Universidade Federal de Pernambuco
Centro de Informática
Processamento Gráfico - 2021.2
Especificação do Projeto Ray-Tracing

1 Introdução

O projeto *Ray-Tracing* será organizado em três entregas: as duas primeiras valem ponto extra e possuem um escopo mais básico (usam apenas *Ray-Casting*), mas que servem como um excelente arcabouço para o *Ray-Tracing* de Whitted exigido na terceira entrega, esta última, por sua vez, valendo nota. Mais detalhes sobre pontuação, prazos, e correção encontram-se no Classroom da disciplina.

2 Primeira Entrega

É o mínimo de código necessário para conseguir obter uma cena visível. Basta colorir os objetos de maneira uniforme, contanto que os contornos estejam corretos. Os componentes que já devem estar funcionando na primeira entrega são: a câmera, o *Ray-Casting* e os cálculos de interseção de raios com objetos.

2.1 *Ray-Casting*

Para determinar a superfície visível, raios primários são disparados em direção à cena, atravessando os pixels na *tela* (uma malha quadriculada que existe no mesmo espaço euclidiano que os objetos da cena, com o número de linhas e colunas da imagem final) e assumindo a cor do objeto mais próximo atingido (Figura 1). A resolução da *tela* é dada na entrada como `v_res` $\times$ `h_res` (linhas, colunas), e os pixels na *tela* possuem dimensão $s \times s$ (Figura 2).

2.2 Interseções

Apenas dois tipos de objetos precisam ser renderizados: planos e esferas. Na entrada, as esferas são definidas pelo centro O e raio r, e os planos por um ponto amostral P (contido nele) e um vetor normal $\mathbf{n}$. Ambos os objetos também possuem como atributo a cor `C`, uma tripla *RGB* na escala discreta de `0` a `255`.

Deve ser possível renderizar normalmente a cena, mesmo que haja dois objetos com interseção.

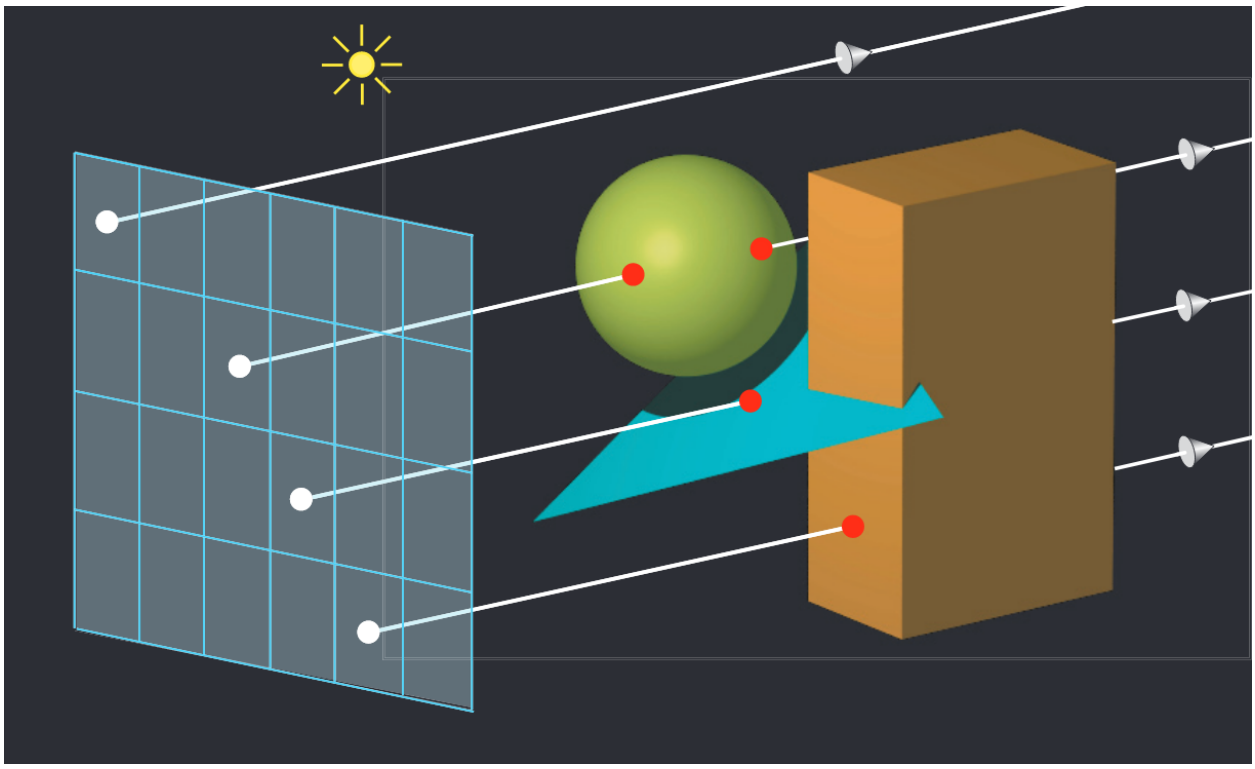


Figura 1: *Ray Casting*.

