Modelowanie i wizualizowanie 3W-grafiki

Ray tracing

Alexander Denisjuk

denisjuk@matman.uwm.edu.pl

Uniwersytet Warmińsko-Mazurski w Olsztynie

Wydział Matematyki i Informatyki

ul. Żołnierska 14

10-561 Olsztyn

Ray tracing

Najnowsza wersja tego dokumentu dostępna jest pod adresem

`http://matman.uwm.edu.pl/~denisjuk/`

Scena fotorealistyczna

Ray tracing (śledzenie promieni)

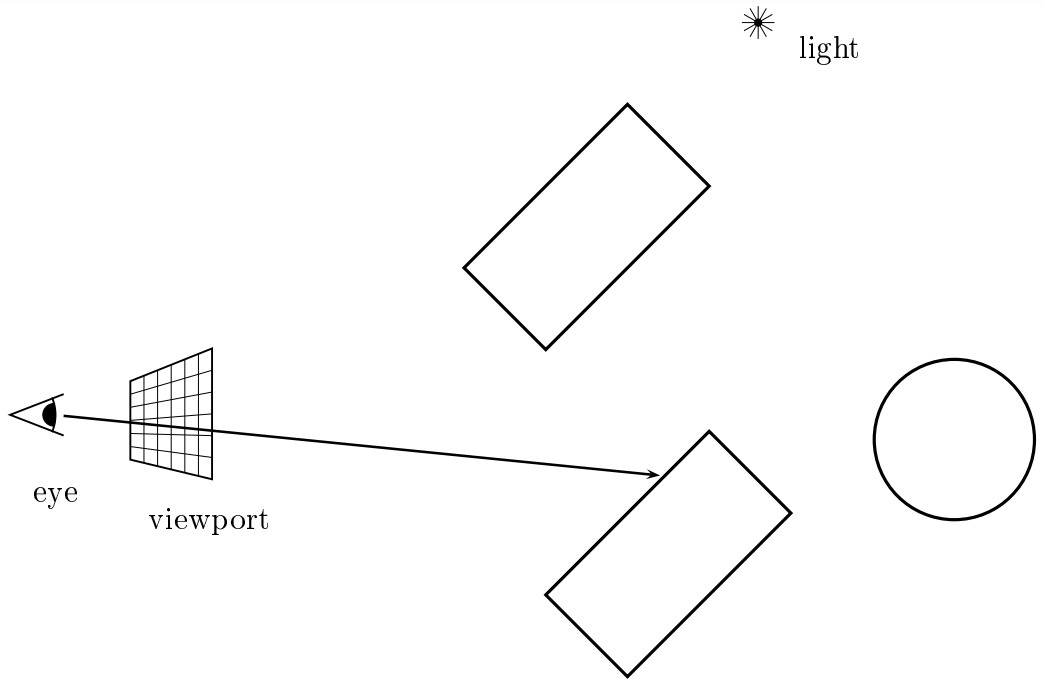

Figure IX.1: The simplest kind of ray tracing, non-recursive ray tracing, involves casting rays of light from the view position through pixel positions. A local lighting model is used to calculate the illumination of the surface intersected by the ray.

Shadow feeler (czujnik cieni)

