2018

# Moduł 2

## GRAFIKA KOMPUTEROWA 3D

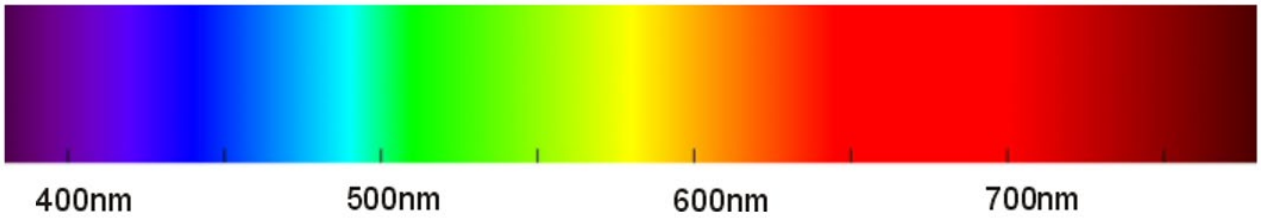
BARBARA PUTZ

[Słowa kluczowe]

[Streszczenie]

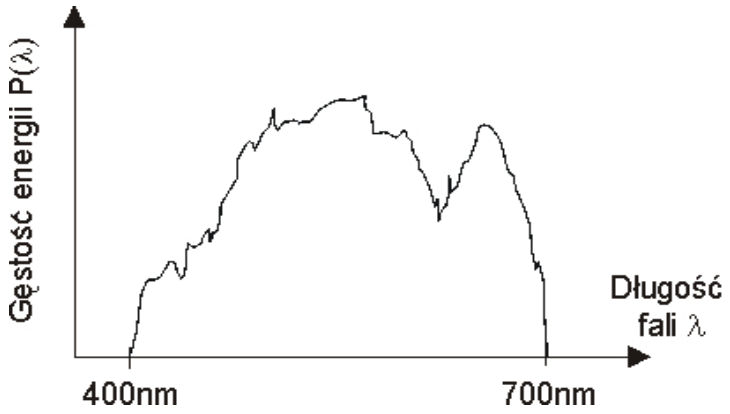
# Spis treści

# 1 Podstawy teorii barw




*Rysunek 1. Rozkład widmowy światła.*

Z fizycznego punktu widzenia światło jest sumą energii fal elektromagnetycznych. Długości fal światła widzialnego zawarte są w przedziale od 380nm (dla fioletu) do 780nm (dla czerwieni), co obrazuje rozkład widma na Rys. 1, będący efektem rozszczepienia światła białego. To oznacza, że światło białe jest mieszaniną wszystkich barw z tego spectrum. Ilość energii przypadającej dla każdej długości fali jest reprezentowana przez widmowy rozkład energii P(λ) (Rysunek 1).




*Rysunek 2. Typowy rozkład energii widmowej P(λ) światła.*

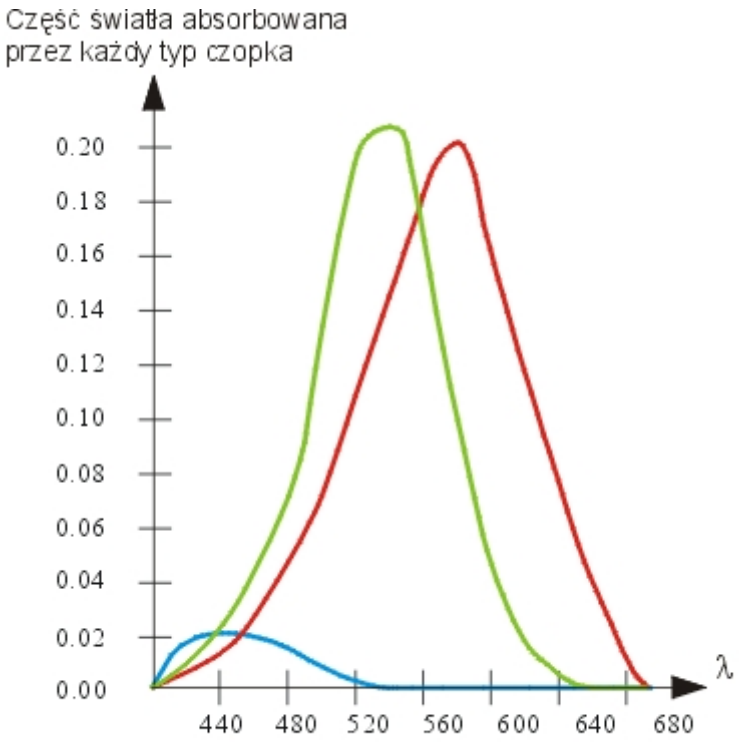
Światło barwne wywołuje o wiele bogatsze wrażenia wzrokowe niż światło achromatyczne. Ze światłem barwnym związane są trzy podstawowe pojęcia: odcień barwy, nasycenie i jasność. Rozpatrzmy po kolei każde z nich.

