

Computação Gráfica

Aula 24: Sombras e Reflexões



in a shadowmap, a depthmap, a pointcloud.

Tópicos da Aula

Sombras

Ambiente Occlusion

Reflexões

Interreflexões

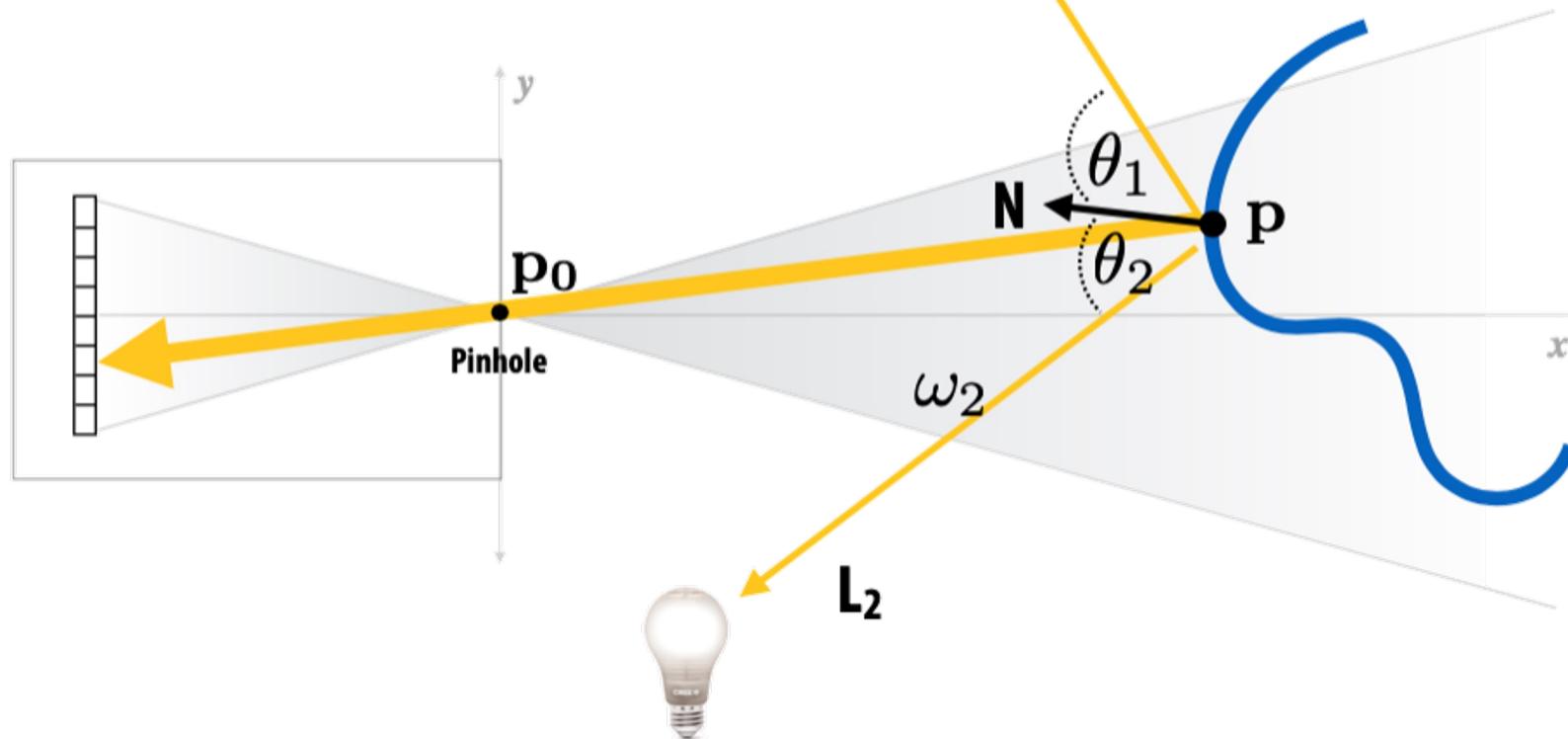
Sombras



Créditos: Grand Theft Auto V

Revisão: quanta luz é refletida de P através do P_0

$$L(\mathbf{p}, \omega_o) = \sum_i f(\mathbf{p}, \omega_i, \omega_o) L_i \cos \theta_i$$
$$\omega_o = \text{normalize}(\mathbf{p}_0 - \mathbf{p})$$

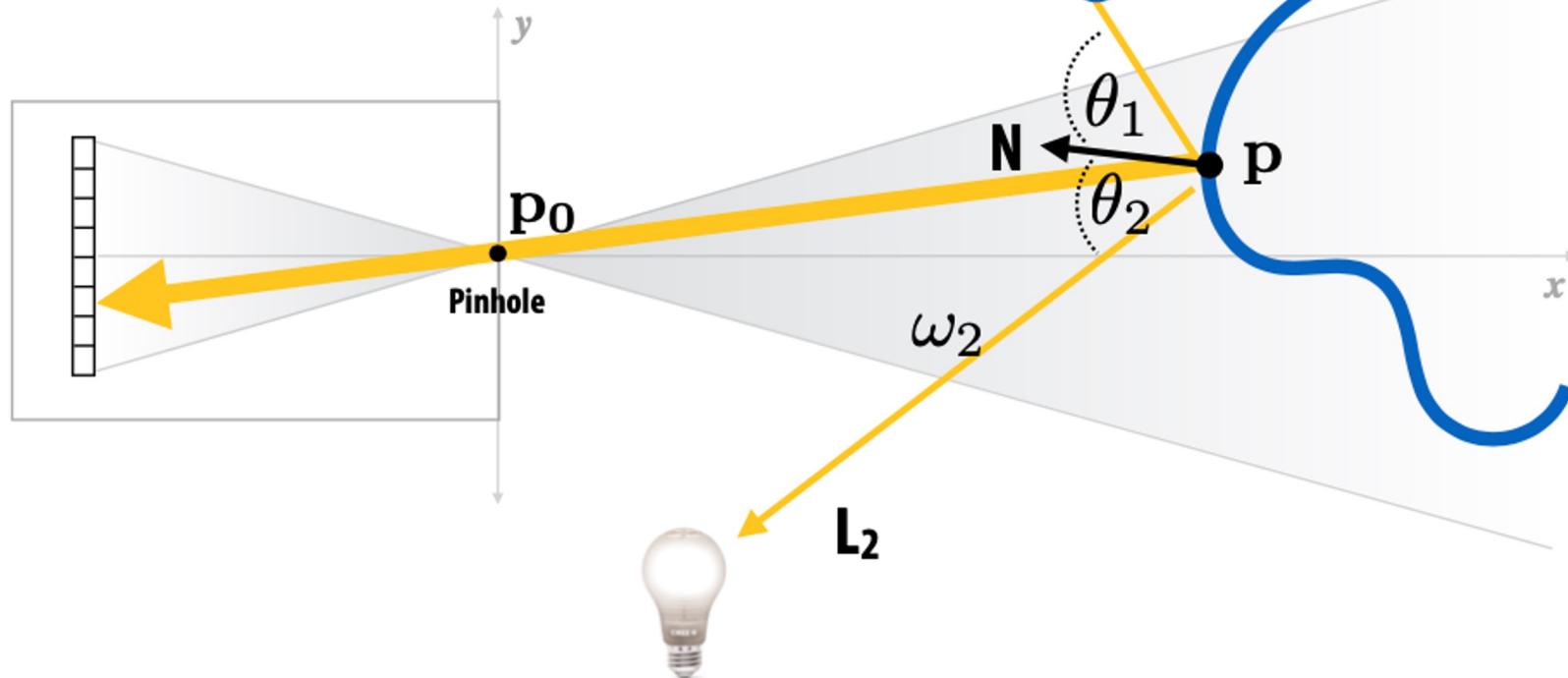


Visibilidade

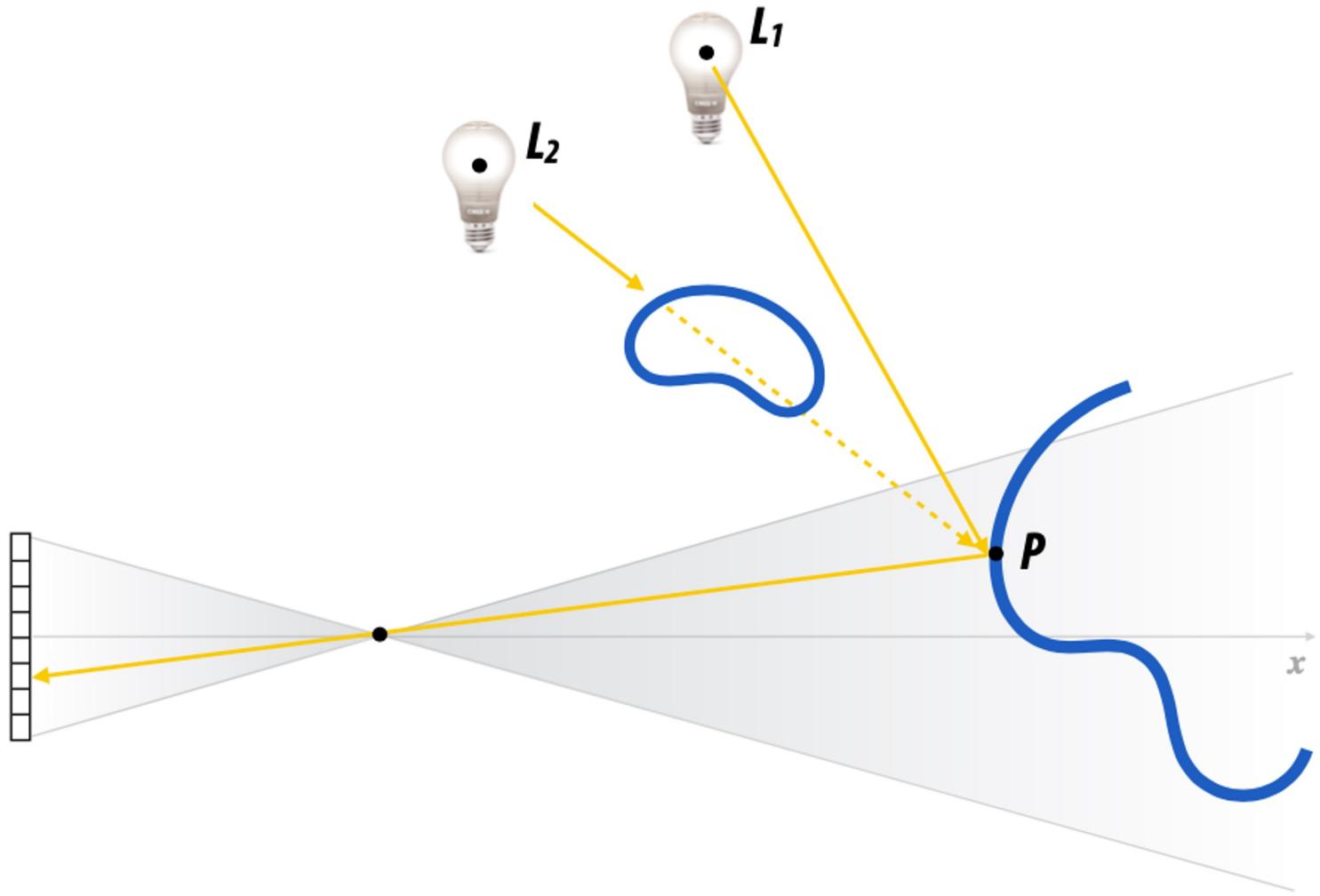
$$L(\mathbf{p}, \omega_o) = \sum_i f(\mathbf{p}, \omega_i, \omega_o) V(\mathbf{p}, \mathbf{L}_i) L_i \cos \theta_i$$

