PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
DEPARTAMENTO DE COMPUTAÇÃO
COMPUTAÇÃO GRÁFICA CMP 1170 – 2019/1
PROF. MSC. GUSTAVO VINHAL

Aula 15
Realismo Visual e Iluminação (Continuação)

PUC GOIÁS

Iluminação

- Iluminação é um aspecto fundamental em qualquer composição.
    - Responsável por permitir a visualização e realismo de cenas e objetos.
- Luz é uma forma de radiação eletromagnética que se desloca em linha reta, transportada por uma onda que determina suas características físicas pelo comprimento de onda e frequência.
- Características:
    - Sombreamento;
    - Sombras;
    - Ray Tracing.

PUC GOIÁS

Sombreamento (*Shading*)

- Sombreamento é diferente de sombras!
- Sombreamento é o efeito da luz sobre superfícies e seus materiais; o obscurecimento de superfícies em função de sua posição, orientação e características da luz.
- Modelo de sombreamento é diferente de modelo de iluminação:
  - No modelo de sombreamento, cada ponto da superfície é iluminado a fim de obter o sombreamento.
- Modelos mais utilizados:
  - Modelo de sombreamento constante;
  - Sombreamento de Gouraud;
  - Modelo de Phong.

PUC GOIÁS

Sombreamento

Modelo de Sombreamento Constante

- Também chamada de *flat shading, faceted shading ou constant shading*;
- Neste modelo, aplica-se o cálculo da componente de luz refletida apenas uma vez por superfície **plana**;
- O valor encontrado é utilizado para preenchimento de toda a superfície;
- Técnica mais utilizada em primitivas poligonais.

PUC GOIÁS

Sombreamento

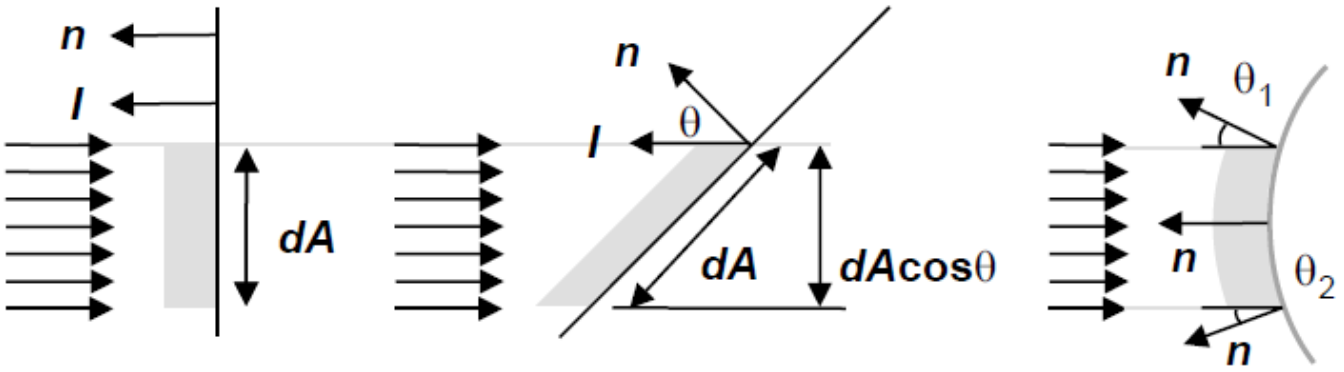
Modelo de Sombreamento Constante

- Somente é aceitável se obedecer as seguintes regras:

1. A fonte de luz localizar no infinito, fazendo com que o ângulo de incidência seja igual em raio de luz;

2. O observador localizar no infinito, fazendo com que os raios que atingem o observador possuam o mesmo ângulo;

3. As superfícies realmente utilizem faces planas e não aproximados.



PUC GOIÁS

Sombreamento

Modelo de Sombreamento Constante

- Se não obedecer as regras anteriores, este modelo não deve ser aplicado pois gerará resultados insatisfatórios;
- Este modelo é o mais simples, pois o cálculo de iluminação é calculado apenas uma vez para cada polígono dos objetos em cena.
    - **Consequência**: realismo fraco.

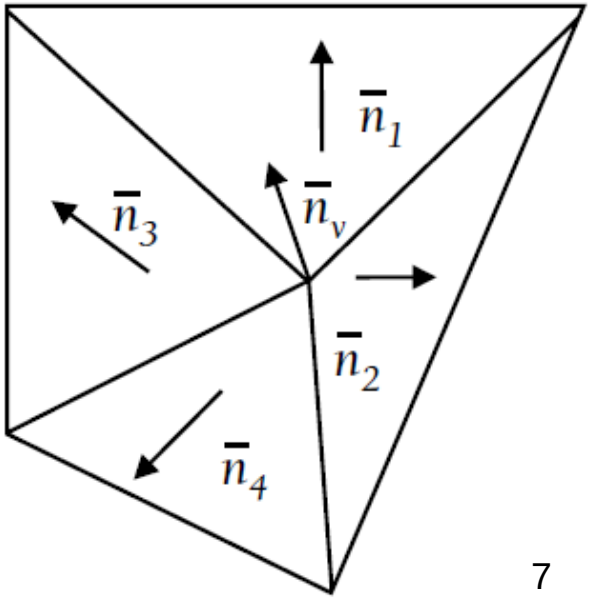
PUC GOIÁS

Sombreamento

Sombreamento de Gouraud

- Uma forma de melhorar o modelo de sombreamento constante é encontrar a normal da superfície em cada ponto do polígono e aplicar o modelo de iluminação desejado;
    - **Problema:** alto custo computacional;