**Odcień barwy** - dotyczy samego koloru, tzn. związany jest z takimi pojęciami jak czerwony, żółty, zielony, pomarańczowy.

**Nasycenie** - właściwość wrażenia wzrokowego umożliwiająca ocenę udziału barwy chromatycznej, czystej, we wrażeniu ogólnym. Różnicuje ono barwy nasycone i nienasycone (do barw nienasyconych możemy zaliczyć pastele). Określa procentowy udział bieli w barwie (barwy wydają się bledsze, im większy jest udział światła białego w mieszaninie). Przy zmniejszaniu nasycenia barwy do zera, niezależnie od odcienia barwy uzyskamy barwę białą.

**Jasność** - oznacza achromatyczny opis odbieranej jasności przy obserwacji obiektu odbijającego światło. Jasność jest właściwością wrażenia wzrokowego powodującą, że ciało albo powierzchnia wydaje się przepuszczać lub odbijać większą lub mniejszą część światła padającego. Przy zmniejszaniu jasności niezależnie od odcienia barwy uzyskamy barwę czarną.

Często można się spotkać z czwartym określeniem tzn. z jaskrawością. **Jaskrawość** związana jest z przypadkiem, gdy nie mamy do czynienia z obiektami odbijającymi światło, ale światło emitującymi.




*Rysunek 3. Progi czułości oka w zakresie fal światła widzialnego*

Cechą charakterystyczną receptorów odbierających światło jest to, iż czułość receptorów odbierających światło niebieskie jest znacznie mniejsza niż czułość receptorów światła czerwonego i zielonego. Wynika to z tego, że oko ludzkie jest dostosowane do otaczającego nas źródła światła - Słońca, emitującego maksymalną ilość promieniowania w zakresie światła żółtego, tj. ok. 550 nm.

Oko ludzkie może rozróżnić do kilkudziesięciu tysięcy różnych barw zawartych w przestrzeni barw. Aby takie rozróżnienie było możliwe, należy porównywać ze sobą barwy nieznacznie różniące się od siebie. W przeciwnym przypadku mózg ludzki zarejestruje obie barwy jako identyczne (nawet jeśli ich rzeczywiste barwy znacznie się różnią). Przyczyna tkwi w tym, że człowiek określa kolory na podstawie wywoływanych przez te kolory wrażeń wzrokowych, a nie na podstawie ilościowego wyznaczenia rzeczywistych barw.
W badaniach tego typu sprawdzane są najmniejsze różnice barw, które są dostrzegalne przez oko ludzkie. Porównywalne barwy różnią się od siebie nieznacznie pod względem:

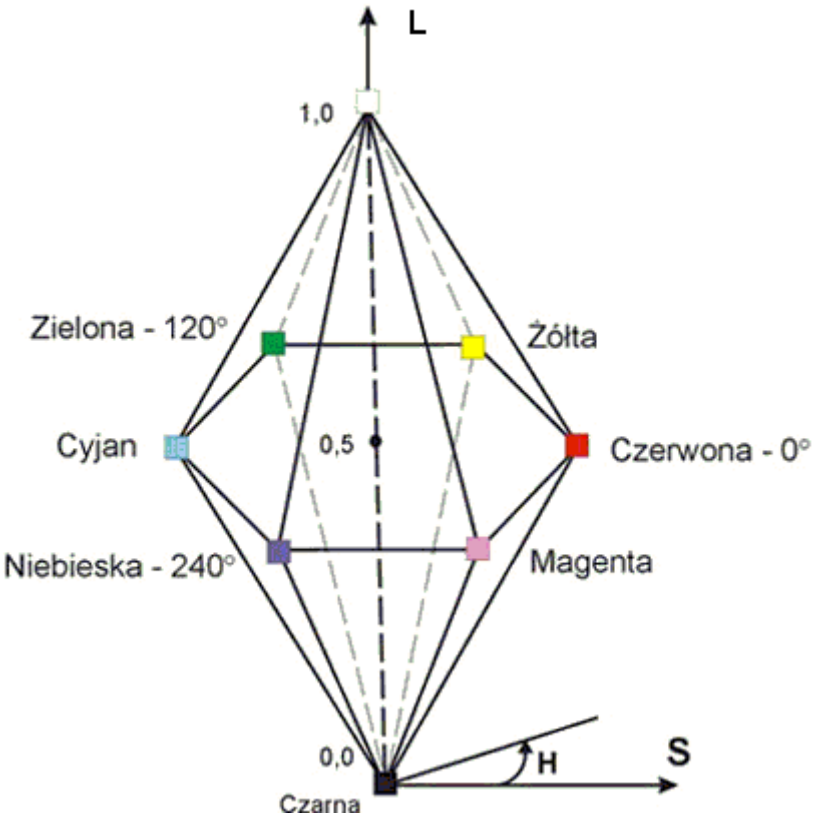
- Luminancji (jasności)
- Odcienia barwy
- Nasycenia.

Pozwala to określić tzw. *progi czułości* oka ludzkiego.
Rozdzielczość oka w sensie widzenia barw (progi czułości):

- **Różnica luminancji (jasności)** - określa ilość odcieni szarości, które oko ludzkie jest w stanie rozróżnić. Człowiek rozróżnia te odcienie jako jaśniejsze lub ciemniejsze. Rozróżnialnych jest około **40-50 poziomów szarości**.
- **Różnica odcieni** - określa ilość odcieni, które oko jest w stanie rozróżnić przy w pełni nasyconej barwie. Różnica ta wynosi około 4nm, co powoduje, że jest rozróżnialnych około **150 różnych odcieni**.
- **Różnica nasycenia** - określa ilość rozróżnianych poziomów nasycenia barwy. Testy takie są prowadzone przy ustalonej wartości jasności i odcienia. Wartość ta zależy przede wszystkim od długości fali dominującej (czyli odcienia) i samego nasycenia. Rozróżnialnych jest około **25 poziomów**

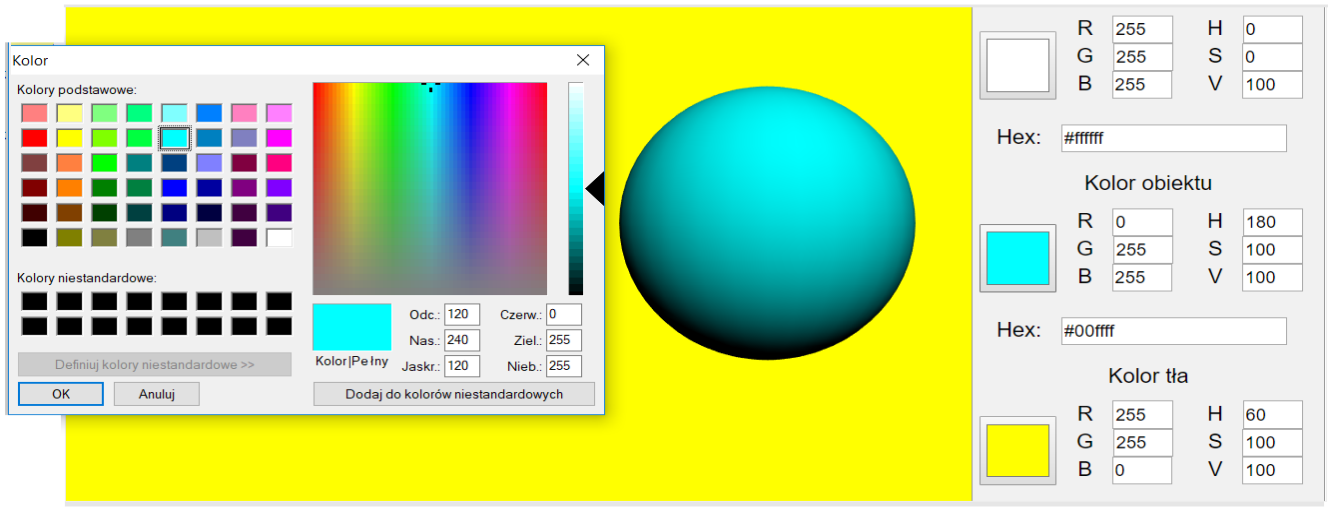
**nasycenia** barwy dla odcieni fioletowych lub czerwonych, czyli na krańcach widma. W środku widma ta ilość spada do 20-15 w związku z tym, że w tym zakresie oko posiada większą rozdzielczość w stosunku do odcienia barw.