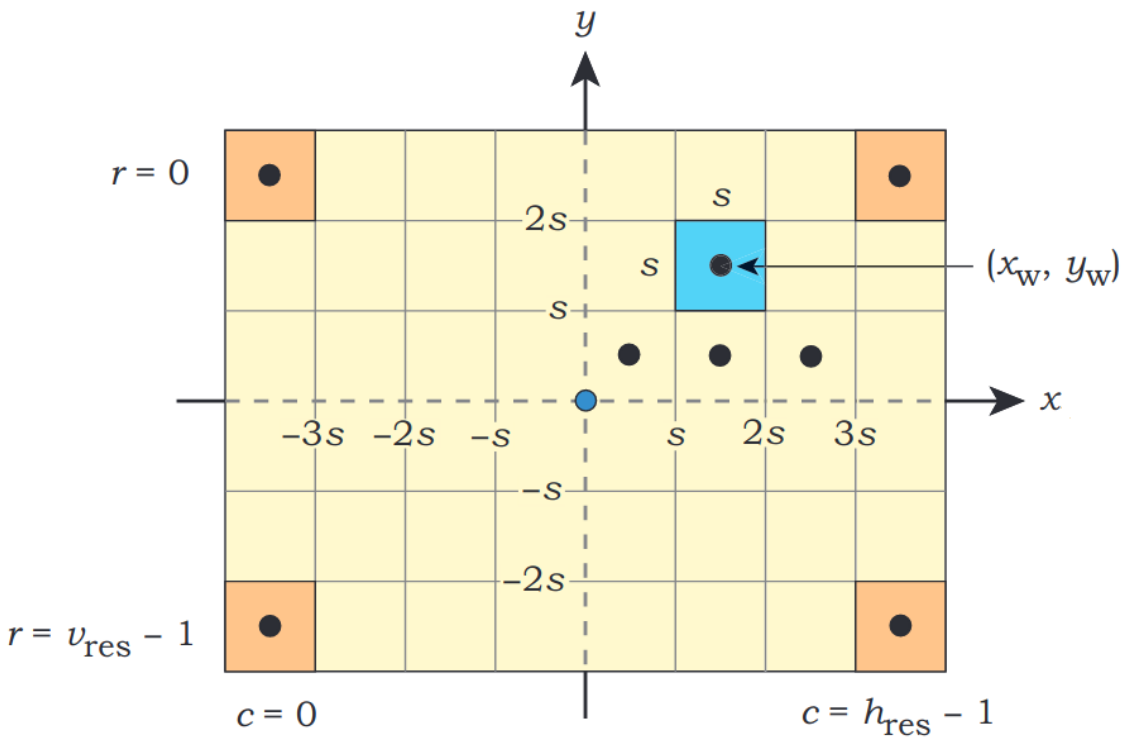


Figura 2: A *tela*.

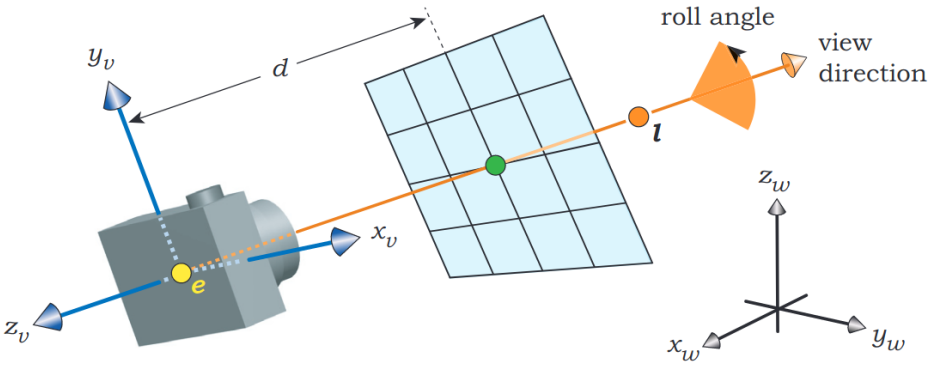


Figura 3: Câmera virtual.

2.3 Câmera

A câmera virtual (Figura 3) deve proporcionar uma vista da cena em perspectiva, através dos seguintes parâmetros: o foco E (origem do sistema de coordenadas e dos raios primários), a mira L (para onde a câmera aponta), a distância d entre o foco E e a *tela* (regula o *zoom*) e um vetor **up** apontando "para cima"(controla a rotação da câmera em torno do eixo *view direction*).