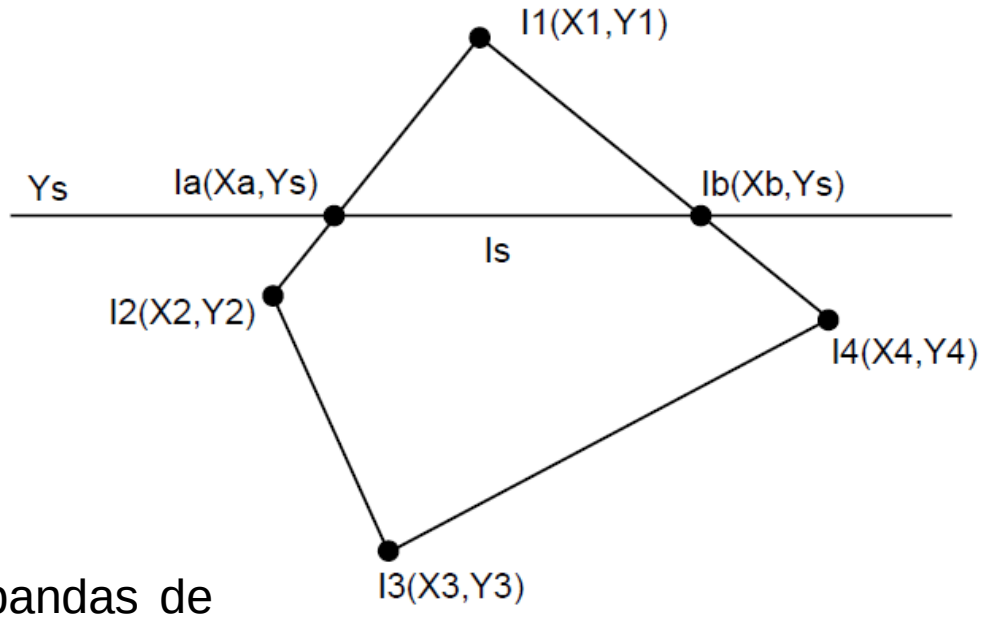
- O modelo de sombreamento de Gouraud consiste em aplicar a iluminação em um subconjunto de pontos da superfície e, em seguida, **interpolar** a intensidade nos pontos restantes na superfície;



PUC GOIÁS

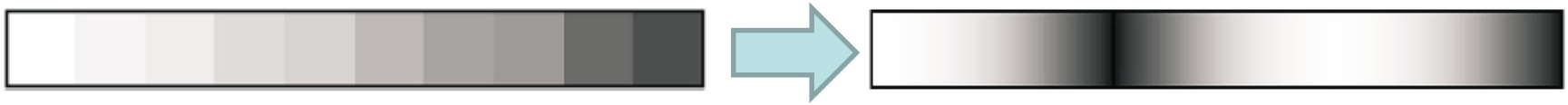
Sombreamento

Sombreamento de Gouraud



- Também conhecido como *intensity interpolation shading* ou *color interpolation shading*;

- Este modelo diminui o efeito das bandas de Mach;



PUC GOIÁS

Sombreamento

Sombreamento de Gouraud

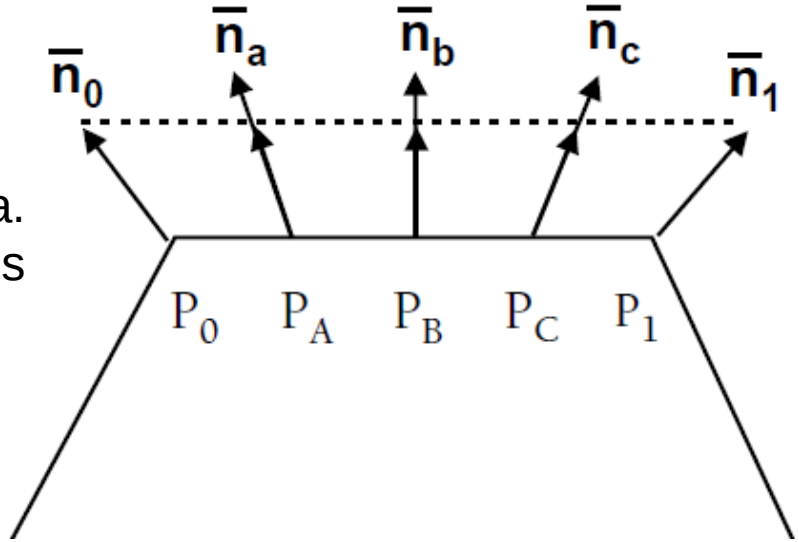
- Essa técnica se mostra ineficiente em reflexões especulares, quando essas se apresentam distantes dos pontos selecionados.
    - Neste caso, a interpolação ignora a reflexão e esta desaparece (ou tem o efeito diminuído).
- Lembrando:
    - Reflexão especular é responsável pela sensação de brilho no objeto;
    - Reflete toda a luz incidente;

PUC GOIÁS

Sombreamento

Modelo de Phong

- Este modelo é similar ao anterior (Gouraud), porém ao invés de interpolar a intensidade da luz refletida, o modelo de Phong interpola a variação do ângulo de incidência;
    - **Vantagem:** os pontos de reflexão especular afastados podem ser determinados;
- A normal de cada vértice é encontrada. Interpola-se as normais, copiando os valores de intensidade.