**Modele HSB/HSL/HSV** - addytywne modele barw, w których barwy zostały zdefiniowane poprzez ich atrybuty, tzn.: kolor, jasność i nasycenie. Nazwa pochodzi od nazw atrybutów definiujących barwę, tzn.: ang. *Hue* - odcień, ang. *Saturation* - nasycenie, ang. *Brightnes* - jaskrawość, ang. *Lightness* - jasność, ang. *Value* - wartość jasności. Model HLS obejmuje przestrzeń barw zbliżoną do podwójnego stożka, w którym barwy widmowe są rozmieszczone wokół wspólnej podstawy stożków.

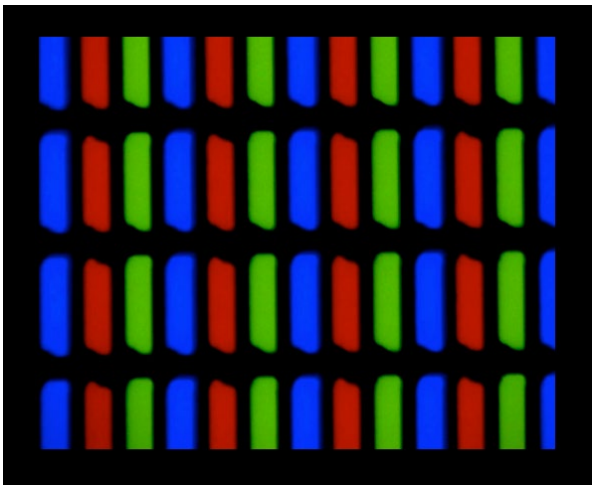



*Rysunek 5. Model HLS*

Na zakończenie zajmijmy się modelem RGB, bezpośrednio odpowiadającym mieszaniu barw trzech rodzajów świecących pikseli na matrycy monitora czy telewizora.

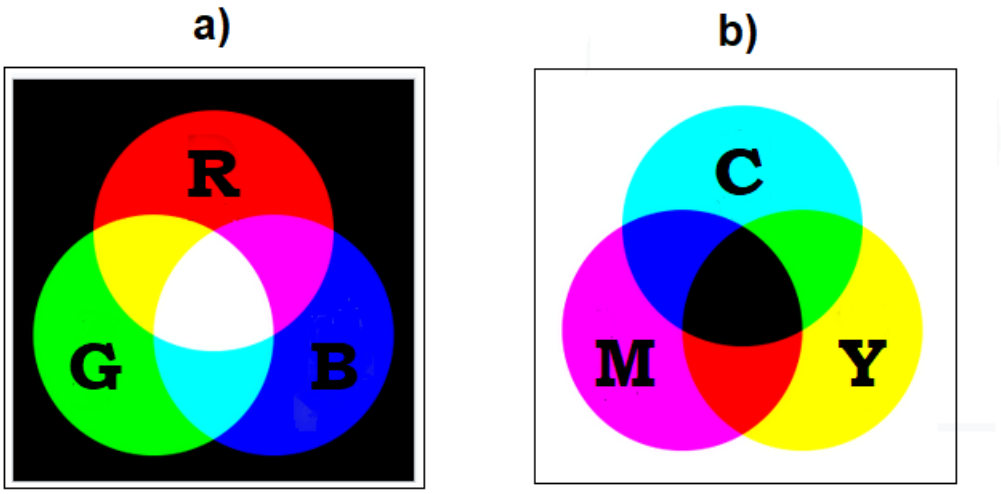



*Rysunek 4 – **Aplikacja nr 1.** Model RGB. Każdą ze składowych R, G, B można zmieniać w zakresie od 0 do 255. Warto się przyjrzeć, jak wygląda zapis wybranej barwy w modelu HSV i w układzie szestnastkowym (HEX). Kliknięcie w kolorowe okienko przenosi nas do okna z innymi metodami wyboru barwy.*