Assim, $\overrightarrow{EL}$ corresponde à direção do vetor normal à *tela*, enquanto que o parâmetro d fixa a sua posição no espaço, i.e., resolve o termo independente da equação cartesiana do plano no $\mathbb{R}^3$. O vetor **up**, por sua vez, auxilia na construção da base ortonormal $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ do sistema de coordenadas da câmera (Figura 4):

$$\mathbf{w} = \frac{E - L}{\|E - L\|}, \quad \mathbf{u} = \frac{\mathbf{up} \times \mathbf{w}}{\|\mathbf{up} \times \mathbf{w}\|}, \quad \mathbf{v} = \mathbf{w} \times \mathbf{u}$$

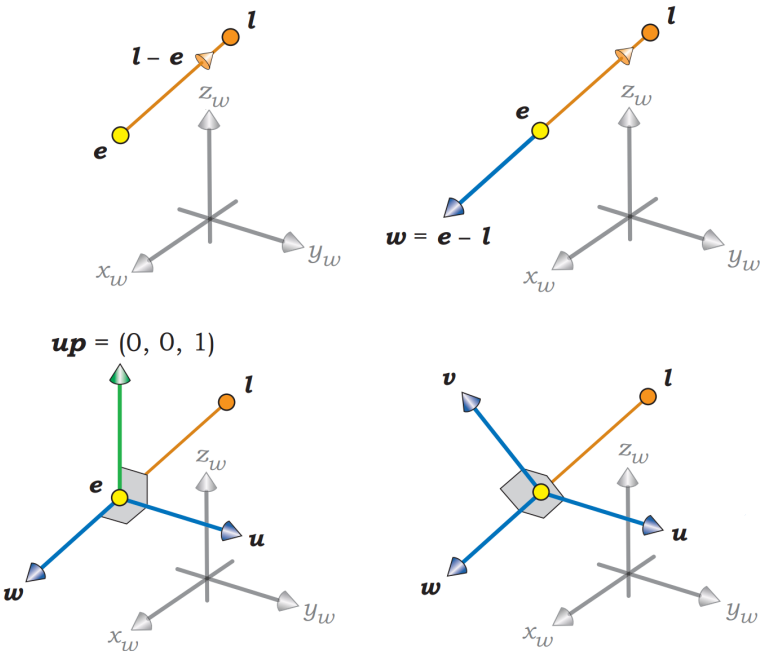


Figura 4: Base ortonormal. O sistema de coordenadas está orientado pela regra da mão esquerda.

Note que $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ são montados de forma que o vetor **up** possa ser escolhido de forma totalmente arbitrária: não é necessário que **up** e $\overrightarrow{EL}$ sejam perpendiculares entre si, nem que **up** seja unitário. Mas, por simplicidade, todos os testes usarão o valor *default* $\mathbf{up} = (0, 0, 1)$.

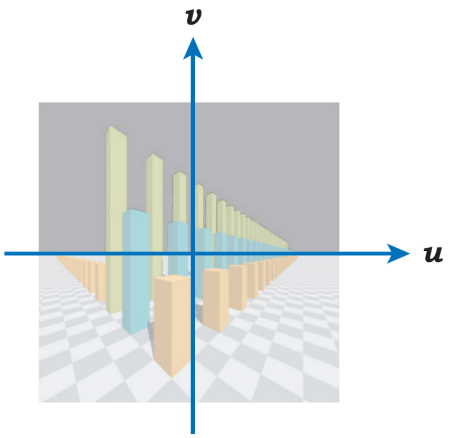


Figura 5: Do ponto de vista da câmera, que mira no sentido contrário do eixo **w**, **u** está na horizontal, e **v** na vertical.

Uma vez definido o sistema de coordenadas da câmera, fica fácil expressar a posição do centro de cada quadrado na *tela* (para onde os raios primários devem apontar). Seja $Q_{\mathtt{r},\mathtt{c}}$ o centro do pixel na linha `r` e na coluna `c`, em coordenadas mundiais. Então:

$$Q_{\mathtt{r},\mathtt{c}} = E + x_{\mathtt{r},\mathtt{c}}\mathbf{u} + y_{\mathtt{r},\mathtt{c}}\mathbf{v} - d\mathbf{w}, \quad x_{\mathtt{r},\mathtt{c}} = s(\mathtt{c} - \mathtt{h\_res}/2 + 0.5), \quad y_{\mathtt{r},\mathtt{c}} = -s(\mathtt{r} - \mathtt{v\_res}/2 + 0.5)$$

2.4 Entrada

As cinco primeiras linhas do arquivo de entrada contém os parâmetros da câmera, da *tela* e da imagem:

```
v_res h_res
s   d
E_x   E_y   E_z
L_x   L_y   L_z
up_x   up_y   up_z
```

Na linha seguinte, a cor `B` do plano de fundo. Essa é a cor que o pixel deve assumir no caso em que o raio primário não atinge nenhum objeto.

```
B.r  B.g  B.b
```