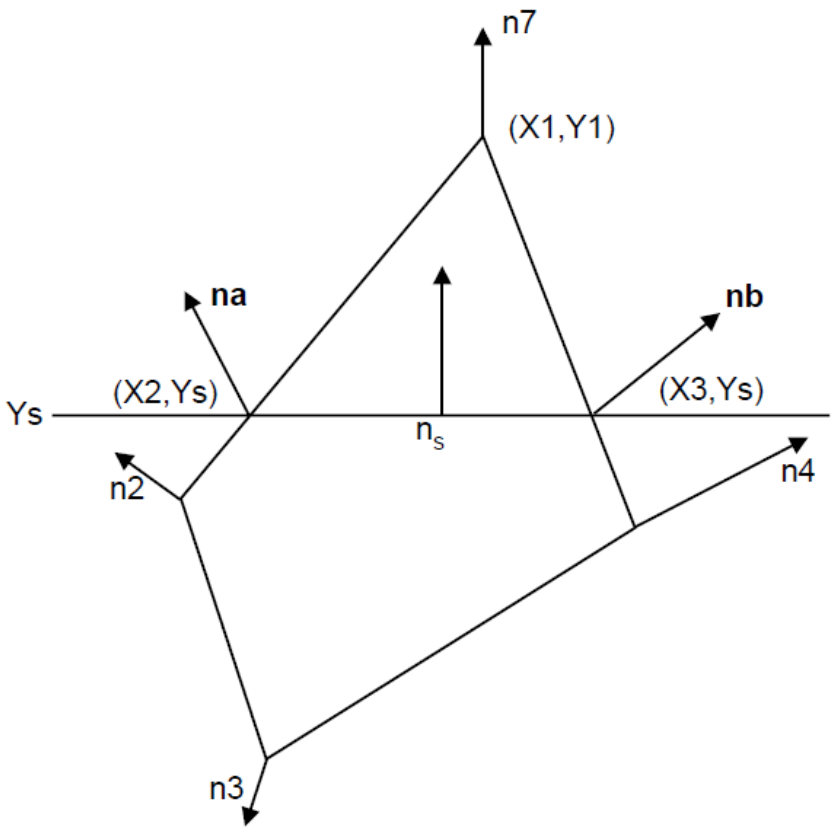


PUC GOIÁS

Sombreamento

Modelo de Phong

- Também conhecido como *normal-vector interpolation shading*;



PUC GOIÁS

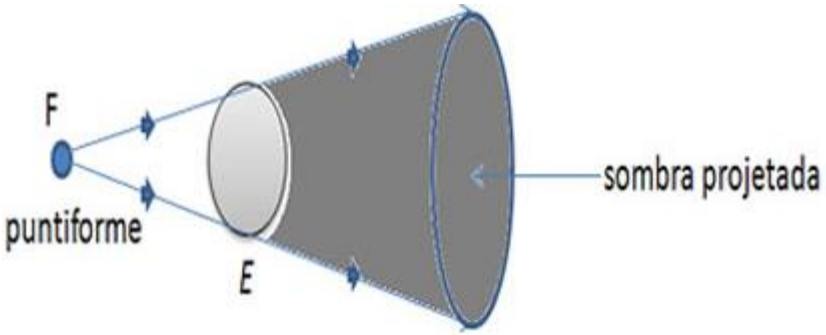
Sombras

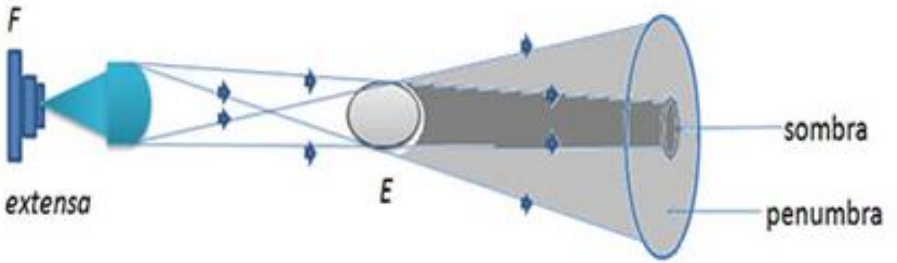
- Sombras é de fundamental importância no realismo das cenas, pois evitam que os objetos pareçam estar flutuando no ar;