termo de visibilidade:

$$V(\mathbf{p}, \mathbf{L}_i) = \begin{cases} 1, & \text{se } \mathbf{p} \text{ é visível de } \mathbf{L}_i \\ 0, & \text{caso contrário} \end{cases}$$

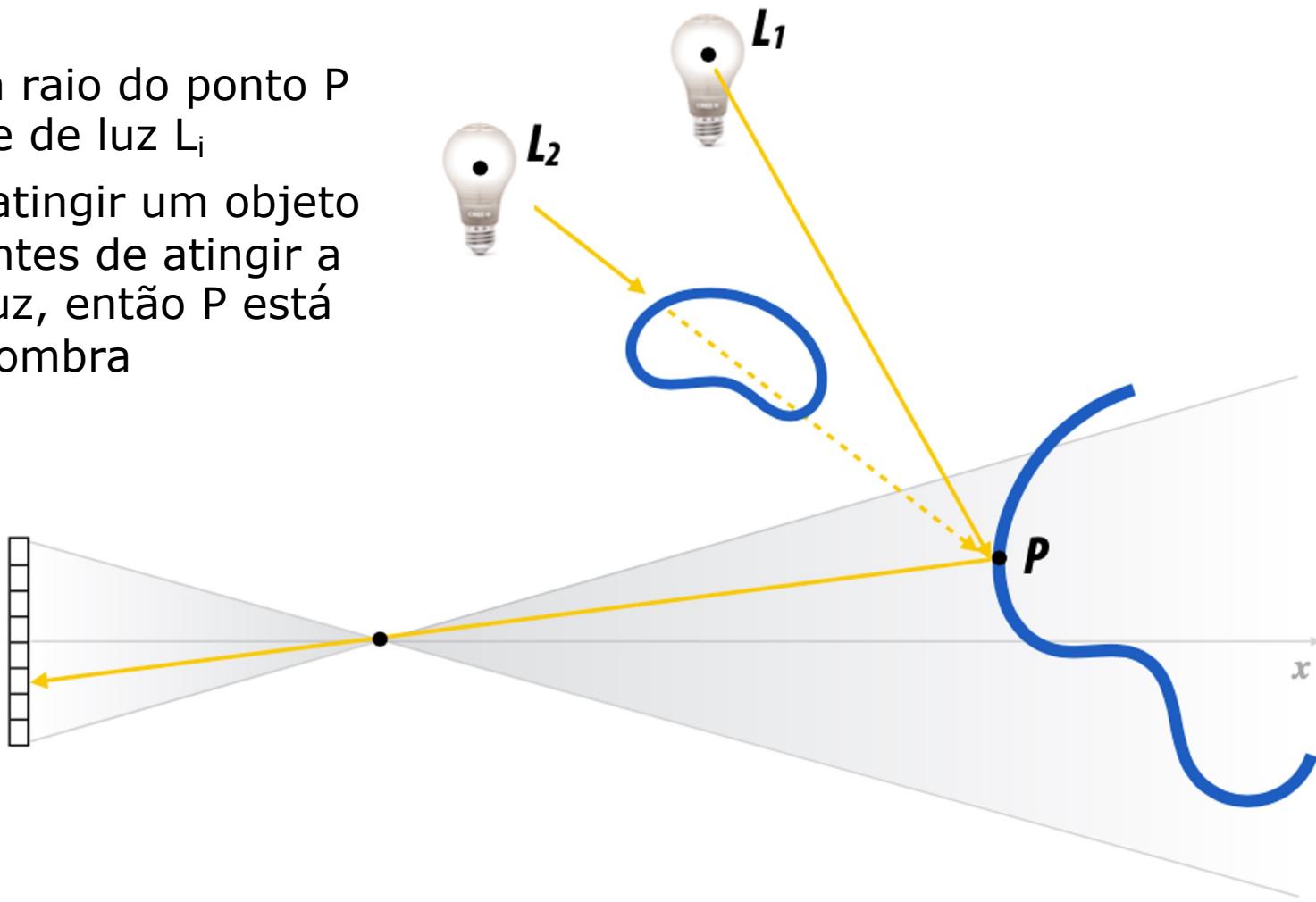


Como saber se superfície está em uma sombra?



Como calcular $V(p, L_i)$?

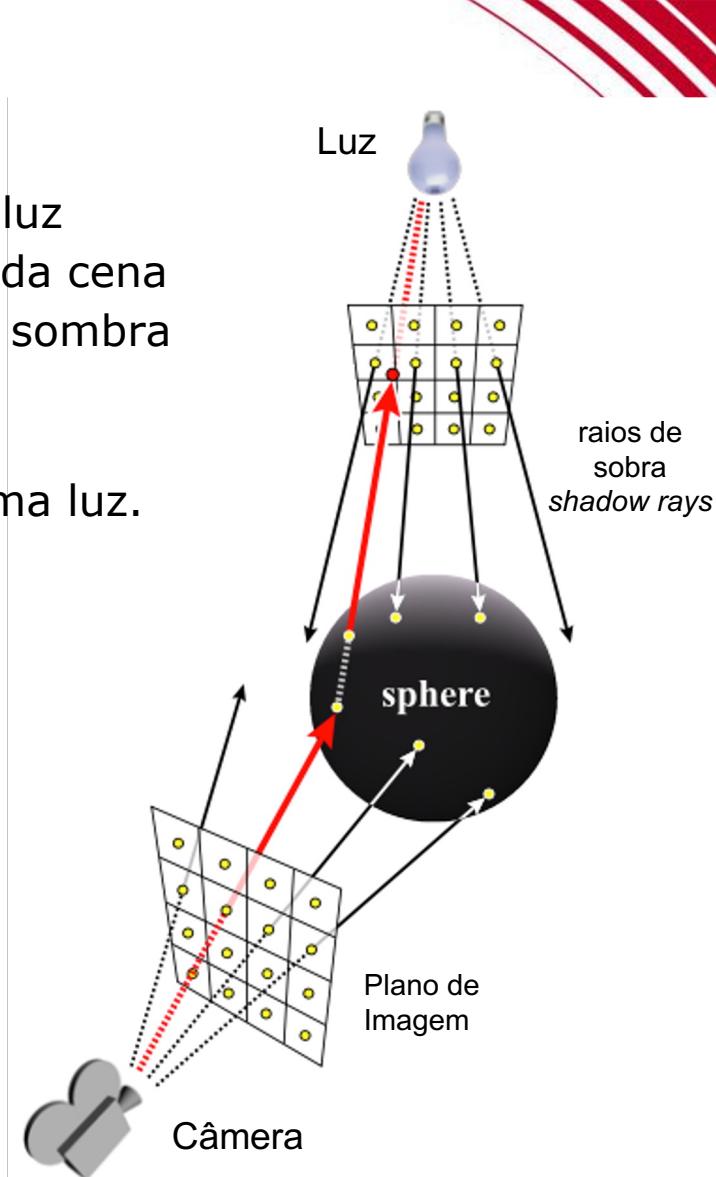
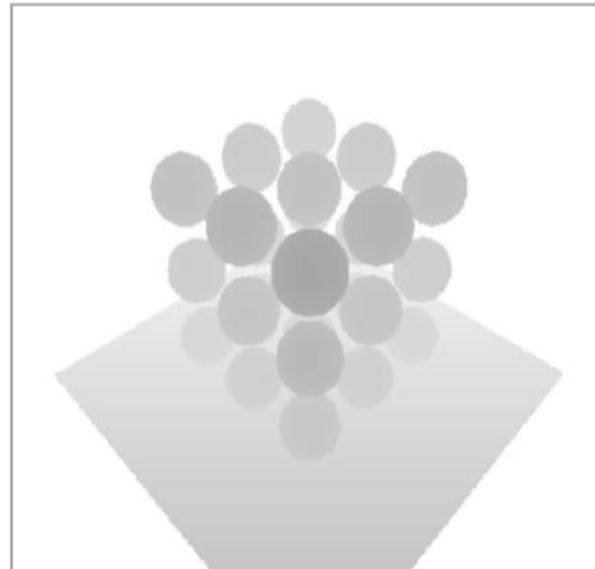
- Lançar um raio do ponto P até a fonte de luz L_i
- Se o raio atingir um objeto da cena antes de atingir a fonte de luz, então P está em uma sombra



