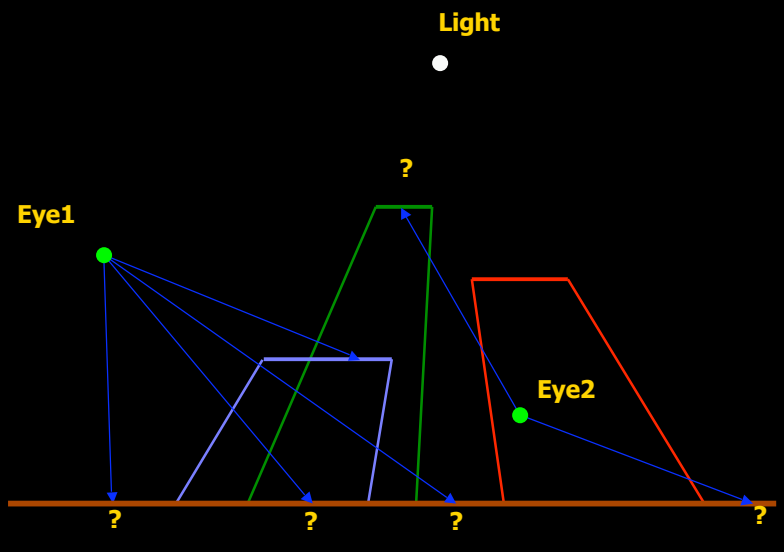
Shadow Volumes

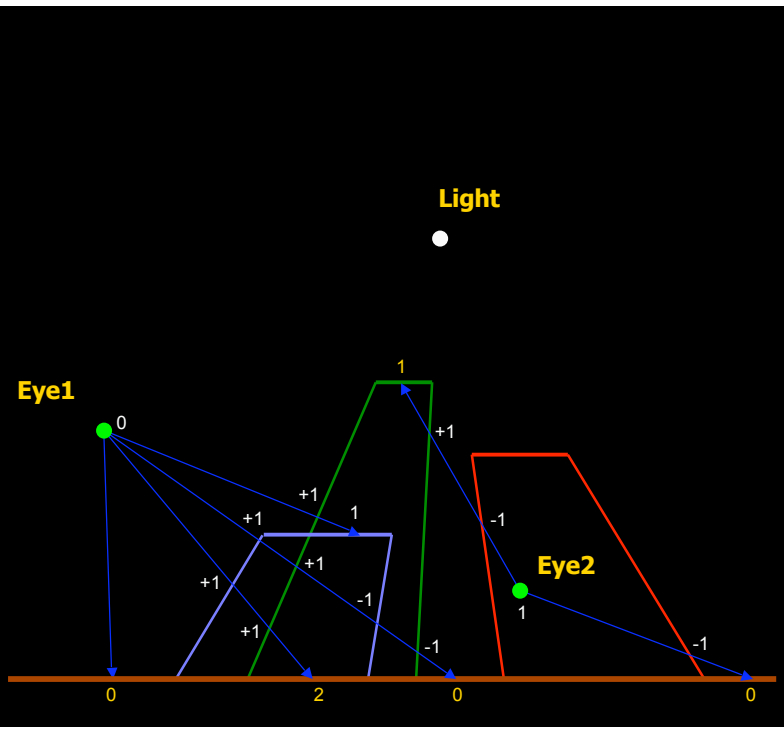
Determinando se um ponto *P* está na sombra:

- Inicializa um contador de cruzamentos dos volumes de sombra = número de volumes de sombra que englobam o olho. Porque? Porque o raio precisa cruzar no mínimo este número para atingir uma região iluminada
- Acompanhando o raio, incrementar o contador cada vez que o raio entra num volume e diminui um cada vez que o raio deixa o volume
- Se o contador for >0, *P* o ponto está na sombra
- Se P estiver num polígono occluder, não atualiza o contador
- Ponto na borda não conta como cruzamento



Utilizando as regras anteriores, indicar se os pontos marcados estão na sombra ou não





Real-Time Shadow Volumes

- Calcular os volumes a cada frame
  - Não é um grande problema se limitarmos a poucos objetos que se movem
  - Utilizar geometria simplificada do objeto para acelerar o cálculo
- Algoritmo em 4 passadas
  - Você realmente quer saber? ☺

Detalhes, detalhes,...