PUC GOIÁS

Sombras

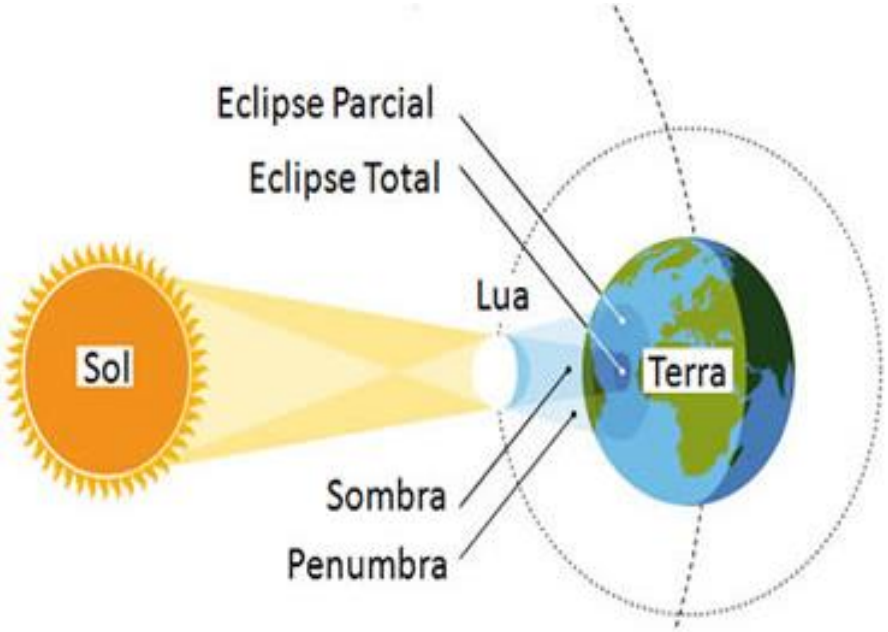
- A análise realística das sombras se caracteriza por duas regiões:
  - **Região de sombra:** região onde a intensidade luminosa é nula;
  - **Região de penumbra:** região onde a intensidade luminosa varia de zero até a intensidade da luz ambiente.





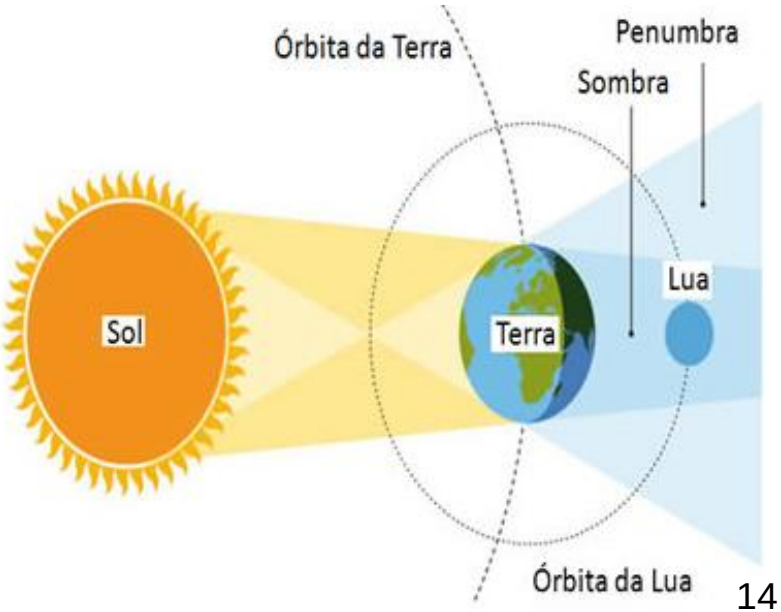
PUC GOIÁS

Sombras



Eclipse Solar

Eclipse Lunar



PUC GOIÁS

Sombras

- *Hardwares* mais sofisticados conseguem executar algoritmos mais elaborados para geração de sombra realistas. Os mais comuns são:
  - Volume de sombra;
  - Sombra projetada.

PUC GOIÁS

Sombras

Algoritmo Volume de Sombra

- É um algoritmo geométrico;
- Consiste em definir a região da sombra a partir de conjunto de pontos do espaço da imagem que não são visíveis (quando o observador é colocado na posição da fonte luminosa);
- Pode ser vista como uma pirâmide, onde o ápice é a origem da fonte da luz.

PUC GOIÁS

Sombras

Algoritmo Sombra Projetada

- Também é um algoritmo geométrico;
- Consiste em modelar a região da sombra a partir da projeção do objeto cuja sombra se deseja definir.
- Assume-se que o ponto de vista é a origem da fonte luminosa e o plano de projeção é o lugar onde a sombra irá ser formada.
- Para uma única superfície plana, este algoritmo funciona bem. Porém, se torna complexo quando o número de superfícies planas aumenta (escada, por exemplo).

PUC GOIÁS

Ray Tracing

- Ray tracing surgiu na década de 60. Em 1968 ele foi desenvolvido como algoritmo para simulação de trajetórias de projéteis balísticos e partículas nucleares;

- Apple foi a primeira a apresenta-lo como uma ferramenta para cálculo de sombras em Computação Gráfica;

- Antigamente, não era uma técnica muito utilizada devido a alta complexidade computacional (tempo de execução);

    - 1980: Criação de sombras, reflexões, transparências e refrações;
    - 1984: Adição de penumbras, *motion blur,* entre outros.