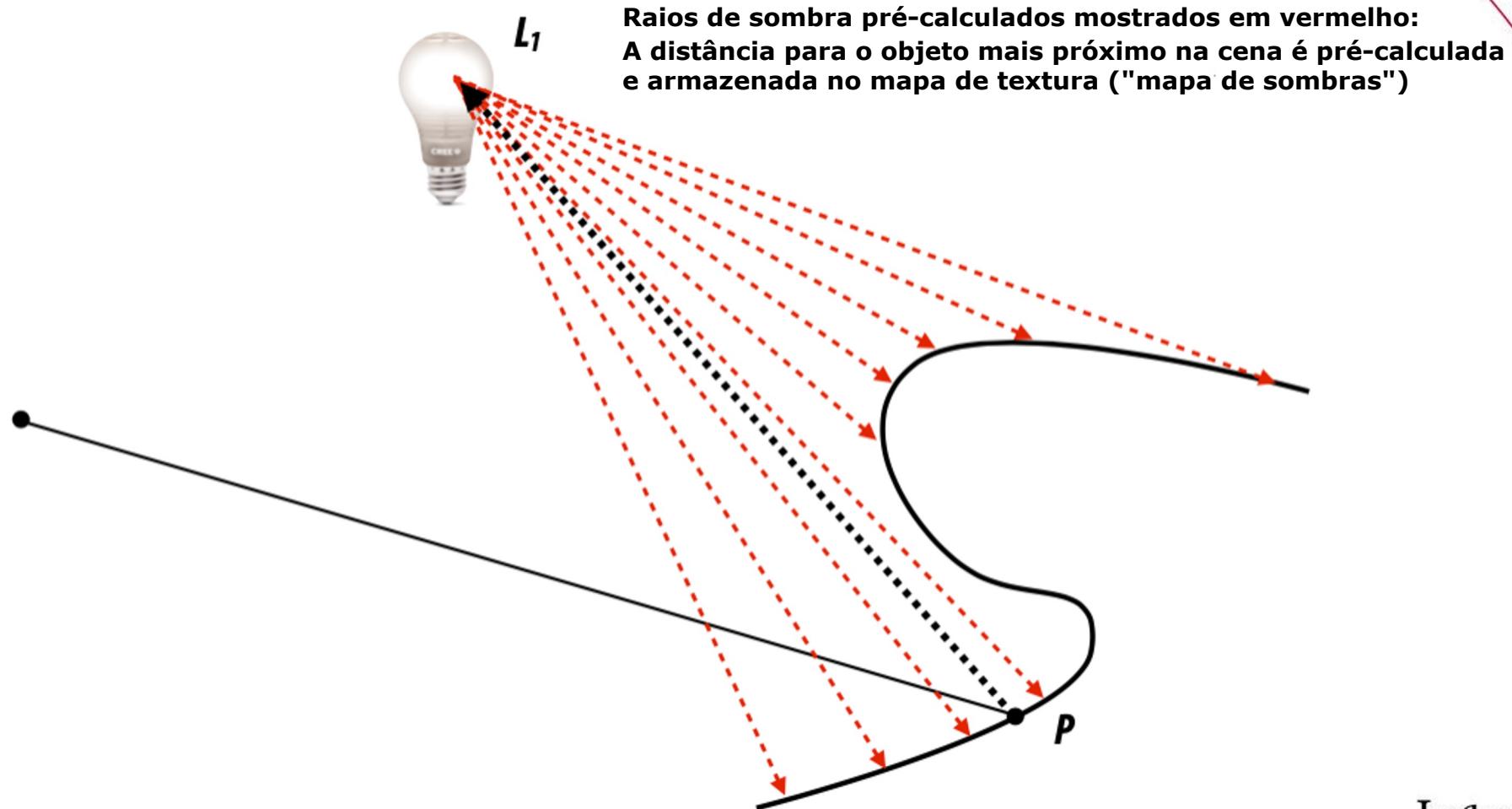
Shadow Mapping [Williams 78]

1. Coloque a câmera na posição de uma fonte de luz
2. Renderize a cena para calcular a profundidade da cena
3. Armazene resultados de interseção de raios de sombra pré-computados em uma textura

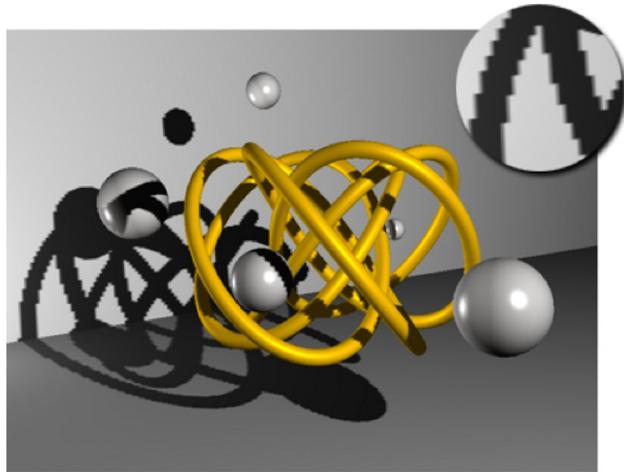
“Mapa de sombra” = mapa de profundidade de uma luz.



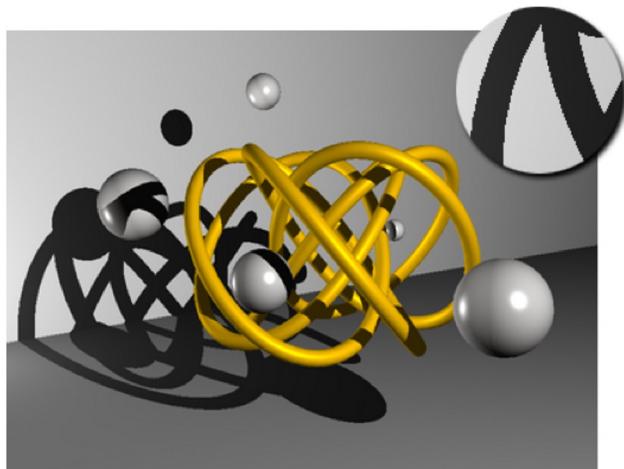
Pesquisa de textura de sombra



Artefatos na sombra devido a baixa amostragem

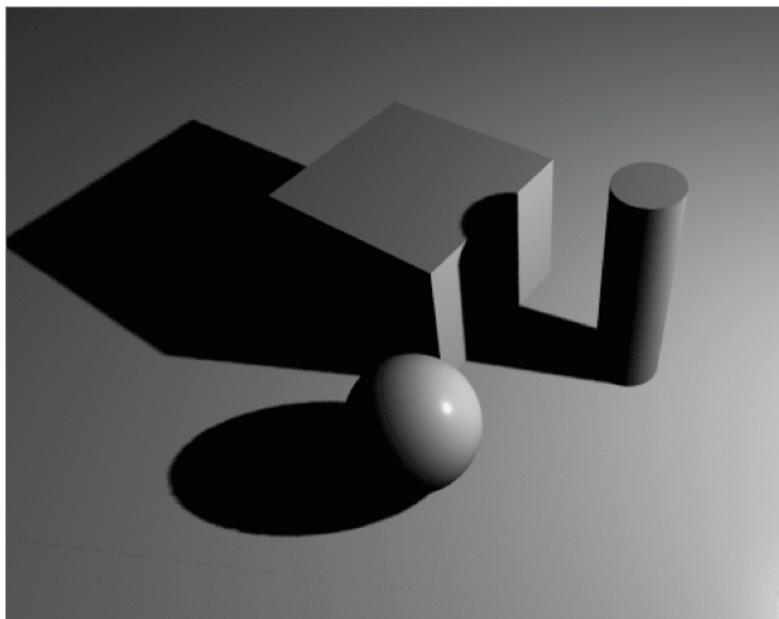


Sombras calculadas usando mapa de sombra



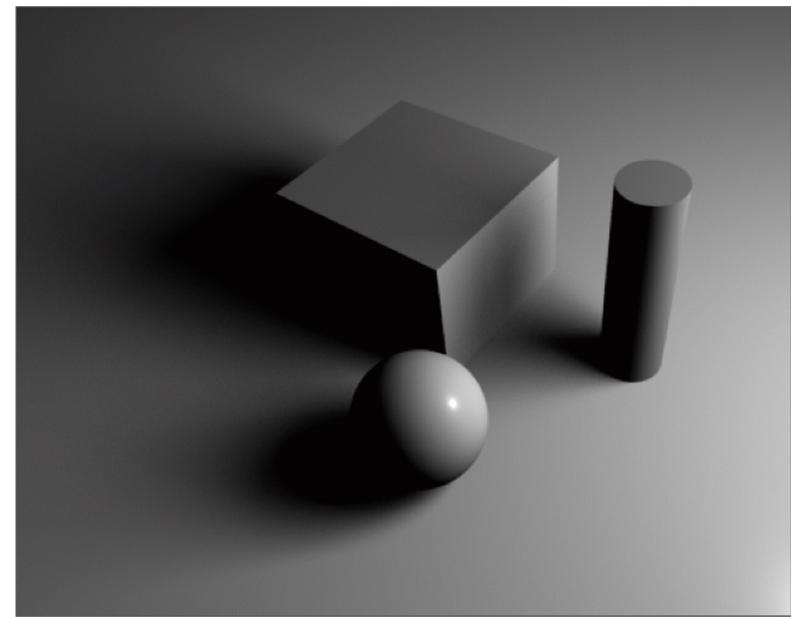
Sombras usando ray tracing
(resultado do cálculo da visibilidade ao longo do
raio entre o ponto de superfície e a luz
diretamente usando ray tracing)

Tipos de sombras (Hard vs Soft Shadows)