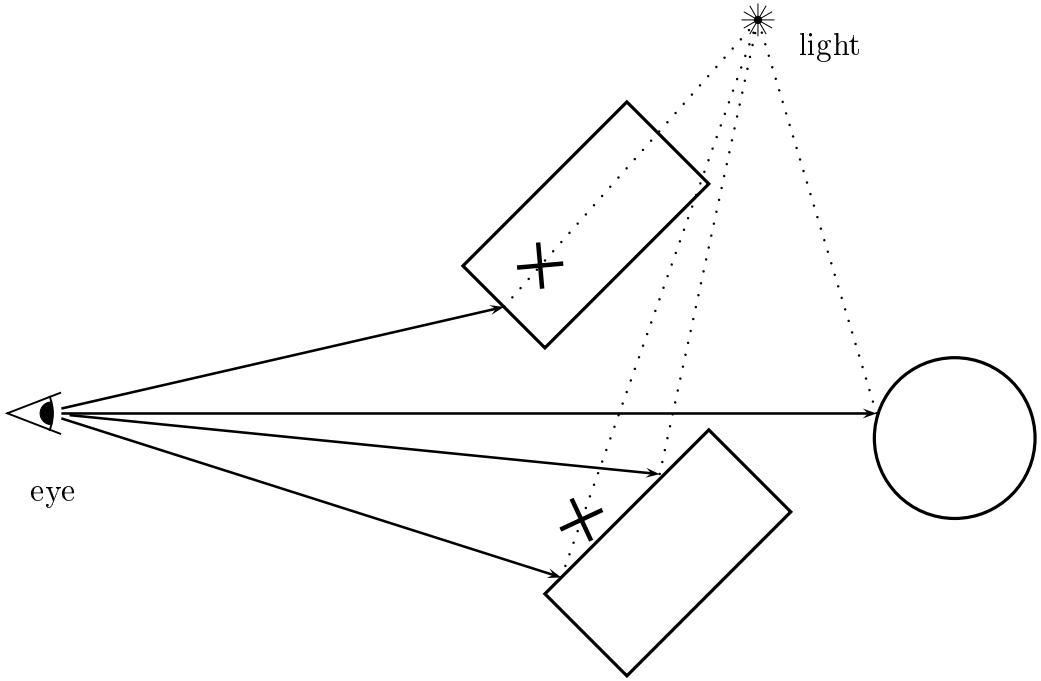

Figure IX.2: Shadow feelers: Rays from the eye are traced to their intersections with objects in the scene. Shadow feeler rays, shown as dotted lines, are sent from the points in the scene to each light in order to determine whether the point is directly illuminated by the point light source or whether it is in a shadow. The two shadow feelers marked with an "**X**" show that the light is not directly visible from the point.

Śledzenie promieni odbijanych

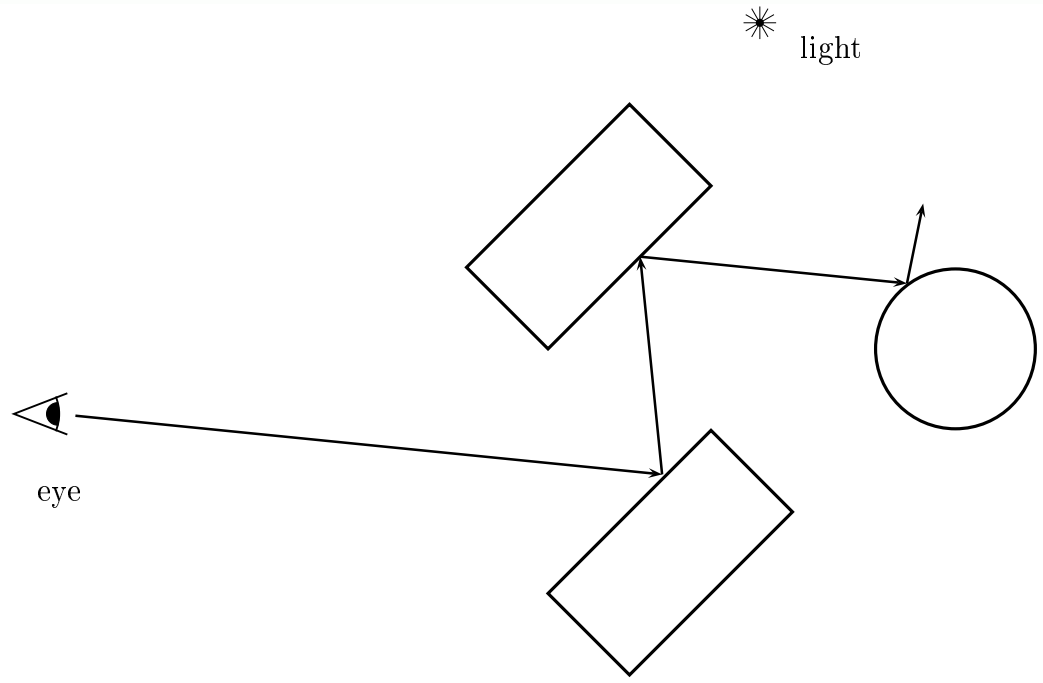

Figure IX.3: Reflection rays: The path of the ray from the eye is traced through multiple reflections. This calculates approximations to the lighting effects of multiple reflections.