O restante da entrada descreve os componentes da cena. A próxima linha contém o número de objetos `k_obj`:

```
k_obj
```

Seguem `k_obj` linhas. Se o próximo objeto for uma esfera, a linha segue o formato:

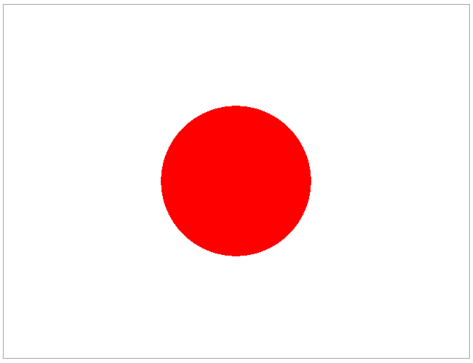
```
C.r  C.g  C.b  *  O_x  O_y  O_z  r
```

E se for um plano:

`C.r C.g C.b` / P_x P_y P_z $\mathbf{n_x}$ $\mathbf{n_y}$ $\mathbf{n_z}$

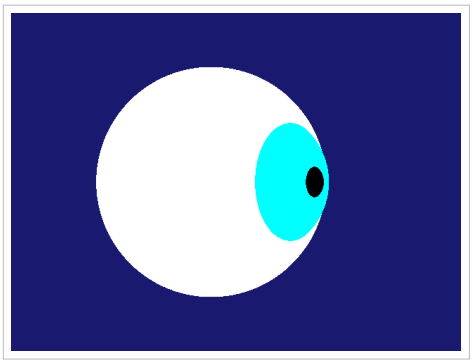
2.5 Testes

(1) `polonia.txt`

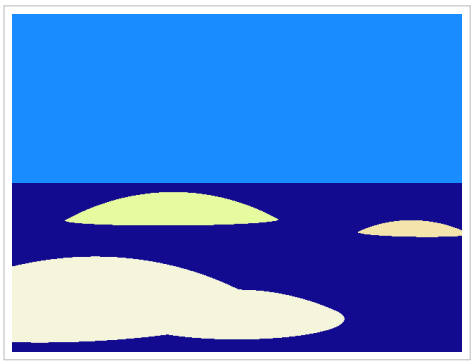


(2) `japao.txt`

(3) `canto.txt`



(4) `olho.txt`



(5) `ilhas.txt`

3 Segunda Entrega

Na segunda entrega, luzes e sombras são introduzidas na cena. Adotaremos o modelo de iluminação de Phong, considerando somente as componentes ambiental, difusa e especular. Isso implica que nenhuma função recursiva precisa ser implementada por enquanto.

3.1 Iluminação

Apenas dois tipos de luzes são requisitados: luz ambiente e fontes pontuais. Ambos os tipos possuem como atributo a cor `C` (na escala de `0` a `255`). Para a luz pontual é preciso definir também a localização L da fonte.

O material de cada objeto é determinado pelos seguintes parâmetros: a cor difusa `Cd` (na escala de `0` a `255`), o coeficiente ambiental k_a, o coeficiente de reflexão difusa k_d, o coeficiente especular k_s e o expoente de Phong η, com $0 \leq k_a, k_d, k_s \leq 1$, e $\eta > 0$.

Ainda que as componentes RGB das cores sejam representadas na entrada e na saída numa escala discreta de `0` a `255`, dentro do programa deve-se trabalhar com valores em ponto flutuante entre 0 a 1. Assim, a cada cor `C` na entrada, associamos um vetor corresponde $\mathbf{c}$ no $\mathbb{R}^3$:

$$\mathbf{c} = \frac{(\texttt{C.r}, \texttt{C.g}, \texttt{C.b})}{255}$$

3.2 Sombras

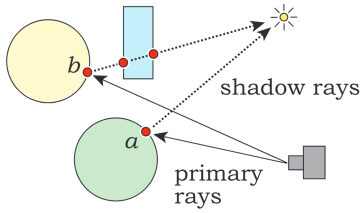


Figura 7: Raios de sombra e raios primários.

Uma sombra é uma parte da cena onde a iluminação que seria fornecida por uma fonte de luz encontra-se bloqueada por ao menos um objeto. Em termos de implementação, essa condição é verificada analisando a interseção de raios de sombra (partindo da superfície visível em direção às fontes de luz) com o restante da cena.