- Turn off any light sources and render scene with ambient light only, depth on, color buffer active
- Disable the color and depth buffers for writing, but leave the depth test active
- Initialize the stencil buffer to 0 or 1 depending on whether the viewer is in shadow or not (ray cast)
- Set the stencil test to *always pass* and the operation to *increment if depth test passes*
- Enable back face culling
- Render the shadow volumes - this will increment the stencil for every front facing polygon that is in front of a visible surface

1

- Enable front face culling, disable back face culling
- Set the stencil operation to decrement if the depth test passes (and leave the test as always pass)
- Render the shadow volumes - this decrements for all the back facing shadow polygons that are in front of visible objects. The stencil buffer now has positive values for placing in shadow
- Set the stencil function to *equality with 0,* operations to *keep*
- Clear the depth buffer, and enable it and the color buffer for writing
- Render the scene with the lights turned on
- Voila, we're done

2

Porque usar Volumes de Sombra?

- Eles calculam corretamente as sombras de todas as superfícies em todas outras superfícies
- Podem ser utlizados com múltiplas fontes de luz
- Não criam falsas sombras, nem anti-sombras
- Podemos controlar a qualidade da sombra pela aproximação que fizermos da geometria do objeto
- Os volumes para pares de objetos e fontes de luz estáticos podem ser pré-calculados

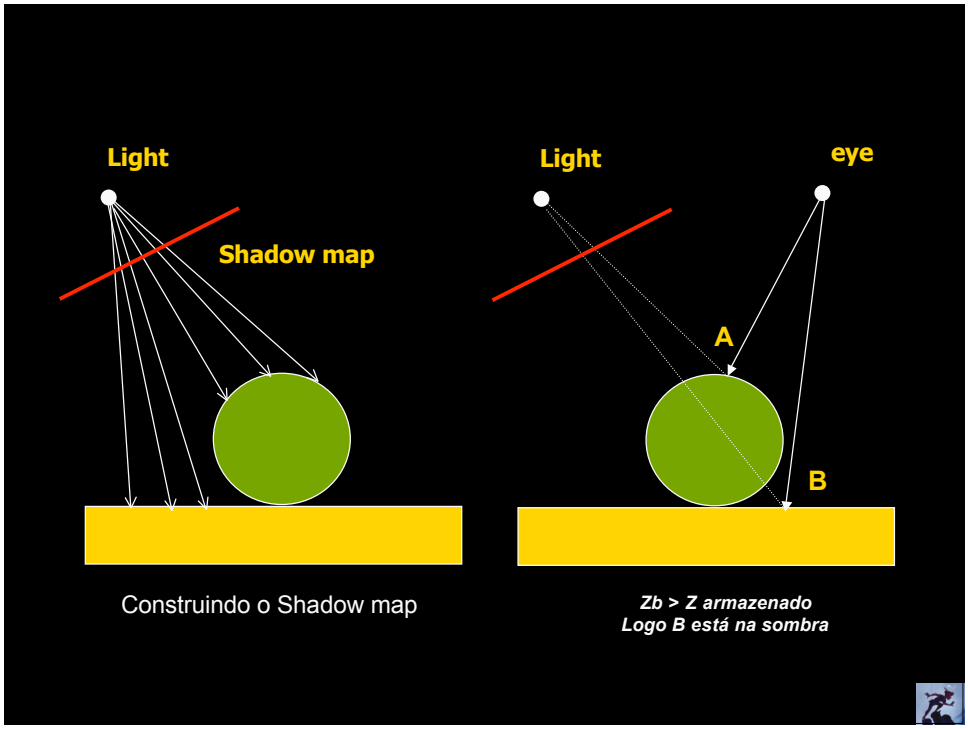
Shadow Maps

- [Williams78]
  Casting curved shadows on curved surfaces
  - Algoritmo no espaço de imagem (assim como zbuffer)