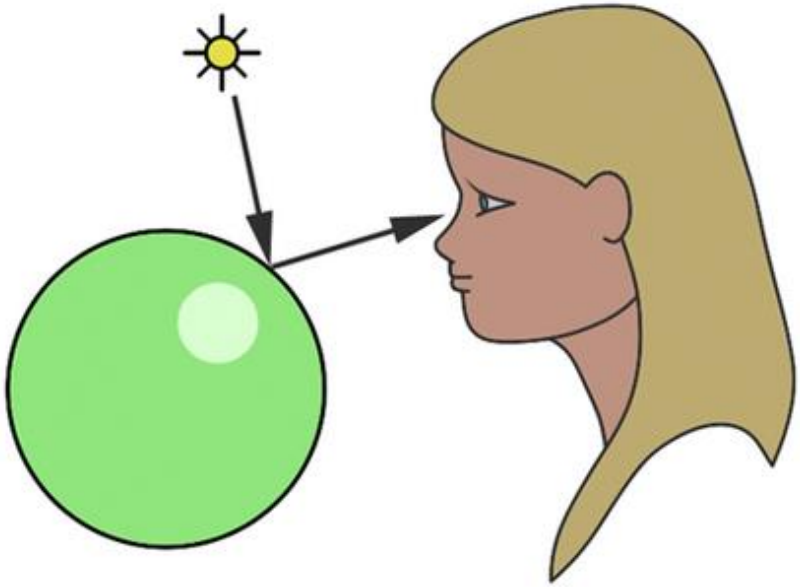
- Com *hardware* mais moderno, atualmente é uma técnica computacional muito utilizada para trazer realismo a cenas de tempo real.