Seja P um ponto da cena atingido por um raio primário, e $\omega_\mathbf{o}$ e $\mathbf{n}$ vetores unitários na direção do observador e da normal, respectivamente (Figura 8). Considere o subconjunto $\{(\mathbf{c_1}, L_1), \ldots, (\mathbf{c_m}, L_m)\}$ das fontes de luz que não fazem sombra em P. A equação de iluminação para o ponto P é dada por:

$$\mathbf{c_P} = k_a \mathbf{c_d} * \mathbf{c_a} + \sum_{j=1}^{m} k_d \mathbf{c_d} * \max(0, \mathbf{n} \cdot \mathbf{l_j}) \mathbf{c_j} + \sum_{j=1}^{m} k_s \big[\max\left(0, \mathbf{r_j} \cdot \omega_\mathbf{0}\right) \big]^\eta \mathbf{c_j}$$

Na fórmula:

- $\mathbf{c_a}$ é a cor da luz ambiente;
- $\mathbf{l_j} = \frac{L_j - P}{\|L_j - P\|}$ é o vetor unitário apontando para a fonte de luz;
- $\mathbf{r_j} = 2(\mathbf{n_j} \cdot \mathbf{l_j})\mathbf{n_j} - \mathbf{l_j}$ é o vetor unitário na direção em que a luz é refletida.

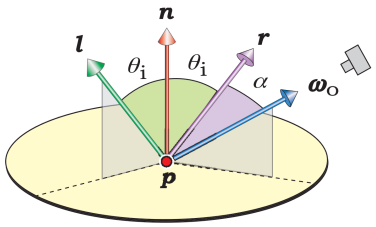
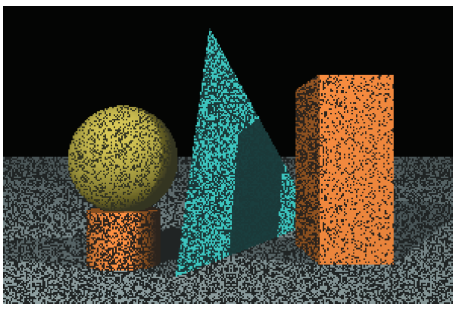
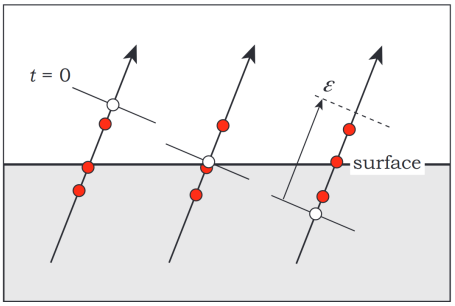


Figura 8: $\mathbf{l}$, $\mathbf{n}$, $\mathbf{r}$ e $\omega_\mathbf{o}$ precisam estar normalizados.

3.3 *Shadow acne*



(1)



(2)

Figura 9: *Shadow acne* com $\epsilon = 0$ (1). Os pontos de partida dos raios de sombra (brancos) e de chegada dos raios primários (vermelhos) podem na verdade estar fora ou dentro do objeto iluminado. (2)

Um *bug* que frequentemente aparece ao tentar renderizar sombras dessa forma é o *salt-and-pepper noise* (*a.k.a. shadow acne*). Esse problema é causado por erros de precisão de ponto flutuante, e ocorre quando um ponto da superfície visível acaba fazendo sombra sobre si mesmo (o raio de sombra intersecta a superfície do próprio objeto). Para tratá-lo, é preciso deslocar a origem dos raios de sombra por um fator ϵ (Figura 9) na direção da fonte de luz (10^{-5} foi utilizado na geração dos testes).

3.4 Entrada

O formato das sete primeiras linhas da entrada (parâmetros da câmera, cor de fundo e quantidade de objetos) permanece o mesmo da primeira entrega.

As diferenças começam nas `k_obj` linhas seguintes. Se o próximo objeto for uma esfera, a linha segue o formato:

`Cd.r` `Cd.g` `Cd.b` k_a k_d k_s η `*` O_x O_y O_z r

E se for um plano:

`Cd.r` `Cd.g` `Cd.b` k_a k_d k_s η `/` P_x P_y P_z $\mathbf{n_x}$ $\mathbf{n_y}$ $\mathbf{n_z}$

O restante da entrada descreve as luzes na cena. A próxima linha traz a cor da luz ambiente `Ca`:

`Ca.r` `Ca.g` `Ca.b`

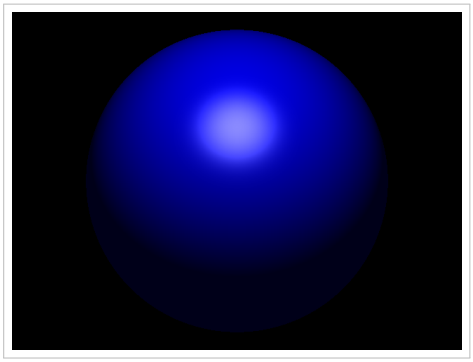
Seguida pelo número de fontes pontuais `k_pl`:

`k_pl`

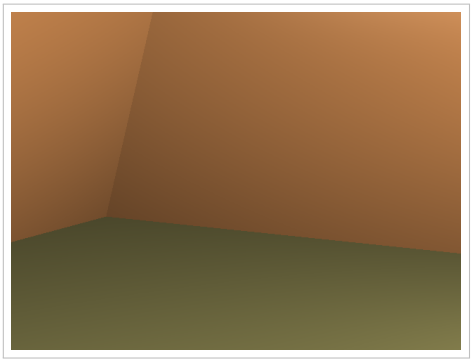
Seguem `k_pl` linhas, trazendo a cor `C` e a localização L das fontes:

`C.r` `C.g` `C.b` L_x L_y L_z

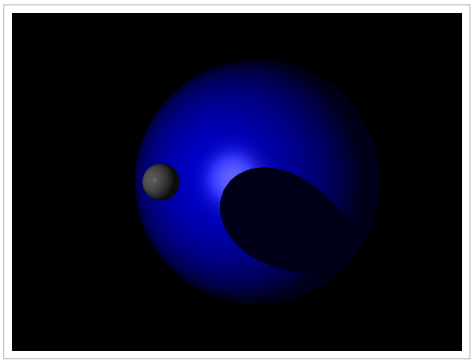
3.5 Testes



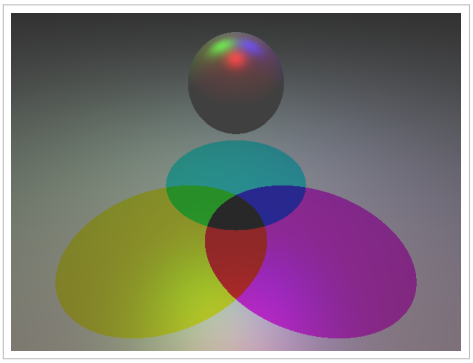
(1) `terra.txt`



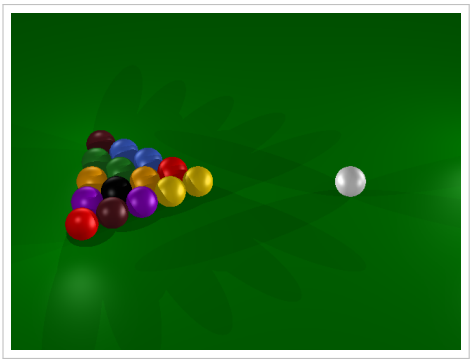
(2) `piramide.txt`



(3) `eclipse.txt`



(4) `venn.txt`



(5) `sinuca.txt`

4 Terceira Entrega

Nas duas primeiras entregas, a cor de um ponto da superfície visível dependia apenas dos coeficientes do material e da iluminação fornecida pelas fontes de luz. Para tanto, apenas dois tipos de raios precisavam ser lançados: raios primários (partindo do foco da câmera) e de sombra (partindo de um ponto atingido por um raio primário e dirigindo-se a uma fonte de luz).

Na terceira entrega, um novo tipo de raio entra em cena. Antes de retornar a cor final do pixel correspondente, um ponto atingido por um raio primário precisa agora somar outras duas cores, além da sua própria: uma é a cor do ponto atingido pelo raio refletido, e a outra é a do raio refratado. A combinação linear das três modela não apenas a iluminação direta recebida pelas luzes, como também a indireta que é refletida pela cena ao redor. Para obter as duas novas componentes, o método que antes fazia apenas o *Ray-Casting* passa a ser chamado de forma recursiva, desta vez traçando raios secundários. Assim como os raios

primários, os raios secundários também retornam a cor do ponto mais próximo atingido, só que estes partem da superfície dos objetos, em vez do foco da câmera.

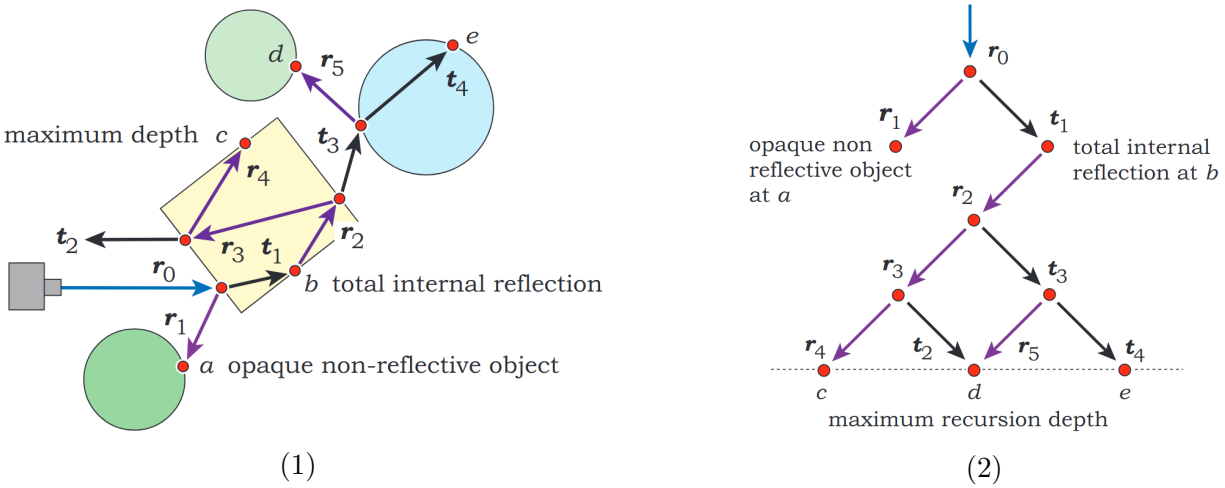


Figura 11: Raios secundários: refletidos (roxos) e refratados (pretos) (1). Árvore de recursão do *Ray-Tracing*. (2)