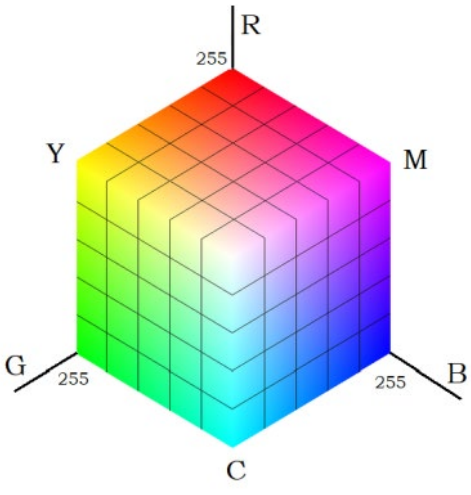
*Rysunek 6. Subpiksele na monitorze LCD.*
*https://en.wikipedia.org/wiki/LCD_television#/media/File:TN_display_closeup_300X.jpg*

## 1.1 Modele RGB i CMYK

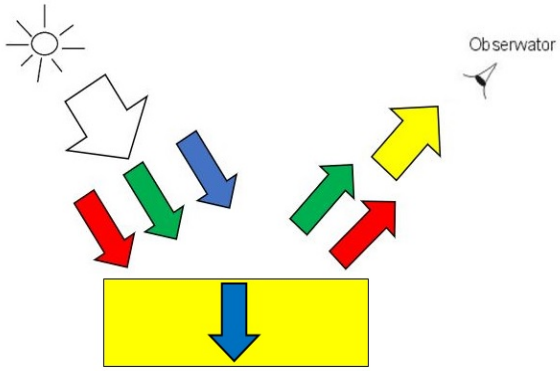



*Rysunek 9. Modele przestrzeni barw: a) addytywny - RGB i b) subtraktywny - CMY*

*Rysunek 7. Model RGB.*
*https://en.wikipedia.org/wiki/File:RGB_illumination.jpg*

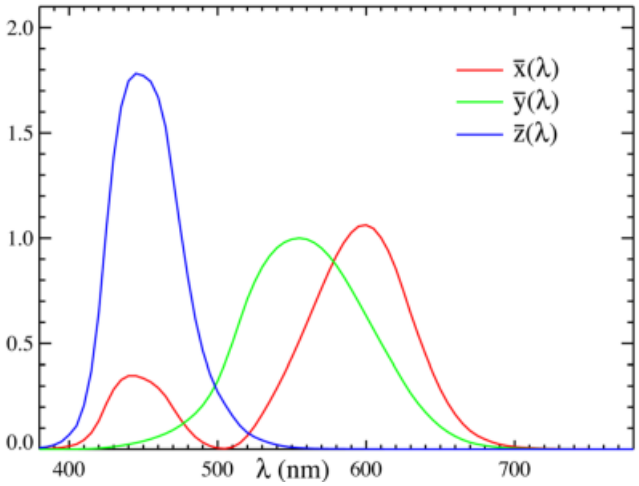



*Rysunek 8. Sześcian RGB*

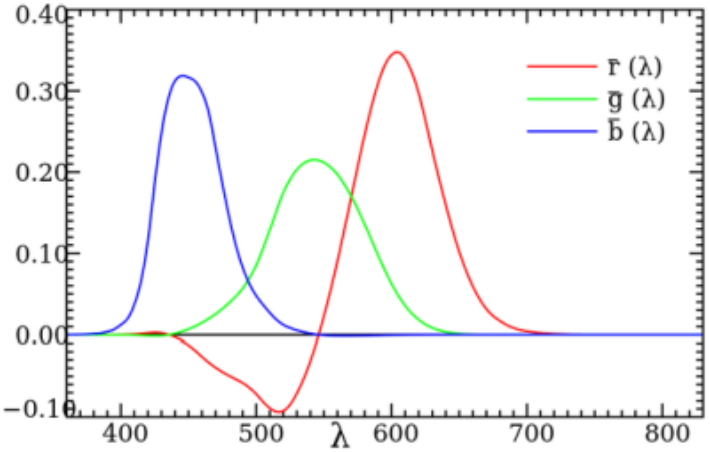



*Rysunek 10. Zasada działania modelu CMY. Światło białe pada na żółtą powierzchnię, która pochłania barwę niebieską. Światło odbite zawiera już tylko składowe czerwoną i zieloną, obserwator widzi barwę żółta.*