Hard shadows

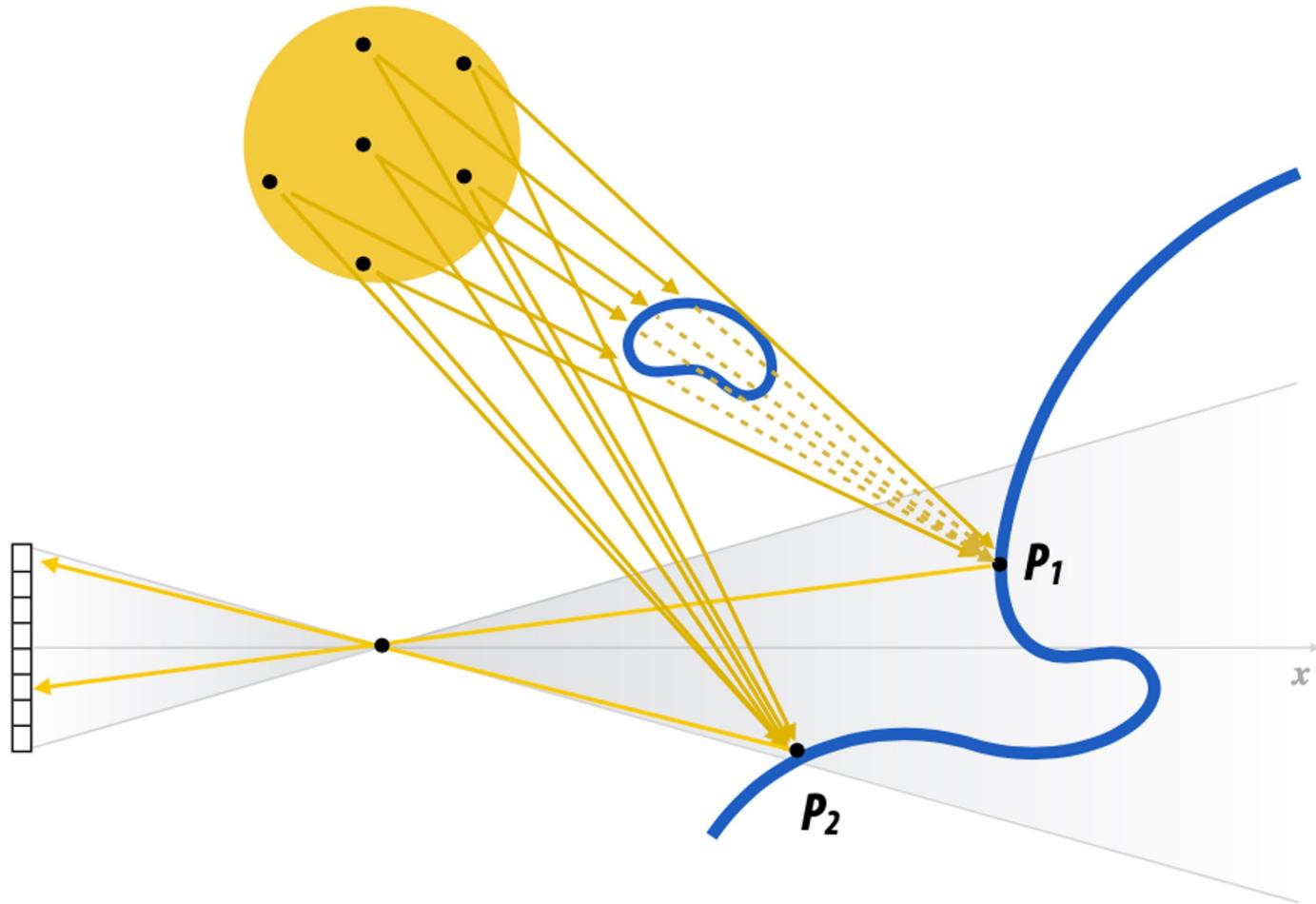
(criadas por uma luz pontual)



Soft shadows

(criadas por uma luz não pontual)

Sobras criadas por uma luz de área



Percentage closer filtering (PCF)

- Em vez de amostrar o mapa de sombra uma vez, execute várias pesquisas em torno da coordenada de textura desejada
- Ajustar a fração das pesquisas que estão na sombra, modular a intensidade da luz de acordo



Hard Shadows
(uma busca por pixel)

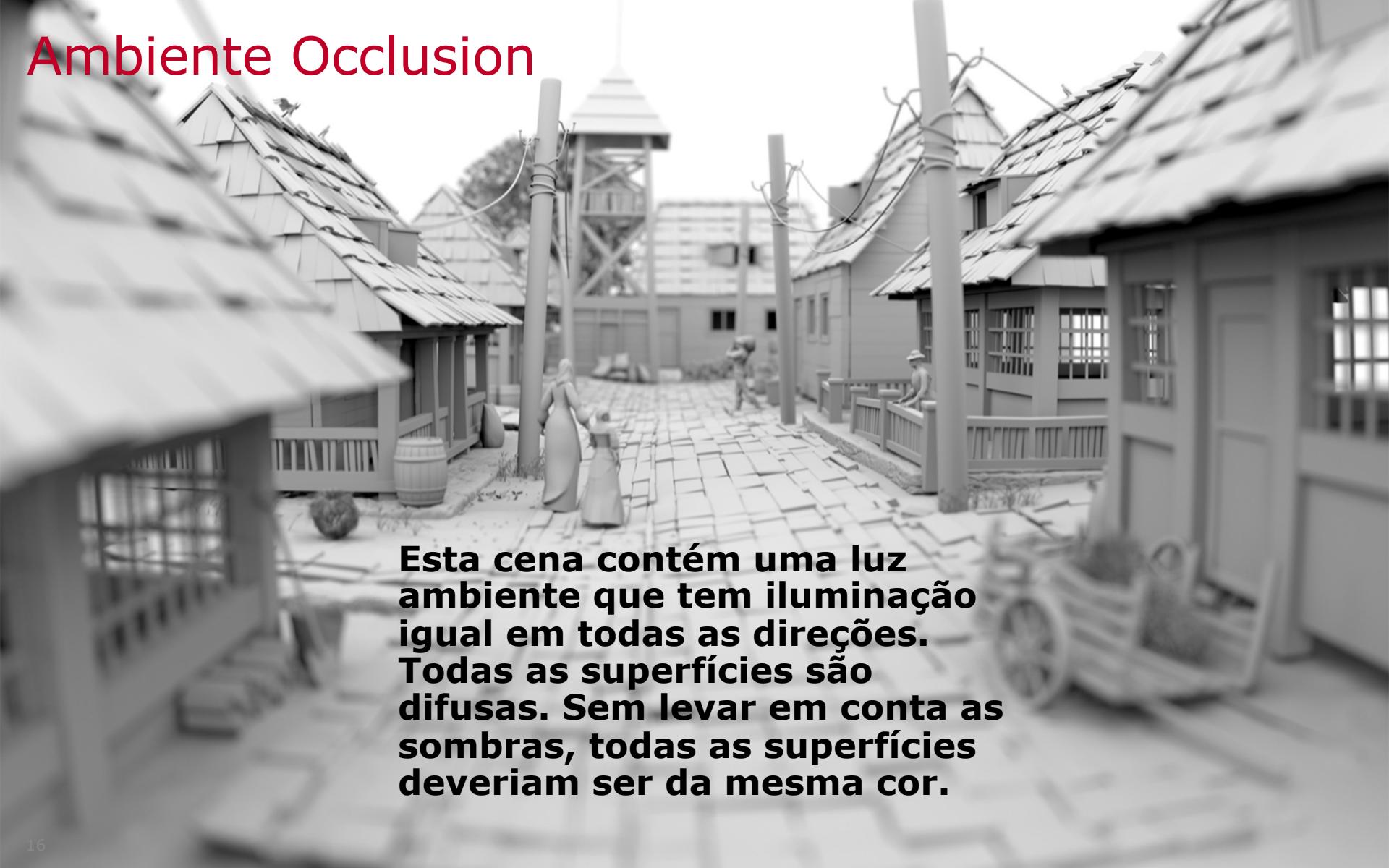


Soft Shadows
(16 buscas por pixel)

Mapa de sombras

0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

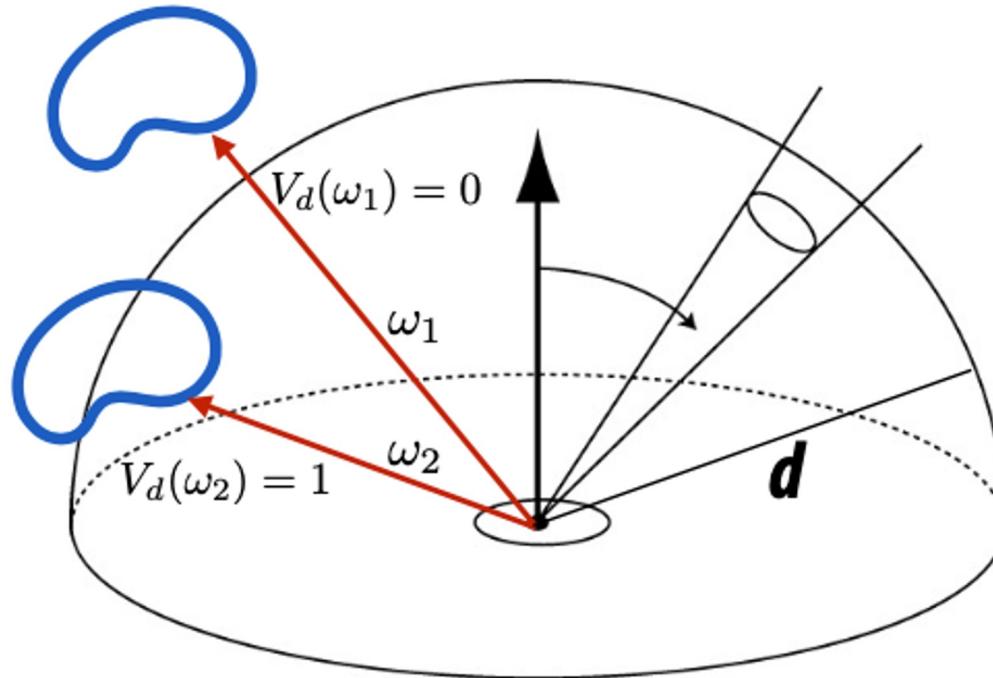
Ambiente Occlusion



Esta cena contém uma luz ambiente que tem iluminação igual em todas as direções. Todas as superfícies são difusas. Sem levar em conta as sombras, todas as superfícies deveriam ser da mesma cor.