$$I = I_{\text{local}} + \rho_{\text{rg}} I_{\text{reflect}}$$

Promienie załamane

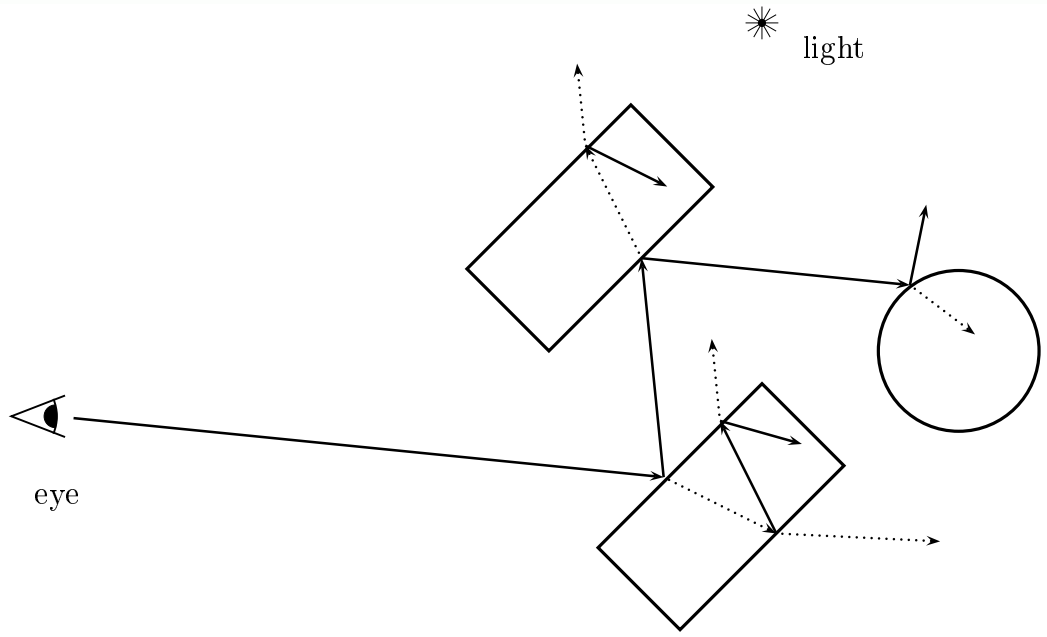

Figure IX.4: Transmission and reflection rays: The path of the ray from the eye is traced through multiple reflections and transmissions. Reflection rays are shown as solid lines, transmission rays as dotted lines. The shadow feeler rays would still be used, but are not shown.

$$I = I_{\text{local}} + \rho_{\text{rg}} I_{\text{reflect}} + \rho_{\text{tg}} I_{\text{xmit}}$$

Lokalne oświetlenie i promienie odbijane

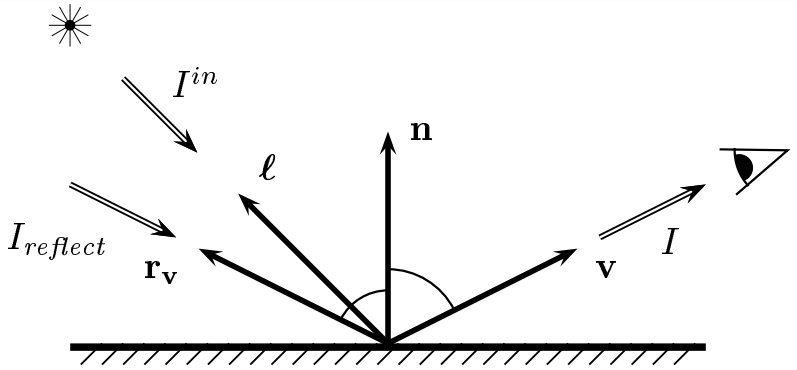

Figure IX.5: The usual setup for reflection rays in basic recursive ray tracing. The vector $\mathbf{v}$ points in the direction opposite to the incoming ray. The direction of perfect reflection is shown by the vector $\mathbf{r_v}$. The vector ℓ points to a point light source. I is the outgoing light intensity as seen from the direction given by $\mathbf{v}$. $I_{reflect}$ is the incoming light from the reflection direction $\mathbf{r_v}$. I^{in} is the intensity of the light from the light source. (Compare this to figure III.7 on page 72.)