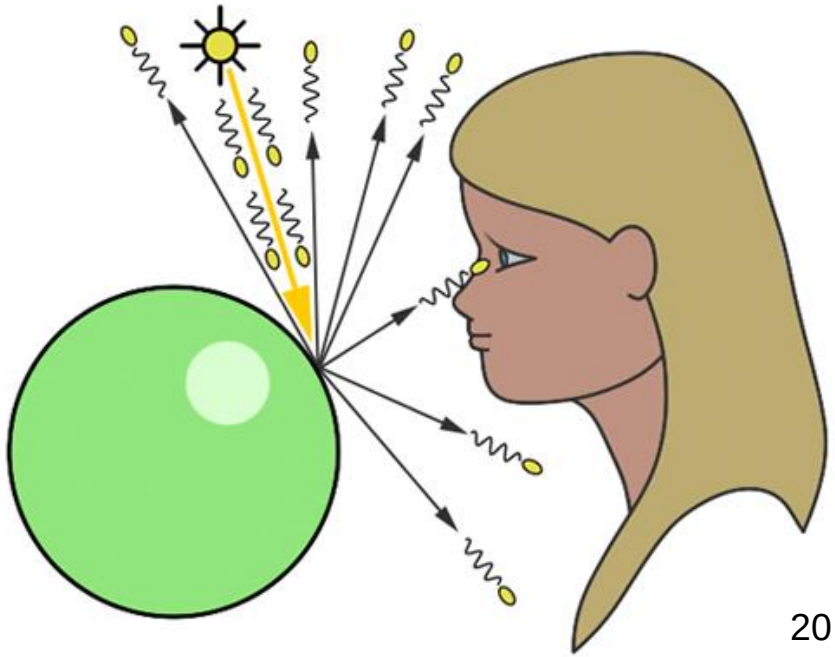
PUC GOIÁS

Ray Tracing

PUC GOIÁS

Ray Tracing

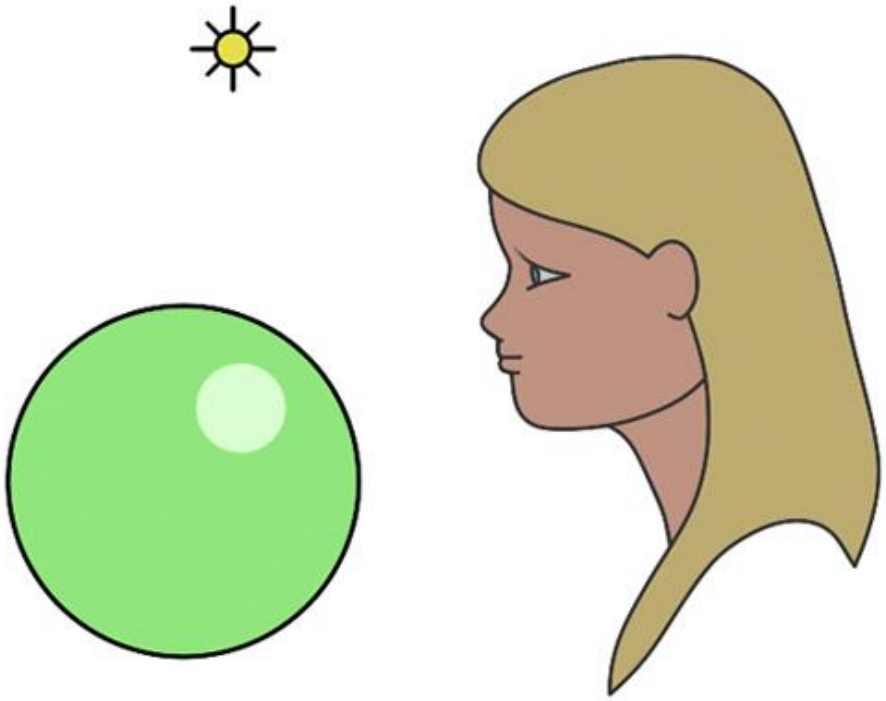




PUC GOIÁS

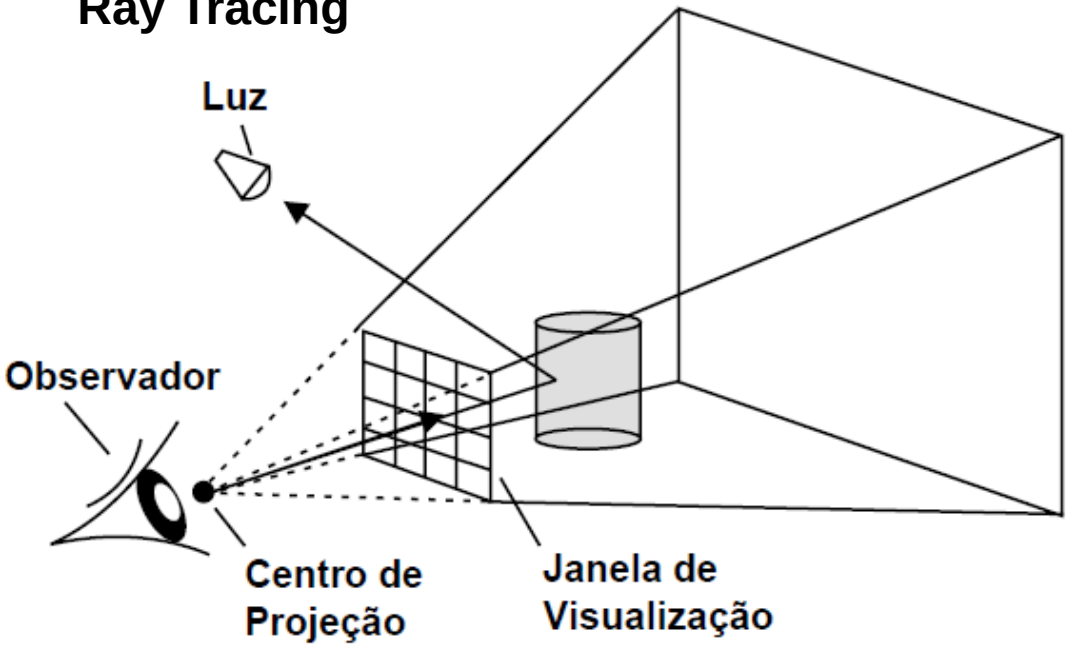
Ray Tracing

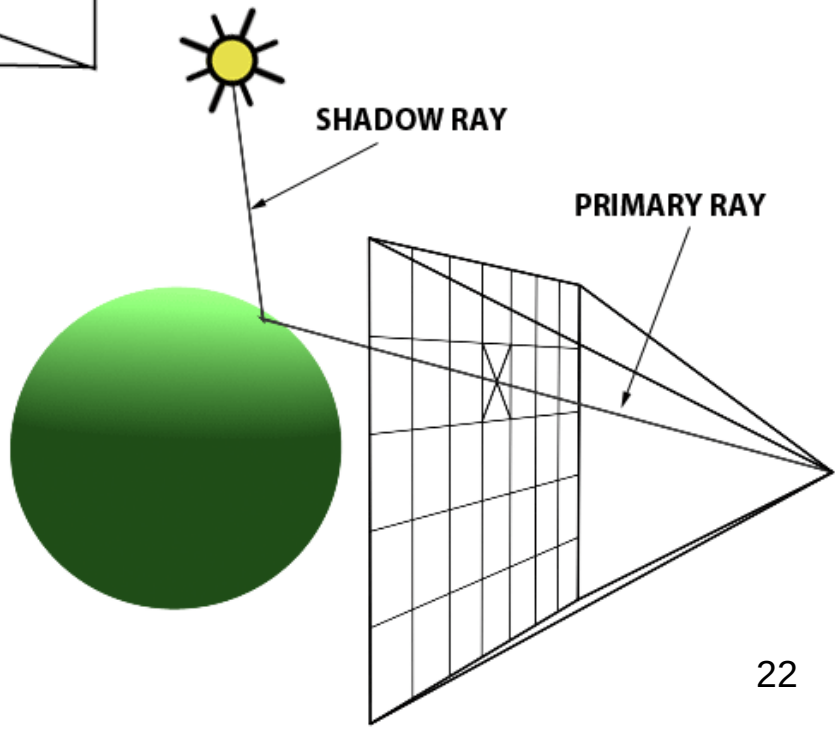
Ray tracing funciona de maneira inversa ao processo de visualização.