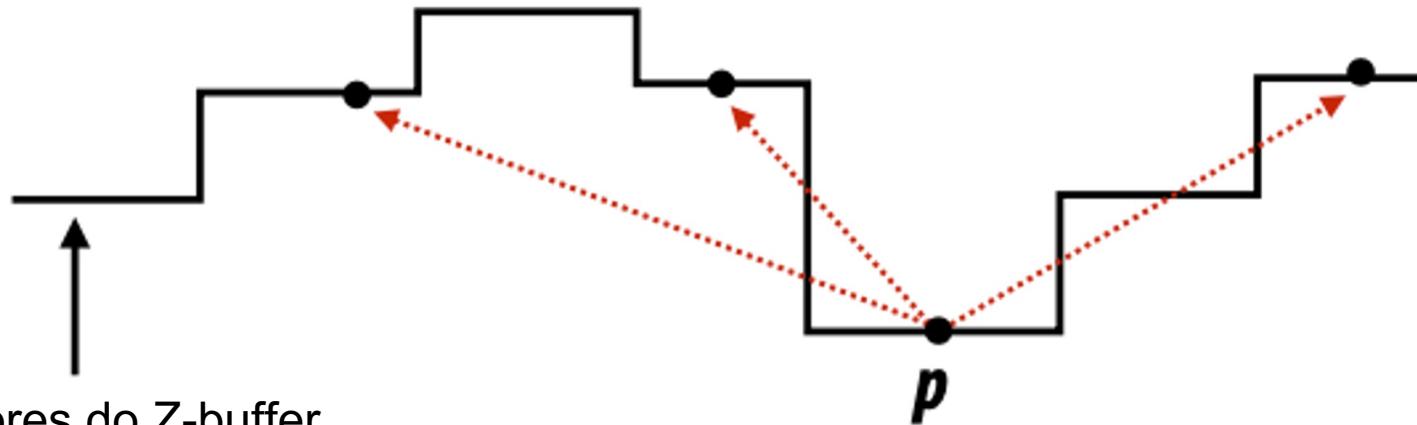
Ambient Occlusion

Idéia: Pré-calcule a “fração de hemisfério” que está ocluída (escondida) até uma distância d de um ponto. Ao sombrear, atenue a iluminação na quantidade proporcional.



Ambient Occlusion usando o "espaço da tela"

1. Renderizar a cena com o buffer de profundidade (z-buffer)
2. Para cada pixel p (estimar a oclusão do hemisfério traçando raios na vizinhança do buffer de profundidade)
3. Faça alguma média do mapa de oclusão para reduzir o ruído
4. Escureça a iluminação ambiente por quantidade de oclusão



Valores do Z-buffer

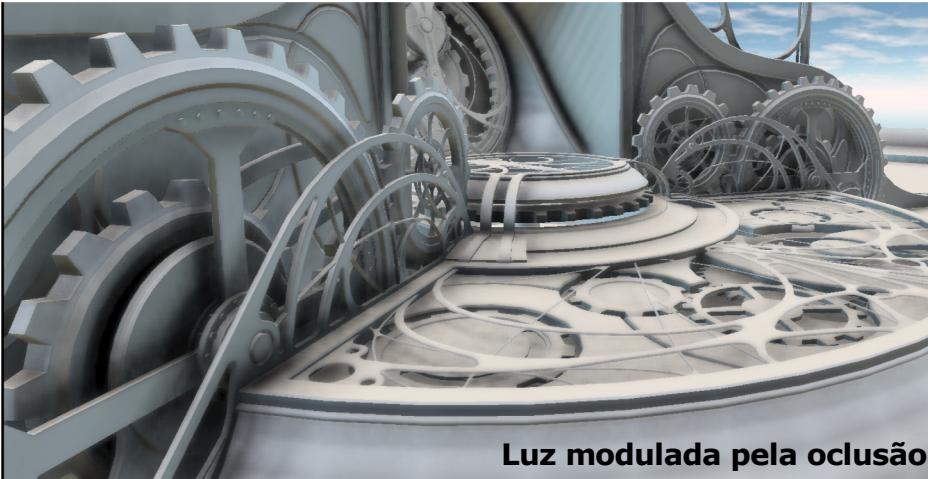
Buda sem Ambiente Occlusion



Buda com Ambiente Occlusion



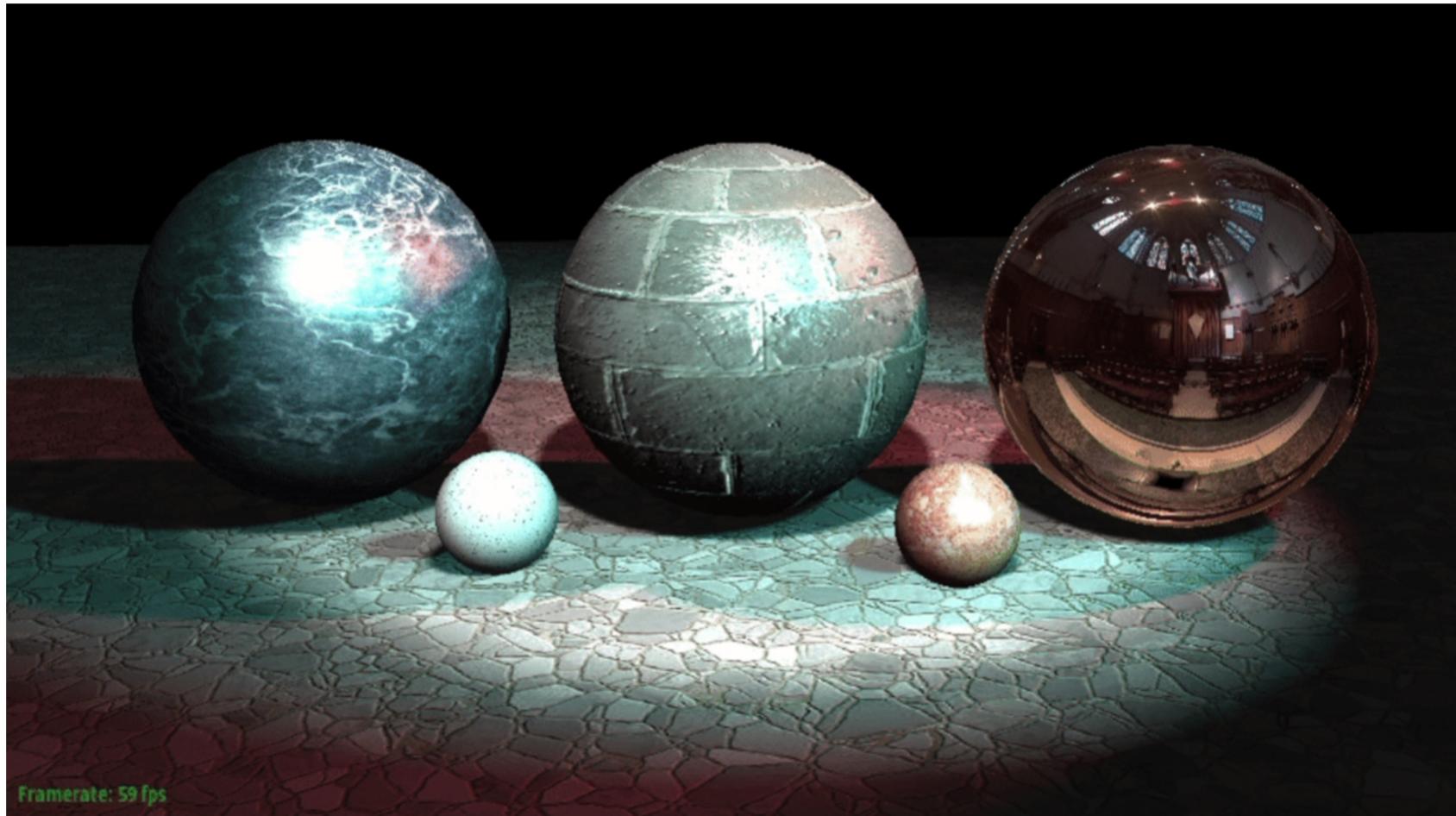
Ambient Occlusion



Reflexões



O que está errado nessa figura?