Wektor odbijany $r_v = 2(v \cdot n) - v$.

Oświetlenie punktu na powierzchni

$$I = I_{\text{local}} + \rho_{\text{rg}} I_{\text{reflect}}$$

$$I_{\text{local}} = \rho_a I_a^{in,i} + \delta_i \cdot \left(\rho_{\text{d}} I_{\text{d}}^{\text{in},i} (\ell_i \cdot n) + \rho_{\text{s}} I_{\text{s}}^{\text{in},i} (r_v \cdot \ell_i)^f \right)$$

- $\delta_i = 1$, jeśli punkt jest bezpośrednio oświetlony światłem i, 0 — w przeciwnym przypadku.
- współczynniki ρ zależą od kolorów (częstotliwości)
- I_{reflect} oblicza się rekurencyjnie powtarzając algorytm ray tracing

Promienie załamane

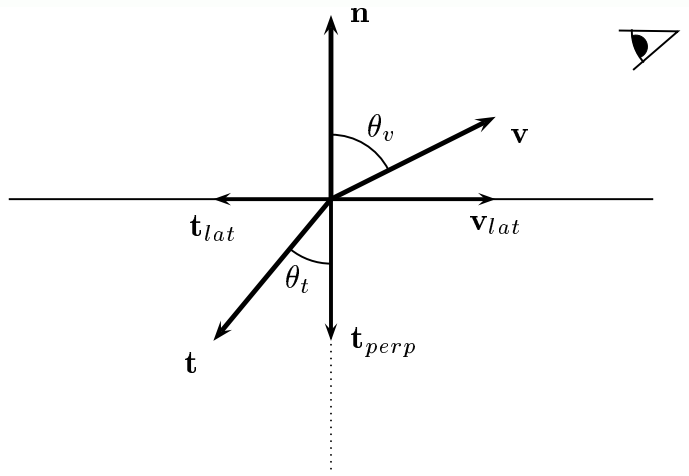

Figure IX.6: Computing the transmission ray direction $\mathbf{t}$. The horizontal line represents the surface of a transmissive material; $\mathbf{n}$ is the unit vector normal to the surface. The vector $\mathbf{v}$ points in the direction opposite to the incoming ray. The direction of perfect transmission is shown by the vector $\mathbf{t}$. The vectors $\mathbf{v}_{lat}$ and $\mathbf{t}_{lat}$ are the projections of these vectors onto the plane tangent to the surface. And, $\mathbf{t}_{perp}$ is the projection of $\mathbf{t}$ onto the normal vector.

Prawo Snelliusa

$$\frac{\sin\theta_v}{\sin\theta_t} = \eta.$$

Współczynnik załamania

- $\eta \approx 1,3$ — powietrze $\rightarrow$ woda.
- $\eta \approx 1,5$ — powietrze $\rightarrow$ szkło.
- $\sin\theta_t = \eta^{-1}\sin\theta_v$.
- Jeżeli $\eta^{-1}\sin\theta_v > 1$, to nie ma załamania, tylko całkowite wewnętrzne odbijanie

Obliczenie wetora t

- $v_{\text{lat}} = v - (v \cdot n)n$
- $\|t_{\text{lat}}\| = \sin\theta_t = \eta^{-1} \sin\theta_v = \eta^{-1}\|v_{\text{lat}}\|$
- $t_{\text{lat}} = -\eta^{-1} v_{\text{lat}}$
- $\cos\theta_t = \sqrt{1 - \sin^2\theta_t} = \sqrt{1 - \|t_{\text{lat}}\|^2}$ $(\|t_{\text{lat}}\| < 1)$
- $t_{\text{perp}} = -\sqrt{1 - \|t_{\text{lat}}\|^2} \cdot n$
- $t = t_{\text{lat}} + t_{\text{perp}}$
- $t_{\text{perp}} = -\sqrt{1 - \eta^{-2}\left(1 - (v \cdot n)^2\right)} \cdot n$
- $t = \eta^{-1}\left((v \cdot n)n - v\right) - \sqrt{1 - \eta^{-2}\left(1 - (v \cdot n)^2\right)} \cdot n$