PUC GOIÁS

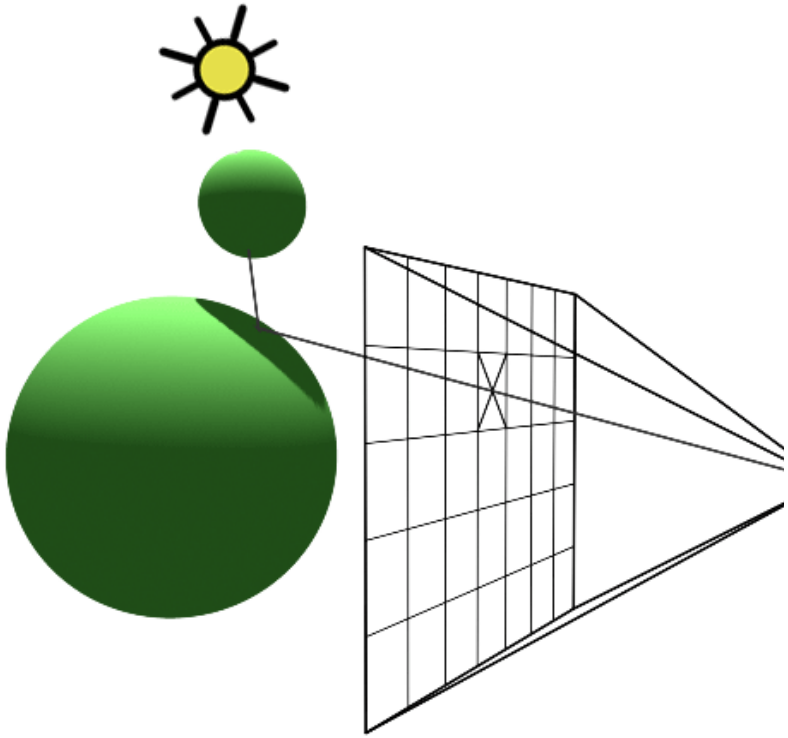
Ray Tracing

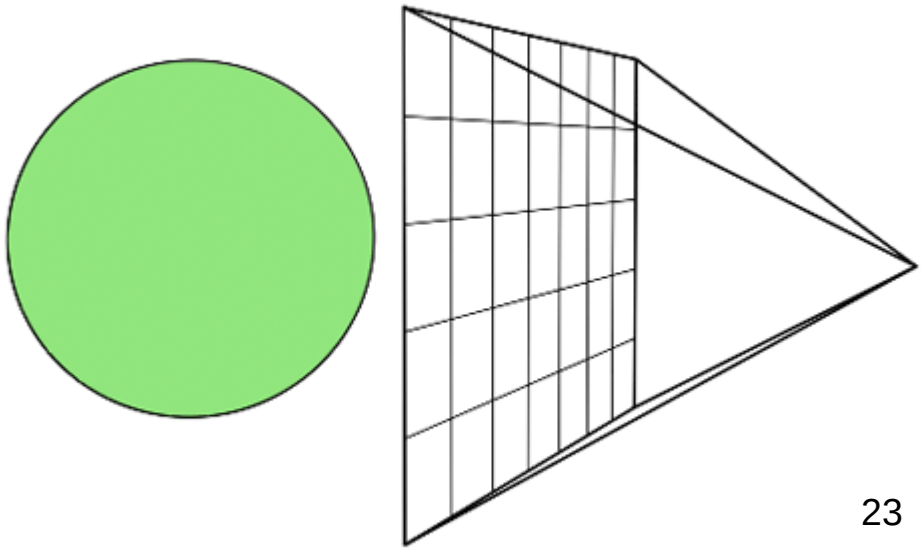




PUC GOIÁS

Ray Tracing





PUC GOIÁS

Ray Tracing

- Algoritmo:

1. Trace um “raio” a partir do observador até a cena a ser representada através de um pixel na 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. Use esse valor para determinar a cor do pixel e se há sombras.

Ray Tracing

```
Considerando um centro de projeção, no plano de visão
Para (cada linha horizontal de varredura da imagem – scan line)
          { Para (cada pixel da linha de varredura)
                    { determinar raio que vai do centro de projeção ao pixel
                              Para (cada objeto da cena)
                        { Se (objeto for interceptado pelo raio &&
                                    é a interseção mais próxima até agora)
                        registrar interseção e o objeto interceptado
                        }
                    atribuir ao pixel a cor do objeto da interseção mais próxima
                    }
          }
```

Ray Tracing

```
001 for (int j = 0; j < imageHeight; ++j) {
002     for (int i = 0; i < imageWidth; ++i) {
003         // compute primary ray direction
004         Ray primRay;
005         computePrimRay(i, j, &primRay);
006         // shoot prim ray in the scene and search for intersection
007         Point pHit;
008         Normal nHit;
009         float minDist = INFINITY;
010         Object object = NULL;
011         for (int k = 0; k < objects.size(); ++k) {
012             if (Intersect(objects[k], primRay, &pHit, &nHit)) {
013                 float distance = Distance(eyePosition, pHit);
014                 if (distance < minDistance) {
015                     object = objects[k];
016                     minDistance = distance; // update min distance
017                 }
018             }
019         }
020         if (object != NULL) {
021             // compute illumination
022             Ray shadowRay;
023             shadowRay.direction = lightPosition - pHit;
024             bool isShadow = false;
025             for (int k = 0; k < objects.size(); ++k) {
026                 if (Intersect(objects[k], shadowRay)) {
027                     isInShadow = true;
028                     break;
029                 }
030             }
031         }
```