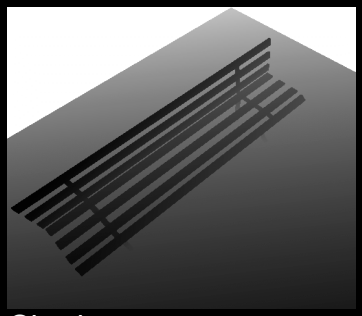
Real-Time Luxo Jr. …uses three dynamic shadow maps (OpenGL, GeForce3)

Shadow Maps - Algoritmo

- Renderiza a cena vista da fonte de luz (para cada fonte de luz)
- Armazena as profundidades (2D *shadow map*)
- Renderiza a cena normalmente
  - Transforma as coordenadas do pixel para a fonte de luz
  - Comparar z com z armazenado no *shadow map*
    - **Pixel está na sombra se z(light) < z(viewer)**

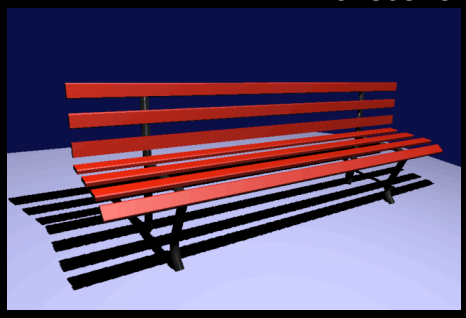


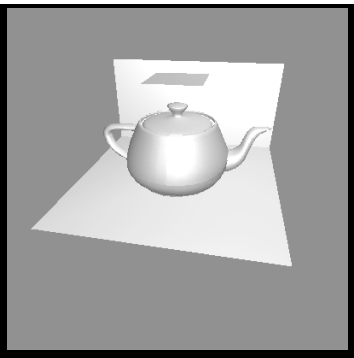
Shadow Maps



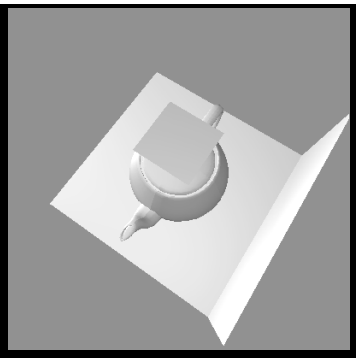
Shadow map

Final scene

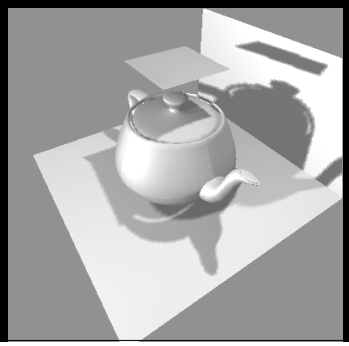




Shadow map 1



Shadow map 2



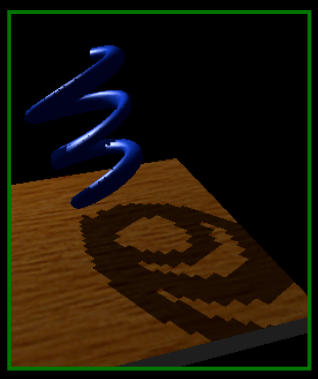
Resultado

Shadow Maps - em resumo...