Obliczenie wetora t

```
CalcTransmissionDirection(v, n, η){
    t_lat = ((v · n)n − v)/η;
    sinSq=||t_lat||^2;
    if (sinSq>1){
        return "Całkowite odbicie!";
    }
    t = t_lat − sqrt(1 − sinSq) · n;
    return t;
}
```

Rozszerzenie modelu Phonga

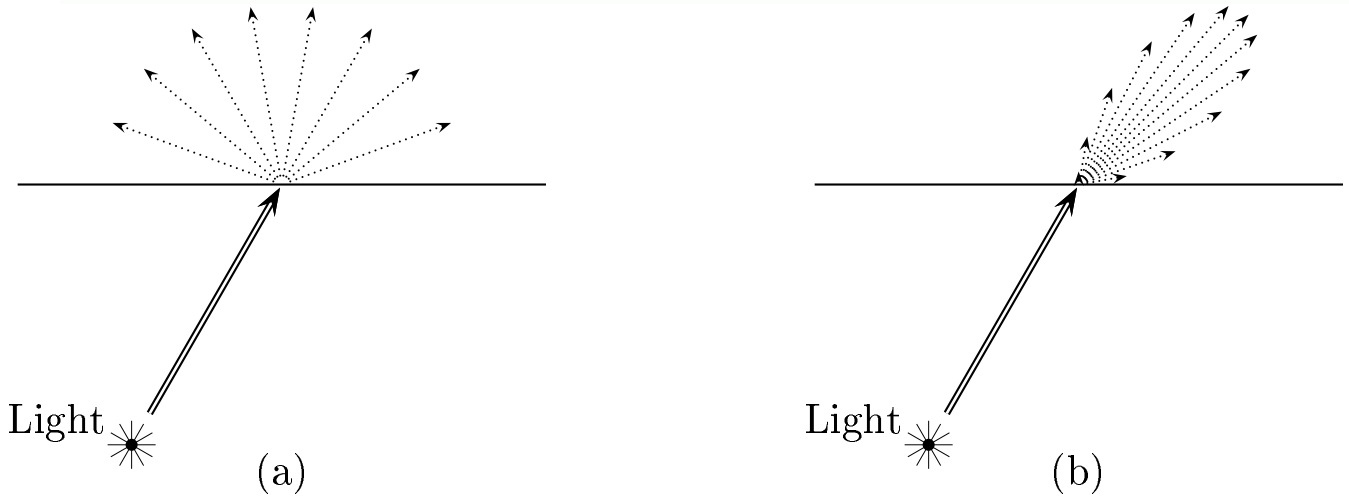

Figure IX.7: (a) Diffusely transmitted light. (b) Specularly transmitted light. The specularly transmitted light is centered around the transmission direction from Snell's law.

Rozszerzenie modelu Phonga

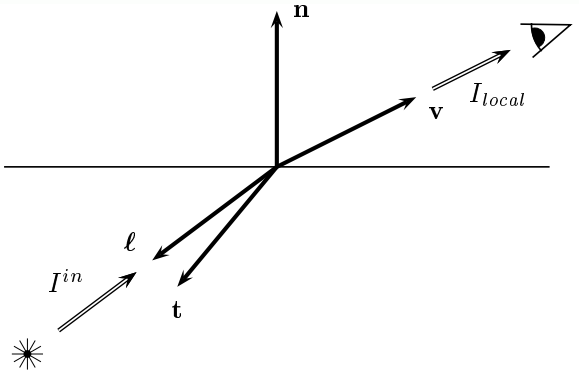

Figure IX.8: The vectors used in the computation of transmitted light are $\mathbf{v}$, $\boldsymbol{\ell}$, $\mathbf{t}$, and $\mathbf{n}$. The vector $\mathbf{v}$ points in the direction opposite to the incoming ray. The direction of perfect transmission is shown by the vector $\mathbf{t}$. The direction opposite to the incoming light is given by $\boldsymbol{\ell}$.