PUC GOIÁS

Ray Tracing

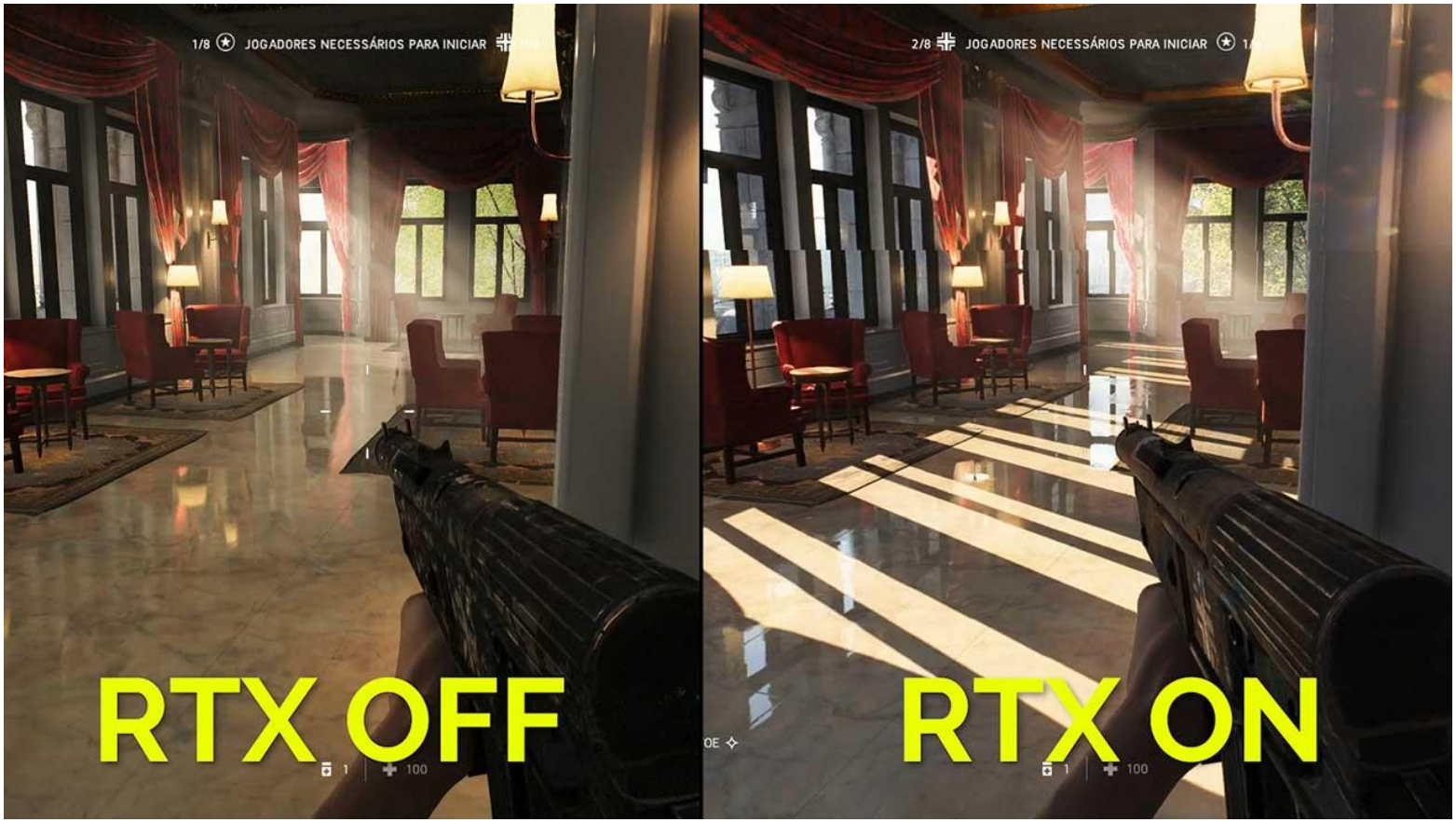
```
020         if (object != NULL) {
021             // compute illumination
022             Ray shadowRay;
023             shadowRay.direction = lightPosition - pHit;
024             bool isShadow = false;
025             for (int k = 0; k < objects.size(); ++k) {
026                 if (Intersect(objects[k], shadowRay)) {
027                     isInShadow = true;
028                     break;
029                 }
030             }
031         }
032         if (!isInShadow)
033             pixels[i][j] = object->color * light.brightness;
034         else
035             pixels[i][j] = 0;
036     }
037 }
```

PUC GOIÁS

Ray Tracing - Exemplos

PUC GOIÁS

Ray Tracing - Exemplos



PUC GOIÁS

Ray Tracing - Exemplos

PUC GOIÁS

REFERÊNCIAS BIBLIOGRÁFICAS:

AZEVEDO, Eduardo; CONCI, Aura. **Computação gráfica:** teoria e prática. Rio de Janeiro: Campus, 2003.