Framerate: 59 fps

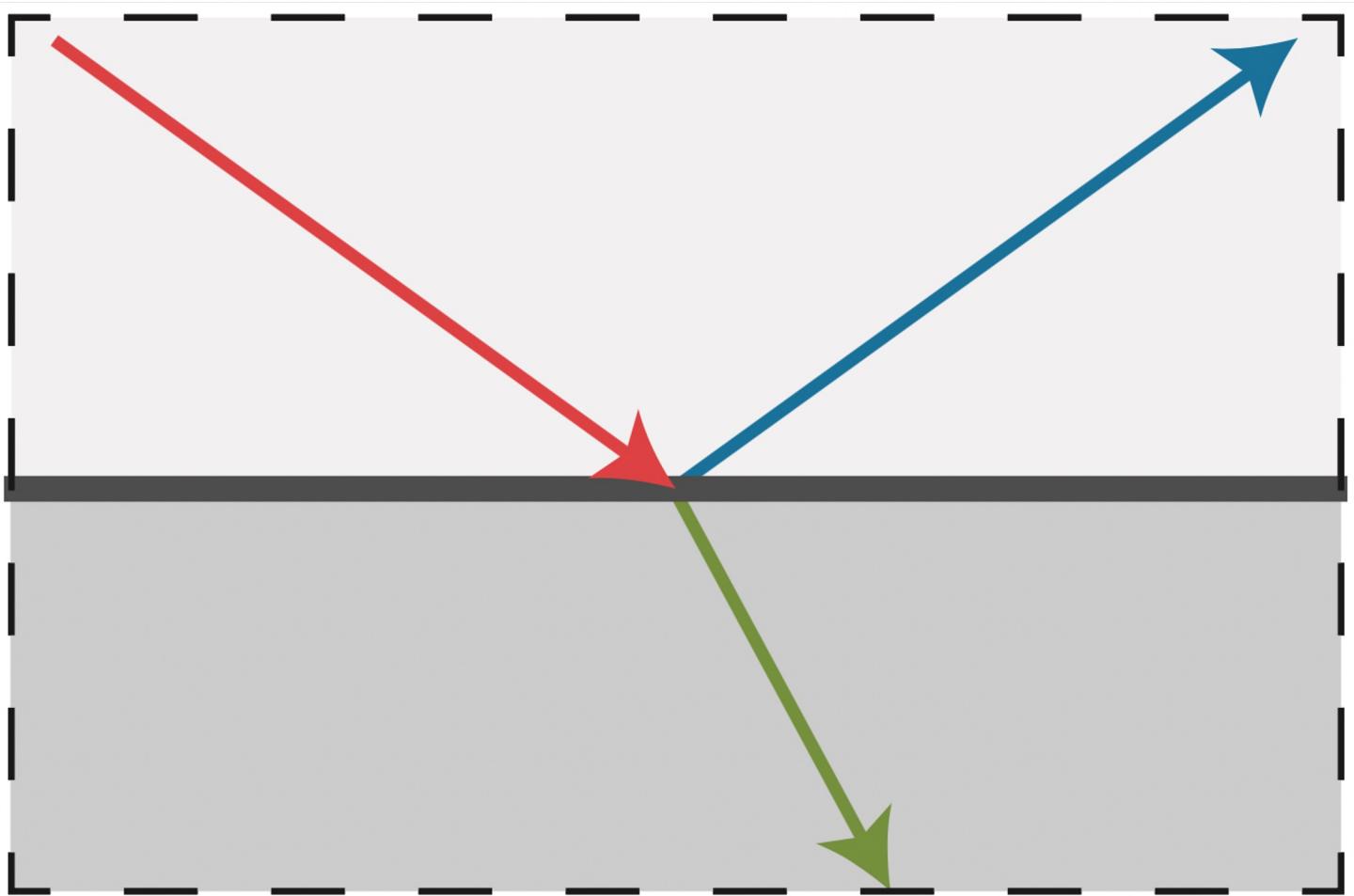
Reflexões



Reflexões

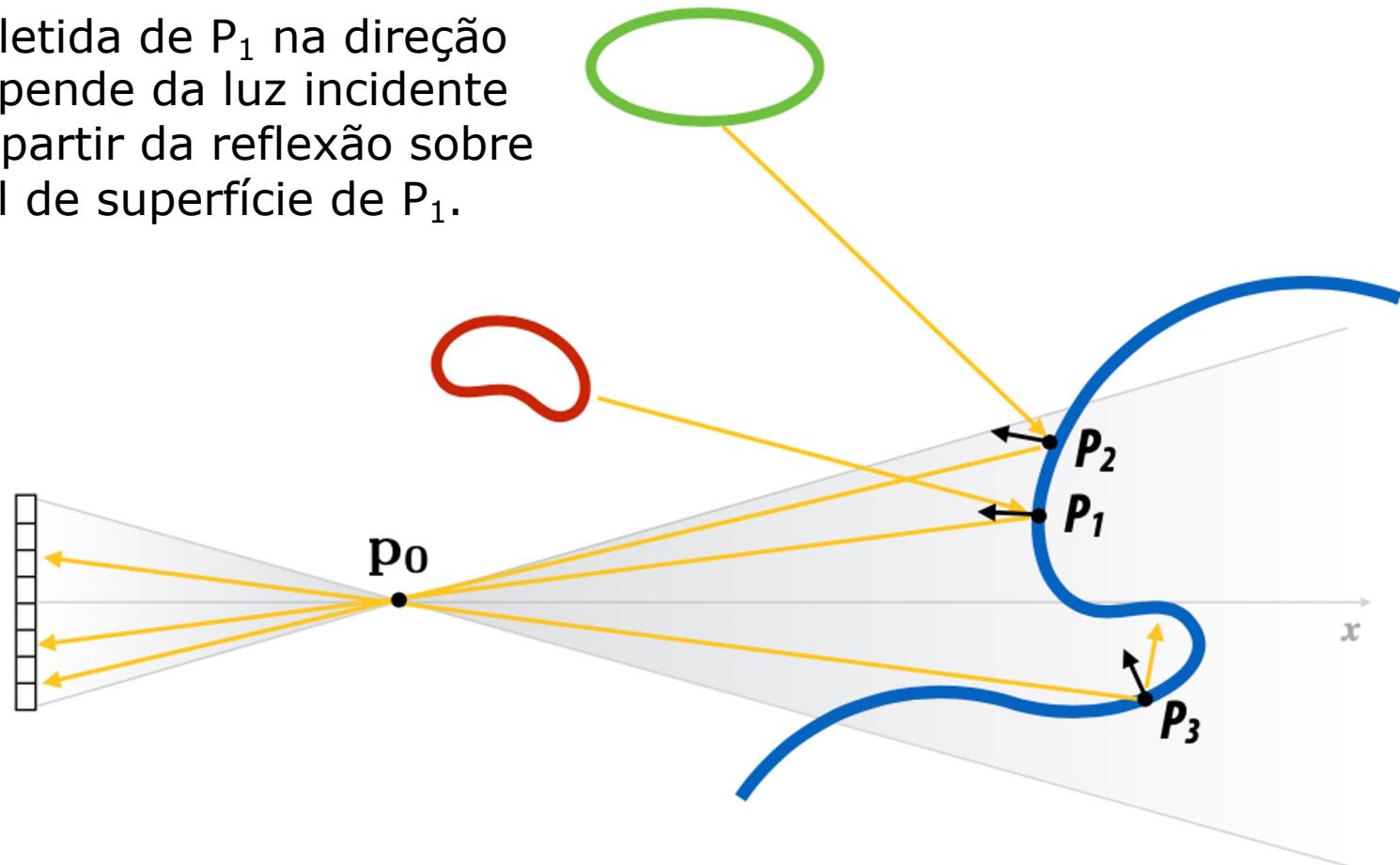


Relembrando: Espelho Perfeito



Reflexão Perfeita

A luz refletida de P_1 na direção de P_0 depende da luz incidente em P_1 a partir da reflexão sobre a normal de superfície de P_1 .

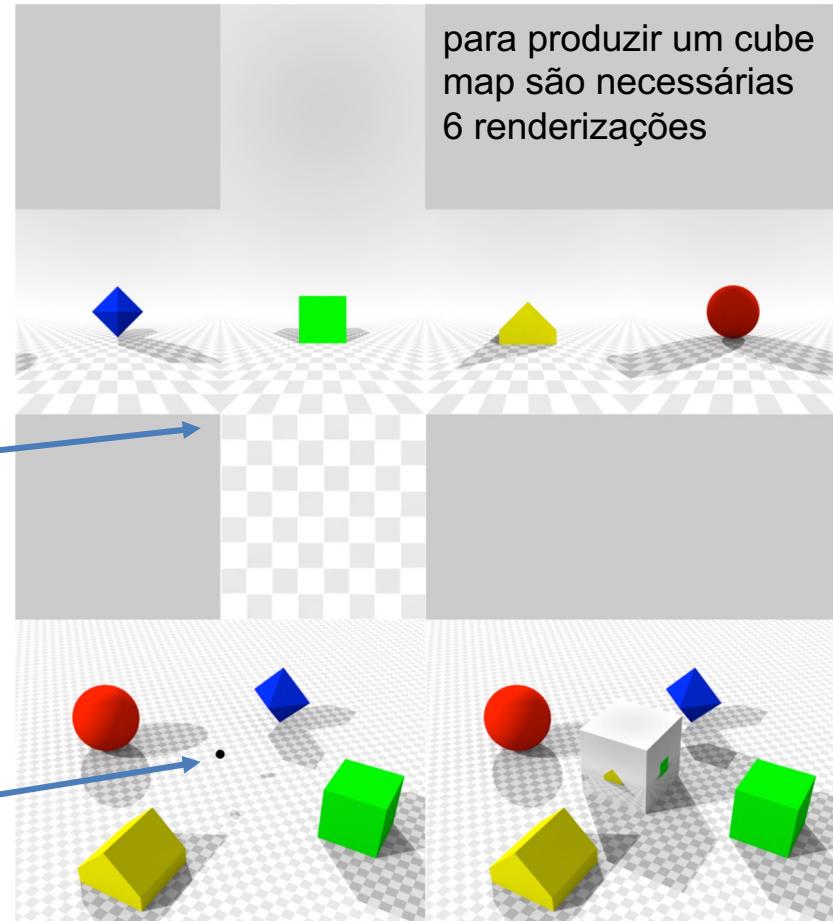


Câmera pode ser usada para simular reflexão

Environment mapping:
coloque câmera na origem
das reflexões

Cube map:
armazena os resultados de raios de
reflexão

Centro da projeção



Obs: Essa técnicas é uma aproximação.