$$I^i_{\text{local}} = \rho_{\text{a}} I^{\text{in},i}_{\text{a}} + \delta'_i \cdot \left(\rho_{\text{dt}} I^{\text{in},i}_{\text{d}} (\ell_i \cdot (-n)) + \rho_{\text{st}} I^{\text{in},i}_{\text{s}} (t \cdot \ell_i)^f \right)$$

Rozszerzenie modelu Phonga — całkowite oświetlenie

$$
\begin{aligned}
I_{\text{local}} = & \rho_{\text{a}} I_{\text{a}}^{\text{in}} + \rho_{\text{d}} \sum_{i=1}^{k} \delta_i I_{\text{d}}^{\text{in},i} (\ell_i \cdot n) + \\
& + \rho_{\text{s}} \sum_{i=1}^{k} \delta_i I_{\text{s}}^{\text{in},i} (r_v \cdot \ell_i)^f + \rho_{\text{dt}} \sum_{i=1}^{k} \delta_i' I_{\text{d}}^{\text{in},i} (\ell_i \cdot (-n)) + \\
& + \rho_{\text{st}} \sum_{i=1}^{k} \delta_i' I_{\text{s}}^{\text{in},i} (t \cdot \ell_i)^f + I_{\text{e}}
\end{aligned}
$$

Algorytm

Dla każdego promienia:

- Znajdź pierwsze miejsce przecięcia ze sceną.
  - Jeśli promień nie przecina żadnego obiektu ze sceny, wykorzystuj „kolor tła”.
- Oblicz oświetlenia punktu zgodnie z modelem oswietlenia.
- Wypuść promienie odbijane oraz załamane.
- Zastosuj rekurencyjnie algorytm do każdego wypuszczonego promienia.
- Dodaj wyniki obliczenia ośiwetleń.

Warunek zakończenia rekurencji: ilość odbić.

Główny program

```
RayTraceMain(){
    // x jest lokacją widza
    // maxDepth>0
    for each pixel p na ekranie do{
        u= wektor jednostkowy xp
        RayTrace(x, p, maxDepth);
        Pokoloruj p obliczonym kolorem;
    }
}
```

Program rekurencyjny I

RayTraceMain(s, u, depth){

// s jest początkiem promienia

// u jest kieunkiem promienia

// depth jest poziomem

// wynik: (R, G, B)

Znajdź punkty przecięcia promienia z każdym z obiektów sceny.

Niech z będzie pierwszym punktem przecięcia, n będzie wektorem normalnym do powierzchni w tym punkcie

Program rekurencyjny II

```
if (brak punktów){
    return kolor tła
}
for (każde światło){
    utwórz czujnik cieni
    sprawdź, czy czujnik napotyka
    obiekty. Ustal δ_i oraz δ'_i
}
Kolor=I_local
```

Program rekurencyjny III

```
if (depth==0){ //koniec obliczeń
    return Kolor;
}
// Dodaj kolor odbijany
if (ρ_rg ≠ 0){
    r = u − 2(u · n)n;
    Kolor=Kolor+ρ_rg∗RayTrace(z,r,depth−1);
}
```

Program rekurencyjny IV

```
    // Dodaj światło załamane
    if (ρ_tg ≠ 0){
        t=ObliczKierunekZałamania(−u, n, η);
        if (t jest określone){
            Kolor=Kolor
                  +ρ_rg∗RayTrace(z,t,depth−1);
        }
    }
    Retunr Kolor;
}
```

Sprawdzenie przecięcia

- Obiekty modeluje się za pomocą prostych figur: sfera, walec, stożek, torus, wielobok płaski, wielobok o bokach w postaci powierzchi Béziera, B-spline powierzchni.
- Sprawdza się dla każdego promienia, dla każdego czujnika cieni.
- Zależy od poziomu maxDepth.
- Najbardziej kosztowne względem obliczeń działanie.