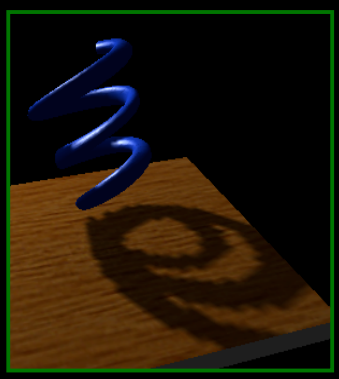
- Padrão em placas gráficas atuais
- Suporte em OpenGL: SGIX_depth_texture, SGIX_shadow
- Problemas de Aliasing
  - Mapas de sombra com alta resolução ou filtragem
  - Problemas numéricos durante o teste de z (utilizar depth bias - `glPolygonOffset`)

Shadow Maps
Amostragem da Geometria

GL_NEAREST: blocky



GL_LINEAR: antialiased edges

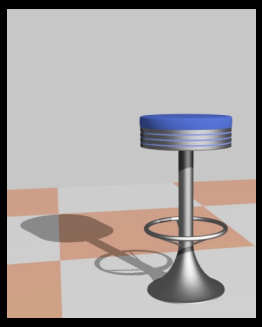


Algoritmos for Soft Shadows

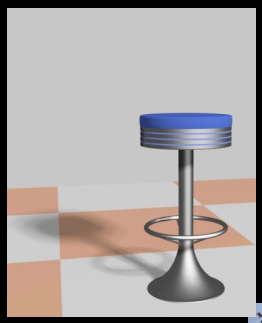
- Amostrando a fonte de luz
- *Plateau*

Várias amostras...

- Escolher um algoritmo para hard shadows
  - Selecionar vários pontos na fonte de luz distribuída
  - Renderizar somando as contribuições



hard shadow



accumulated hard shadows -> soft

Várias Amostras

- Exemplo: shadow texture soft

```
1. Initialize FB (white)
2. For each sample point
(total N) do
   2a. Render scene
   2b. Subtract 1/N from FB
       only once for each
       pixel (stencil) !
```

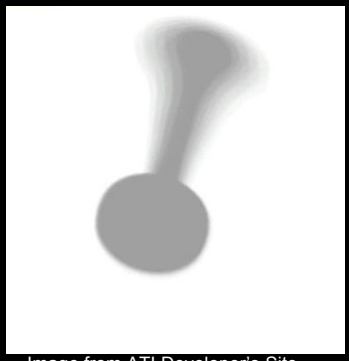


Image from ATI Developer's Site

Várias Amostras

- Resultado

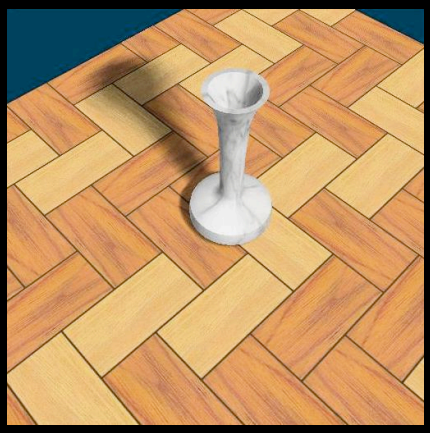


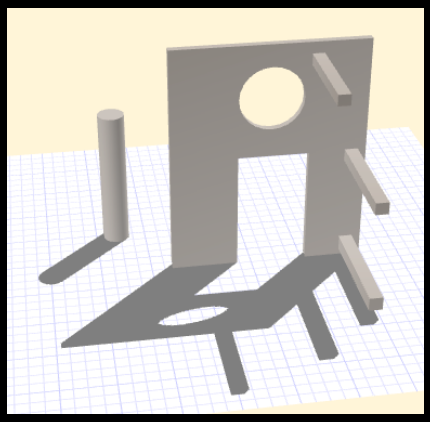
Image from ATI Developer's Site

Considerações

- Número de amostras == número de passadas
  - Problemático para cenas complexas
- Quantas amostras? Regular? Jittered?