Jeśli teraz zmieszamy farbę Yellow (która pochłania światło niebieskie) z farbą Cyan (która pochłania swiatło czerwone) i z farbą Magenta (która pochłania światło zielone), to otrzymamy farbę, która nie odbija żadnego światła, czyli farbę czarną. I to właśnie obrazuje model CMY.

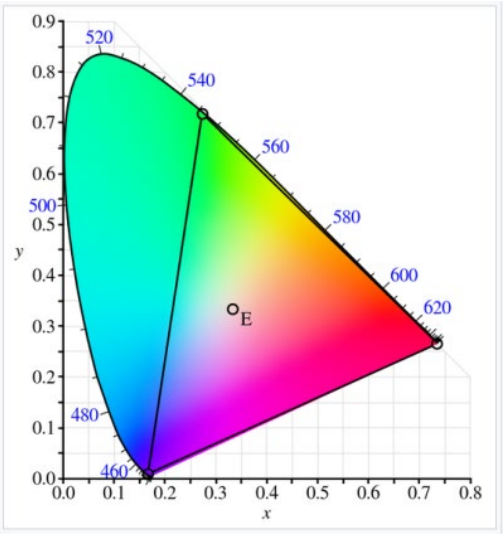



*Rysunek 12. Trójchromatyczne składowe widmowe. https://pl.wikipedia.org/wiki/CIEXYZ#/media/File:CIE1931_XYZCMF.png*




*Rysunek 11. Funkcje składowe CIE RGB. https://en.wikipedia.org/wiki/CIE_1931_color_space#/media/File:CIE1931_RGBCMF.svg*

http://people.westminstercollege.edu/faculty/ccline/courses/resources/light/D-Color/T-CIE-Color%20Space.pdf

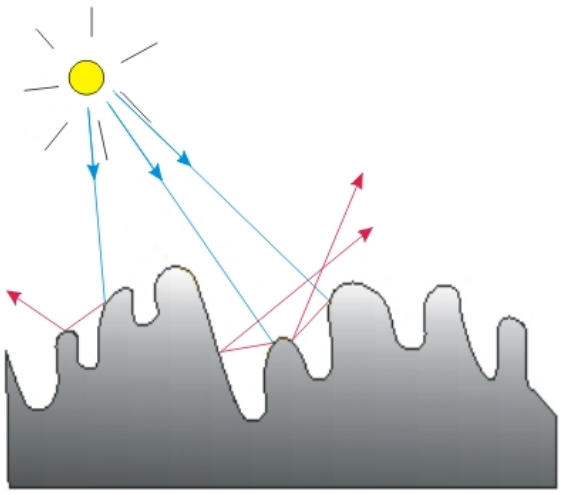



*Rysunek 13. Wykres chromatyczności CIE i RGB gamut oraz punkt światła białego E. https://en.wikipedia.org/wiki/CIE_1931_color_space #/media/File:CIE1931xy_CIERGB.svg*