Raios secundários dão origem a novos raios secundários, então é fundamental garantir que existe sempre um caso base, senão o programa nunca para. Isso é feito estabelecendo um limite máximo `max_depth` para o tamanho da pilha de recursão (Figura 11).

4.1 Reflexão

Um novo parâmetro k_r, o coeficiente de reflexão, será especificado para cada objeto, com $0 \leq k_r \leq 1$. Esse é o escalar que multiplica a parcela da cor final retornada pelo raio secundário refletido.

4.2 Transparência

Há um coeficiente de transmissão k_t, análogo ao k_r, que multiplica a cor do raio na direção refratada, com $0 \leq k_t \leq 1$.

Para a transparência, temos ainda um terceiro parâmetro n: o índice de refração relativo entre o meio interno (sentido oposto da normal) e o meio externo (mesmo sentido da normal).

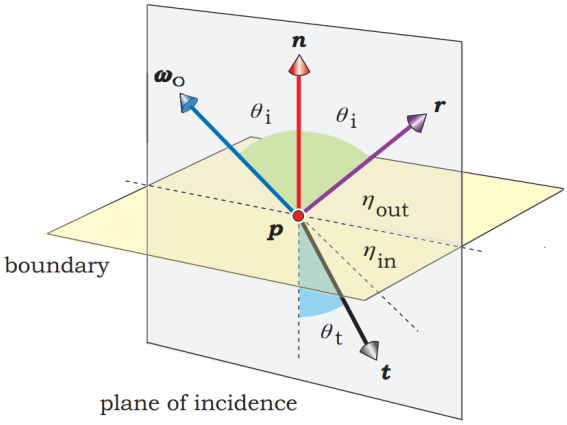


Figura 12: $n = \frac{n_{in}}{n_{out}}$

Na Figura 12, o vetor ω_o é unitário e aponta para o observador (a origem do raio incidente), e o sinal de $\cos\theta_i = \mathbf{n} \cdot \omega_o$ determina o meio onde ele se encontra.

Para o observador no meio externo ($\mathbf{n} \cdot \omega_o > 0$), o vetor $\mathbf{t}$ (unitário por construção) na direção refratada é dado por:

$$\mathbf{t} = -\frac{1}{n}\omega_\mathbf{o} - \left(\sqrt{\Delta} - \frac{1}{n}(\mathbf{n} \cdot \omega_\mathbf{o})\right)\mathbf{n}, \quad \Delta = 1 - \frac{1}{n^2}(1 - (\mathbf{n} \cdot \omega_\mathbf{o})^2)$$

Para o observador no meio interno ($\mathbf{n} \cdot \omega_o < 0$), basta inverter a normal ($\mathbf{n}' = -\mathbf{n}$) e o índice de refração relativo ($n' = \frac{1}{n} = \frac{n_{out}}{n_{in}}$):

$$\mathbf{t} = -\frac{1}{n'}\omega_\mathbf{o} - \left(\sqrt{\Delta'} - \frac{1}{n'}(\mathbf{n}' \cdot \omega_\mathbf{o})\right)\mathbf{n}', \quad \Delta' = 1 - \frac{1}{n'^2}(1 - (\mathbf{n}' \cdot \omega_\mathbf{o})^2)$$

O que acontece se $\Delta < 0$? Essa é justamente a condição que descreve o fenômeno óptico de reflexão total. Nesse caso, o raio refretado simplesmente não existe, e portanto não precisa ser traçado (em termos de implementação, é como se k_t virasse 0 e k_r virasse 1).

4.3 Entrada

O formato das seis primeiras linhas da entrada (câmera, *tela*, cor de fundo) permanece o mesmo da primeira entrega.