crédito: http://en.wikipedia.org/wiki/Cube_mapping

Reflexões Environment Map vs. Ray Tracing



<https://www.techspot.com/article/1934-the-state-of-ray-tracing/>

Reflexões Environment Map vs. Ray Tracing



<https://www.techspot.com/article/1934-the-state-of-ray-tracing/>

Reflexões Environment Map vs. Ray Tracing



<https://www.techspot.com/article/1934-the-state-of-ray-tracing/>

Interreflexões Difusas

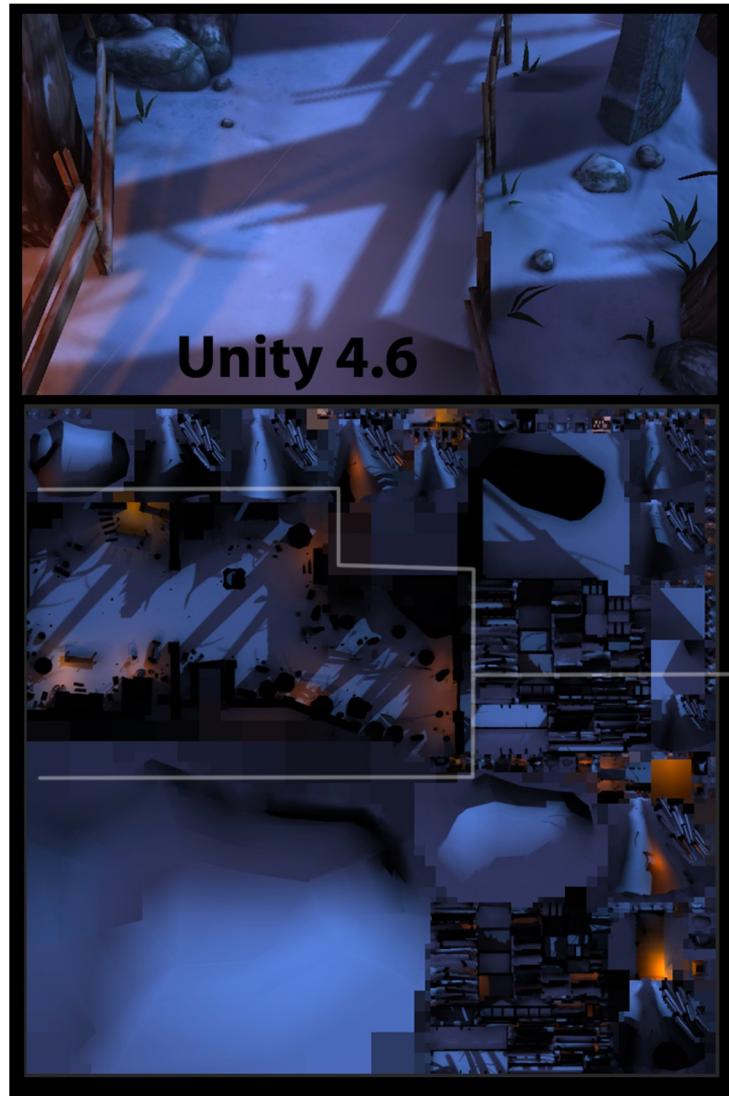


Luzes pré-computadas

Pré-calcular a iluminação para uma cena offline (possível para luzes estáticas)

Cálculos offline podem realizar cálculos de iluminação mais avançados, como os cálculos de interreflexões

Os resultados dos cálculos são armazenados como texturas (ou mapa de texturas)



Sombras em Ray Marching



Criando cenário com Plano e uma Pirâmide

```
float sdPlane( vec3 p, vec3 n, float h ) {
    // n must be normalized
    return dot(p,n) + h;
}

float sdPyramid( vec3 p, float h ) {
    float m2 = h*h + 0.25;
    p.xz = abs(p.xz);
    p.xz = (p.z>p.x) ? p.zx : p.xz;
    p.xz -= 0.5;

    vec3 q = vec3( p.z, h*p.y - 0.5*p.x, h*p.x + 0.5*p.y );
    float s = max(-q.x,0.0);
    float t = clamp( (q.y-0.5*p.z)/(m2+0.25), 0.0, 1.0 );
    float a = m2*(q.x+s)*(q.x+s) + q.y*q.y;
    float b = m2*(q.x+0.5*t)*(q.x+0.5*t) + (q.y-m2*t)*(q.y-m2*t);
    float d2 = min(q.y,-q.x*m2-q.y*0.5) > 0.0 ? 0.0 : min(a,b);
    return sqrt( (d2+q.z*q.z)/m2 ) * sign(max(q.z,-p.y));
}
```

Código base de Ray Marching

```
float sdScene(vec3 p) {
    float pyramid = sdPyramid(p, 1.0);
    float plane = sdPlane(p, vec3(0,1,0), 0.0);
    return min(plane, pyramid);
}

float rayMarch(vec3 ro, vec3 rd, float start, float end)
{
    float depth = start;
    for (int i = 0; i < 255; i++) {
        vec3 p = ro + depth * rd;
        float d = sdScene(p);
        depth += d;
        if (d < 0.001 || depth > end) break;
    }
    return depth;
}

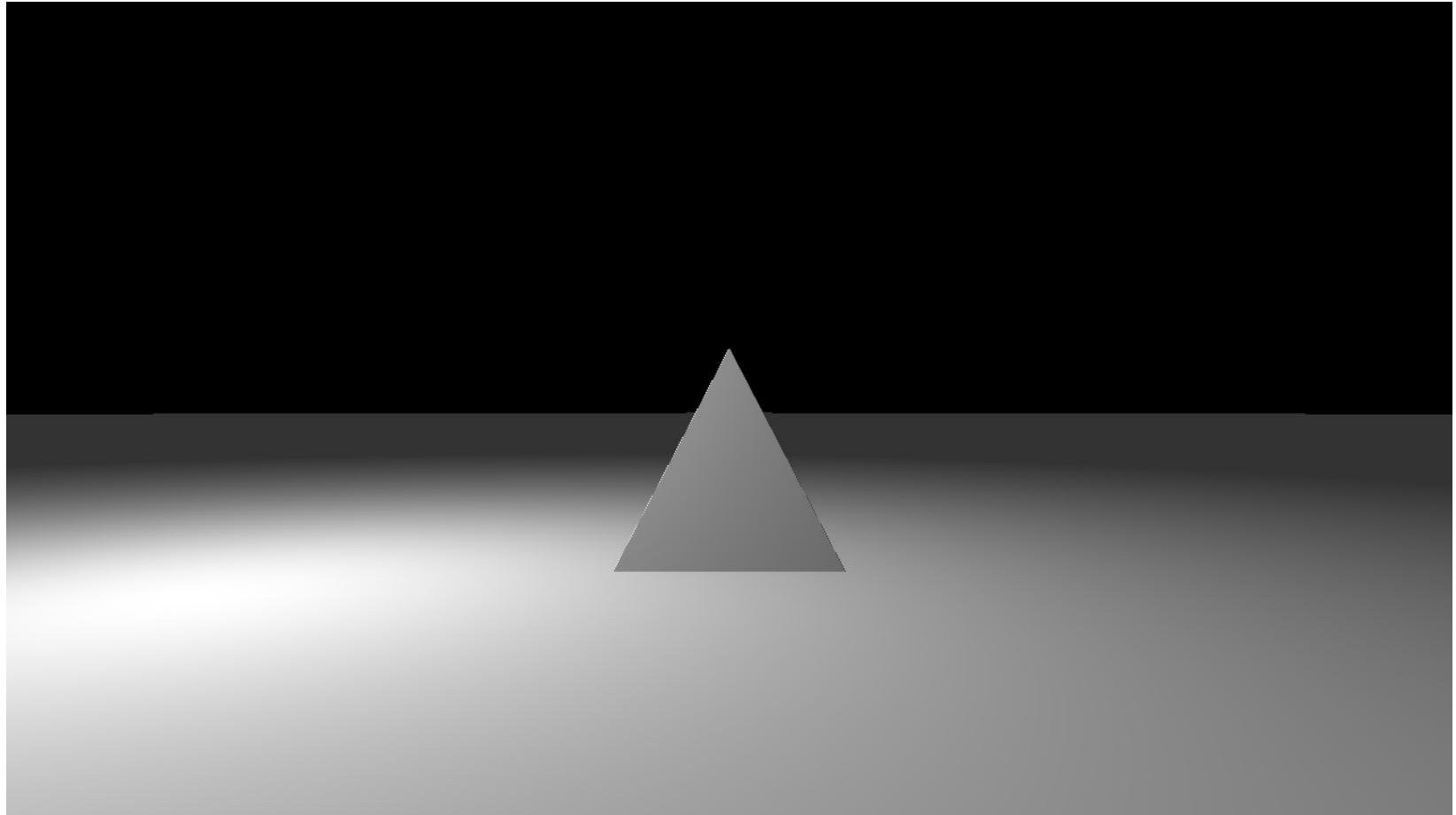
vec3 calcNormal(vec3 p) {
    vec2 e = vec2(1.0, -1.0) * 0.0005; // epsilon
    return normalize(
        e.xyy * sdScene(p + e.xyy) +
        e.yyx * sdScene(p + e.yyx) +
        e.yxy * sdScene(p + e.yxy) +
        e.hxx * sdScene(p + e.hxx));
}
```

```
void mainImage( out vec4 fragColor, in vec2 fragCoord ) {
    vec2 uv = (fragCoord-
    .5*iResolution.xy)/iResolution.y;
    vec3 col = vec3(0);
    vec3 ro = vec3(0, 0.7, 4);
    vec3 rd = normalize(vec3(uv, -1));
    float d = rayMarch(ro, rd, 0.01, 100.0);

    if (d > 100.0) col = vec3(0.0);
    else {
        vec3 p = ro + rd * d;
        vec3 normal = calcNormal(p);
        vec3 lightPos = vec3(-2, 2, 1.0);
        vec3 lightDir = normalize(lightPos - p);
        float ambient = 0.2;
        float scalar = dot(normal, lightDir);
        float diffuse = clamp(scalar, ambient, 1.);
        col = vec3(diffuse);
    }

    fragColor = vec4(col, 1.0);
}
```

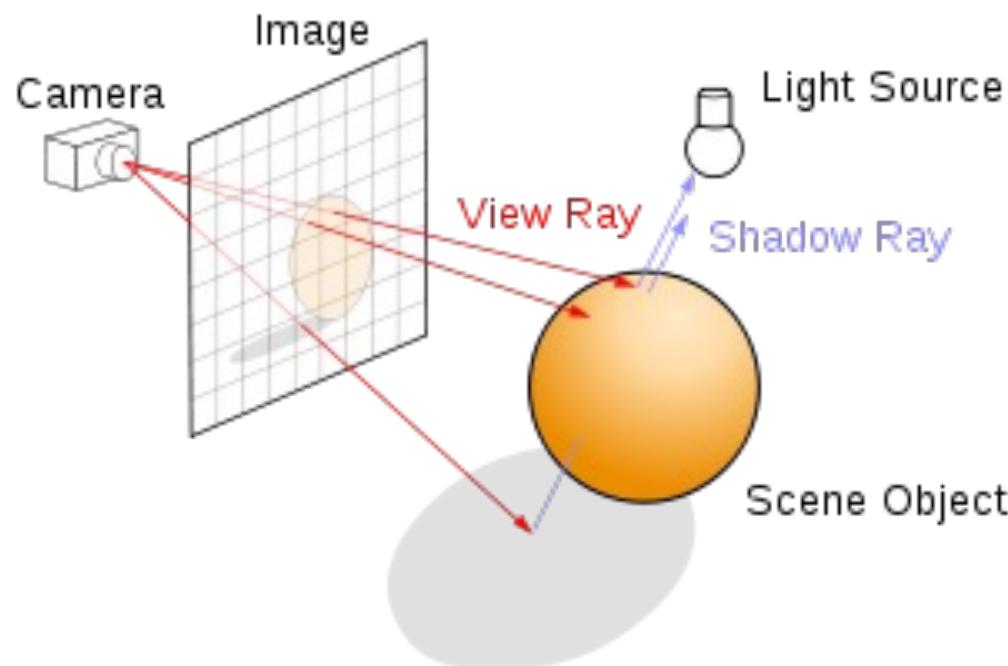
Cenário com Plano e uma Pirâmide



Cálculo de sombra

A proposta da sombra é a partir do ponto encontrado na cena, buscar a(s) fonte(s) de iluminação (Shadow Rays).