Sombras Plateau

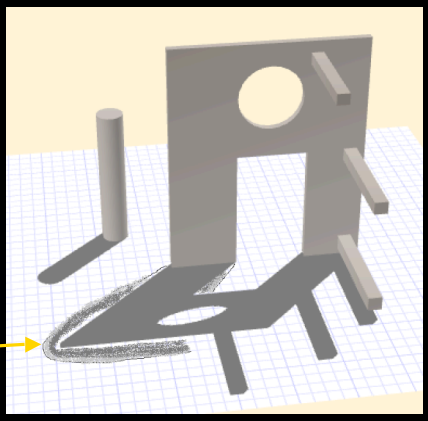
- Vamos supor que queremos sombras soft para uma imagem



Eric Haines. ***Soft Planar Shadows Using Plateaus***
Journal of Graphics Tools, V.6, n. 1, p.19-27, 2001

Sombras Plateau

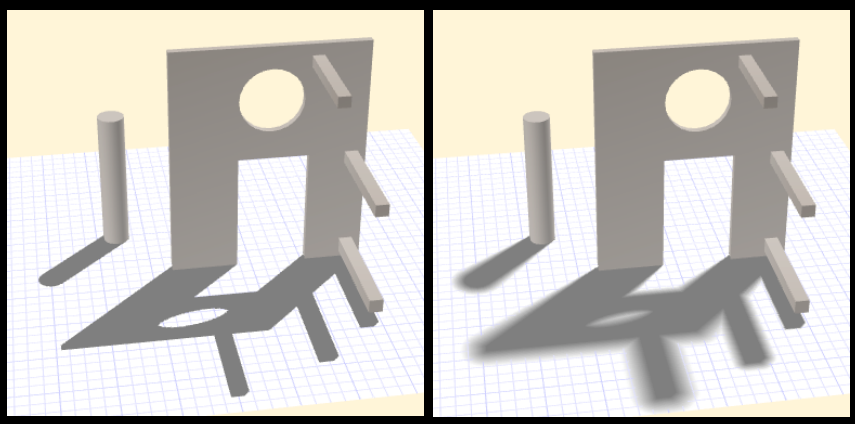
- “Pintar” cada borda da sombra, com um gradiente
- Área do gradiente proporcional à altura da aresta



Calculando os plateaus

- Cada aresta do polígono resulta num quadrilátero de sombra
- Cada vértice resulta num círculo de sombra
- Renderizar estas figuras mais escuro no centro, e mais claro em direção às bordas

Resultado - Plateau



Conclusões?

- Sem sombras o realismo fica comprometido
- Sombras *bonitas* são caras computacionalmente
  - Simulação de um Efeito Global com modelos de iluminação locais!

Otimizações

- Explorar a nossa incapacidade de perceber se a sombra está correta
- Utilizar modelo geométrico menos detalhado
- Atualizar a informação de sombras a cada par de frames (ou menos ainda...)
- Atualização decrescente em função da distância do observador (até chegar em 0)
- Não ter todas as luzes ativas o tempo todo
- Luzes com distância de funcionamento
- Eliminar polígonos que não vêem a luz (*light culling*)

Charles Bloom. *Advanced Techniques in Shadow Mapping*
www.cbloom.com/3d

Custo...

Elder Scrolls IV:
Oblivion
Shadows on Grass

Custo…

Elder Scrolls IV:
Oblivion
Tree Canopy Shadows

Custo…

Elder Scrolls
IV: Oblivion

Referências

T. Moeller, E. Haines: *Real-Time Rendering*

P. Heckbert: *Simulating Soft Shadows with Graphics Hardware* (Tech Report)

L. Williams: *Casting curved shadows on curved surfaces* (SIGGRAPH 78)

M. Segal: *Fast shadow and lighting effects using texture mapping* (SIGGRAPH 92)

A. Woo: *A survey of shadow algorithms* (IEEE Computer&Applications 90)

W. Reeves: *Rendering antialiased shadows with depth maps* (SIGGRAPH 87)

T. Heidmann: *Real shadows real time* (IRIS Universe)

J. Blinn: *Me and My (Fake) Shadow* (Jim Blinns Corner 88)

A. Woo, P. Poulin, A. Fournier: *A Survey of Shadow Algorithms (IEEE CG&A, vol. 10, n. 6, p.13-32, nov. 1990)*