A diferença começa a partir da sétima linha, que desta vez traz o limite da recursão `max_depth`:

```
max_depth
```

A próxima linha contém o número de objetos `k_obj`:

```
k_obj
```

Seguem `k_obj` linhas. Se o próximo objeto for uma esfera, a linha segue o formato:

```
Cd.r  Cd.g  Cd.b  k_a  k_d  k_s  η  k_r  k_t  n  *  O_x  O_y  O_z  r
```

E se for um plano:

```
Cd.r  Cd.g  Cd.b  k_a  k_d  k_s  η  k_r  k_t  n  /  P_x  P_y  P_z  n_x  n_y  n_z
```

O restante da entrada descreve as luzes na cena. A próxima linha contém a cor da luz ambiente `Ca`:

```
Ca.r  Ca.g  Ca.b
```

Seguida pelo número de fontes pontuais `k_pl`:

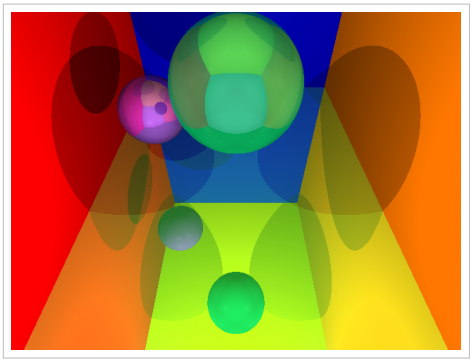
```
k_pl
```

Seguem `k_pl` linhas, trazendo a cor `C` e a localização L das fontes:

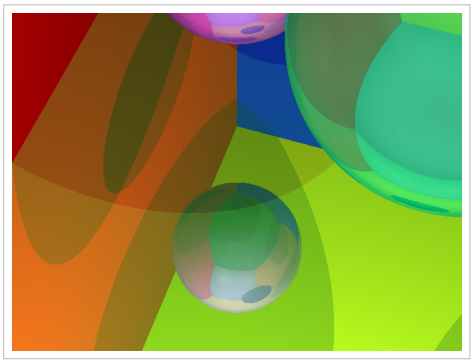
```
C.r  C.g  C.b  L_x  L_y  L_z
```

4.4 Testes

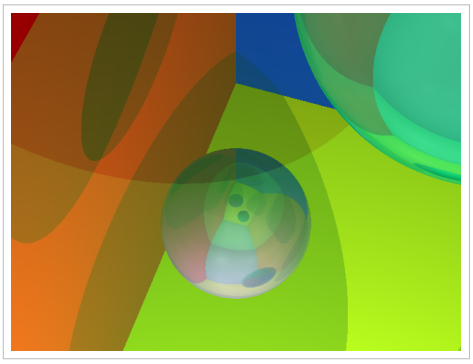
(1) `espelho0.txt`



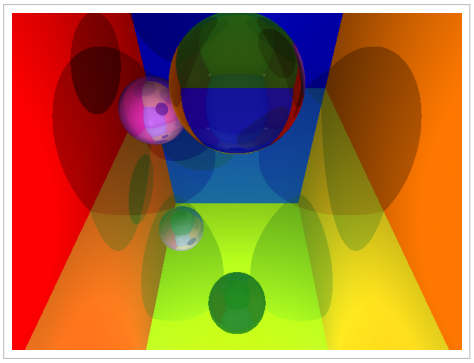
(2) `espelho1.txt`



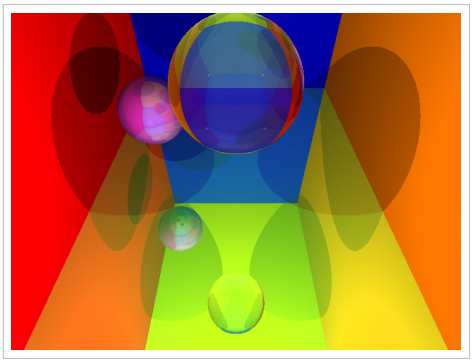
(3) `espelho2.txt`



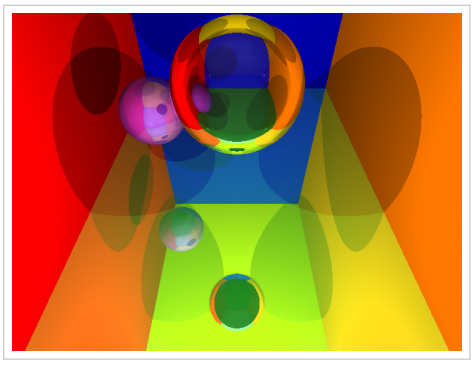
(4) `espelho3.txt`



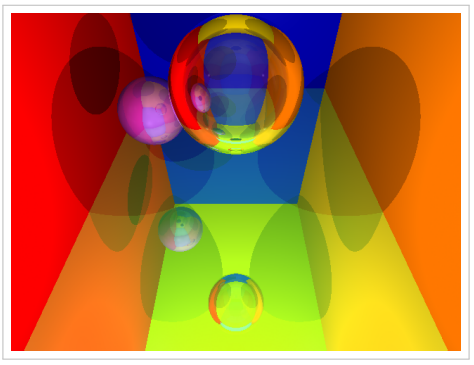
(5) `vidro2.txt`



(6) `vidro4.txt`



(7) `bolha2.txt`



(8) `bolha4.txt`