Se uma fonte for encontrada atenuaremos a luminância daquele ponto.

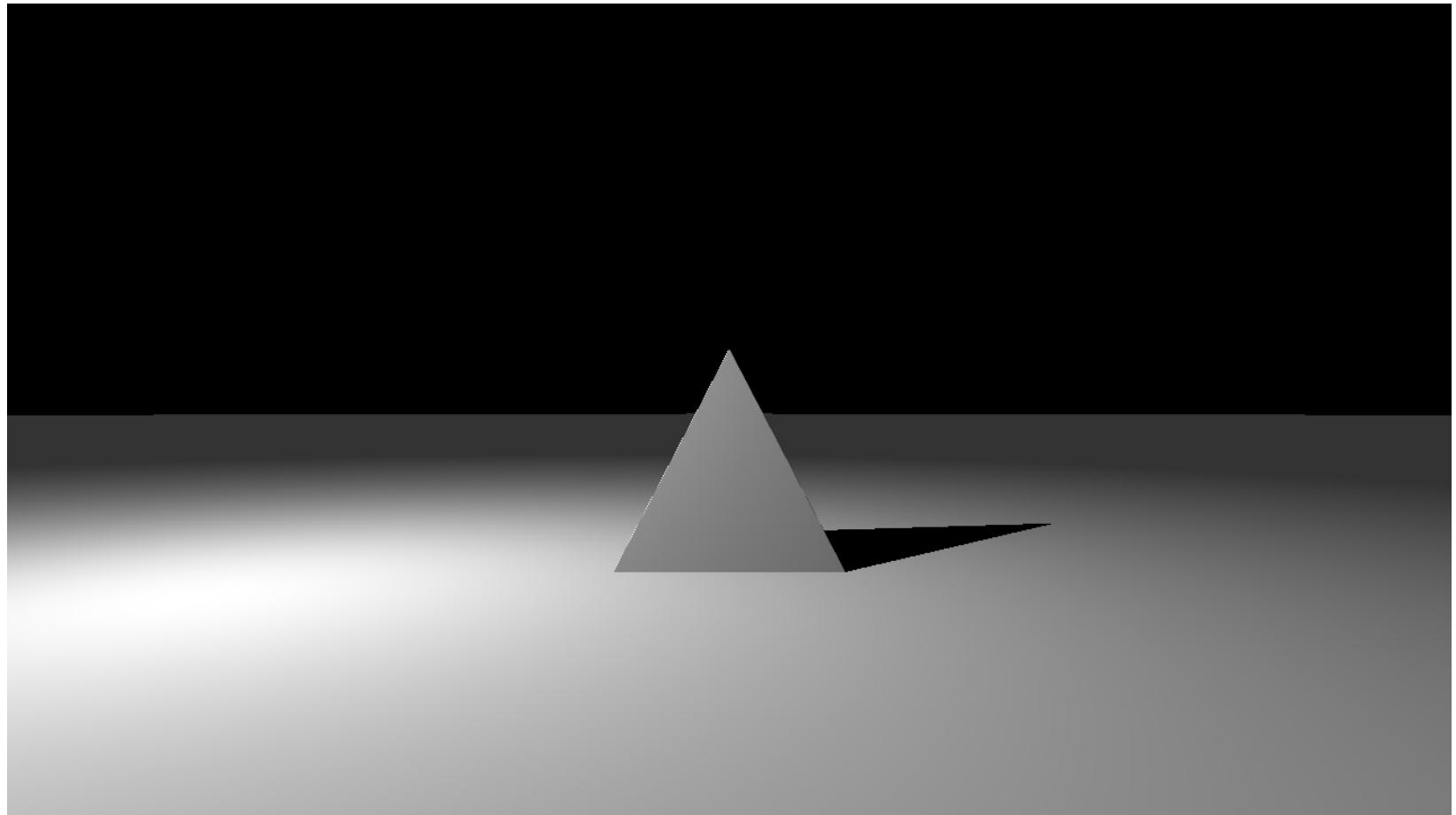


Cálculo de sombra

```
float shadow( in vec3 ro, in vec3 rd, float mint, float maxt ) {  
    float t = mint;  
    for( int i=0; i<256 && t<maxt; i++ ){  
        float h = sdScene(ro + rd*t);  
        if( h<0.001 )  
            return 0.0;  
        t += h;  
    }  
    return 1.0;  
}
```

```
...  
    float occ = shadow(p, lightDir, 0.01, 100.0 );  
    col = vec3(difuse*occ);  
}  
  
fragColor = vec4(col, 1.0);  
}
```

Resultado com Sombra



Implementando Penumbra

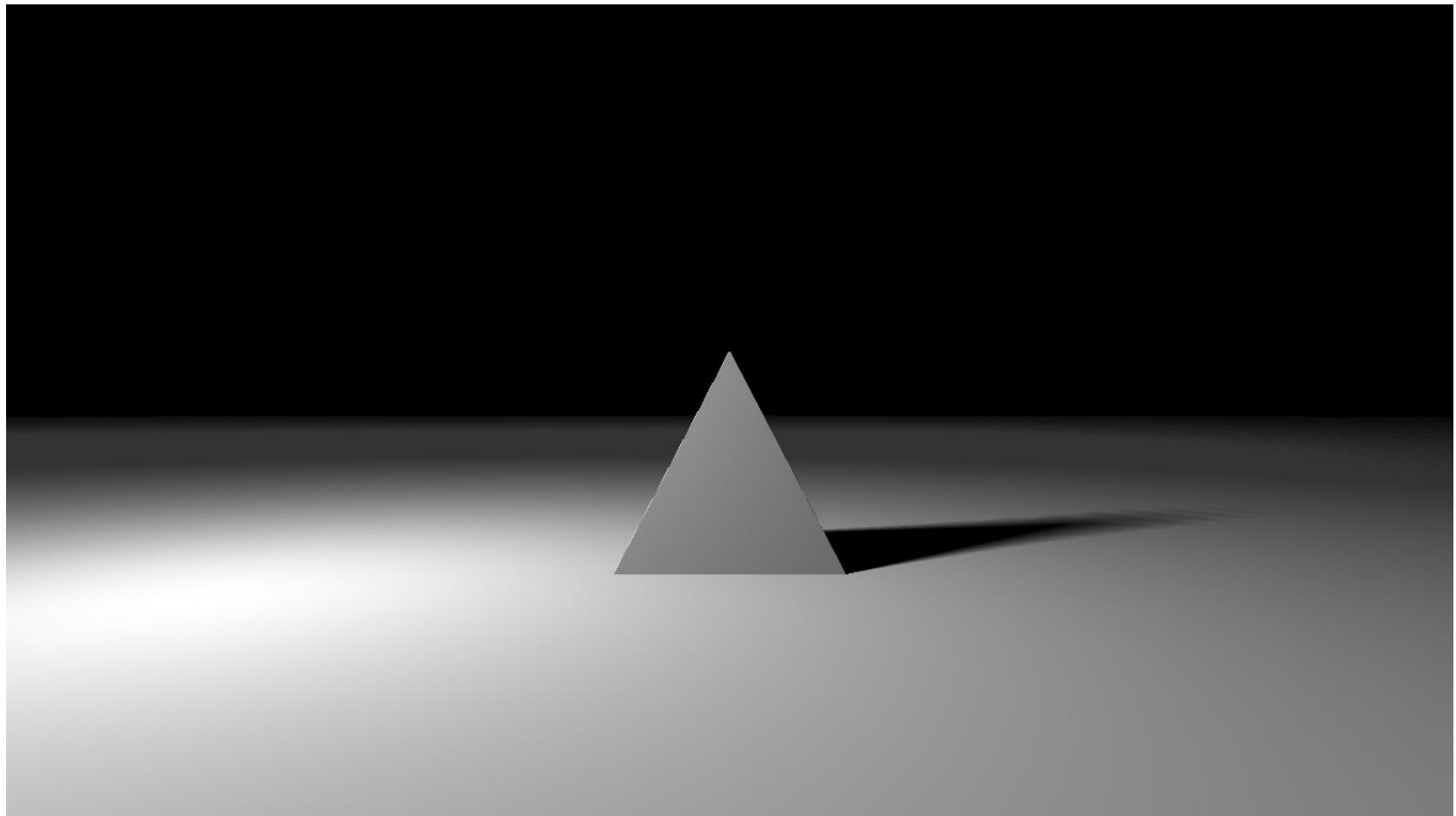
A penumbra é um efeito causado por uma fonte de luz não pontual.

Podemos simular esse efeito na sombra gerada. Para isso o raio de sombra (shadow ray) vai verificar se não passou muito próximo de algum superfície.

Código de Soft Shadow para Ray Marching

```
float softshadow( in vec3 ro, in vec3 rd, float mint, float maxt, float k ) {  
    float res = 1.0;  
    float t = mint;  
    for( int i=0; i<256 && t<maxt; i++ ) {  
        float h = sdScene(ro + rd*t);  
        if( h<0.001 )  
            return 0.0;  
        res = min( res, k*h/t );  
        t += h;  
    }  
    return res;  
}  
  
...  
float occ = softshadow(p, lightDir, 0.01, 100.0, 14.0 );  
...
```

Resultado com sombra



Environment Map no Shadertoy

<https://www.shadertoy.com/view/ll33zS>



Computação Gráfica

Luciano Pereira Soares
[<lpsoares@insper.edu.br>](mailto:lpsoares@insper.edu.br)