# 2 Modelowanie oświetlenia powierzchni

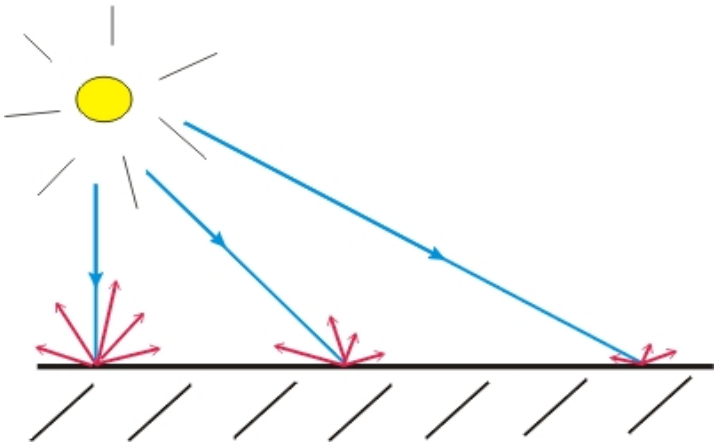
## 2.1 Odbicie rozproszone

Najprostszym sposobem uzyskania wrażenia oświetlonej, trójwymiarowej powierzchni jest modelowanie odbicia rozproszonego. Odbiciem rozproszonym charakteryzują się powierzchnie matowe, bez połysku, odbijające promień padającego światła w najrozmaitszych kierunkach, co jest spowodowane występującymi na nich mikronierównościami:



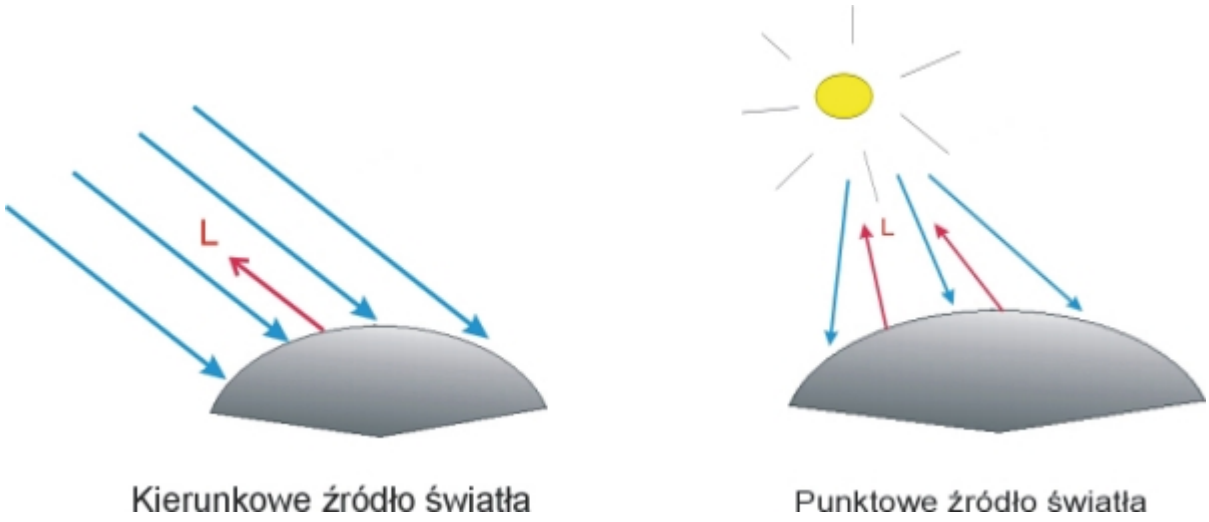
*Rysunek 14. Odbicie promieni światła na mikronierównościach powierzchni*

Intensywność światła rozpraszanego zależna jest wyłącznie od kierunku padania światła na powierzchnię: im bardziej prostopadle pada promień światła na powierzchnię, tym silniej jest ono odbijane:



*Rysunek 15. Intensywność odbicia rozproszonego maleje wraz ze wzrostem kąta padania światła na powierzchnię*

Intensywność odbicia rozproszonego jest przy tym **niezależna od kierunku patrzenia**, czyli od położenia obserwatora. Do obliczenia odbicia rozproszonego potrzebne są zatem dwa wersory, czyli wektory o długości jednostkowej, określające kierunek padania promienia i kierunek normalnej do powierzchni. Pierwszy z tych wersorów wyznacza się jako wersor skierowany z danego punktu powierzchni do źródła światła (oznaczany zwykle jako **L** - od słowa *light*). Jeśli źródło światła leży w nieskończonej odległości od oświetlonej powierzchni (np. jest to światło słoneczne), mamy do czynienia z tzw. kierunkowym źródłem światła, dla którego wersor **L** ma stały zwrot i kierunek:

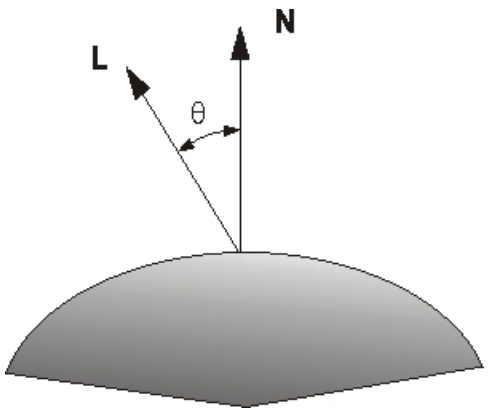



*Rysunek 16. Wersor **L** jest zawsze skierowany do źródła światła*

Drugim wersorem jest wersor normalny do powierzchni, znany już z wykładu o modelowaniu 3D. Nadal będziemy go oznaczać jako **N**. Intensywność odbicia rozproszonego, zależna od kąta padania światła na powierzchnię, jest więc funkcją kąta pomiędzy tymi dwoma wersorami.

Zgodnie z prawem fizyki, znanym jako *prawo Lamberta*, intensywność odbicia rozproszonego jest funkcją cosinusa kąta między wersorami **N** oraz **L**:

$$I_d = I_p k_d \cos\theta$$




*Rysunek 17. Wersory niezbędne do wyznaczenia intensywności odbicia rozproszonego*

gdzie:

$I_p$ - intensywność światła padającego

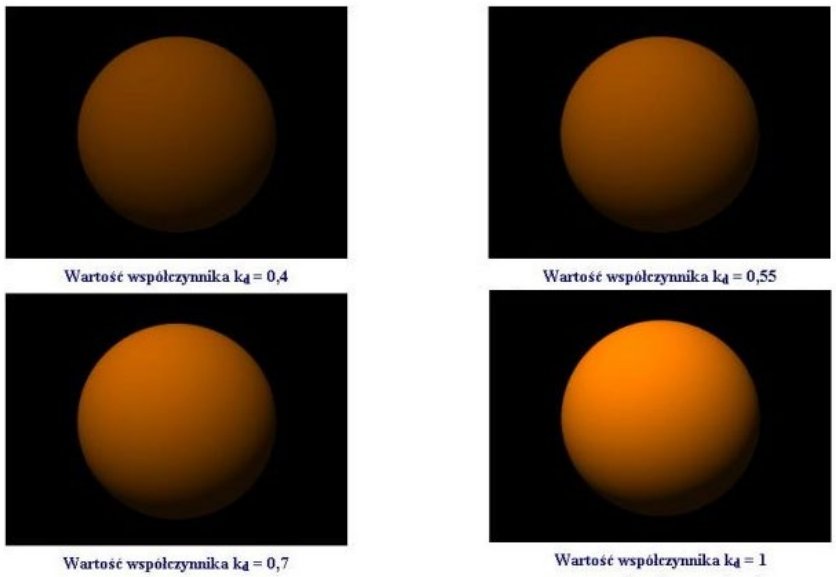
$I_d$ - intensywność światła odbitego (rozpraszanego - ang. *diffuse*)

$k_d$ - współczynnik odbicia rozproszonego, zawarty w przedziale <0, 1>, określający, jaka część światła padającego jest rozpraszana.

Ponieważ **N** oraz **L** są wersorami (czyli mają długość jednostkową), to wartość cosinusa kąta między nimi można - zgodnie z regułami matematyki - zastąpić ich iloczynem skalarnym, czyli:

$$I_d = I_p k_d (\mathbf{N} \cdot \mathbf{L})$$

W ten sposób otrzymujemy niezwykle prosty wzór, pozwalający na szybkie w obliczeniach uzyskanie wrażenia oświetlonej powierzchni 3D. Wpływ współczynnika odbicia rozproszonego $k_d$ na intensywność odbitego światła zilustrowano na Rys. 18.




*Rysunek 18. Wpływ współczynnika odbicia rozproszonego na wygląd oświetlonej powierzchni*

Symulując odbicie rozproszone możemy spotkać się z sytuacją, gdy rzuty dwóch równoległych płaszczyzn będą się nakładać w obrazie. Nie będzie można ich wówczas rozróżnić, gdyż obie będą miały ten sam wersor normalny. W celu ominięcia tej niedogodności wprowadzono **współczynnik tłumienia źródła światła**, pozwalający wyznaczać intensywność odbicia w zależności od odległości $(d_L)$ oświetlonego obiektu od źródła światła. Najczęściej używa się wzoru

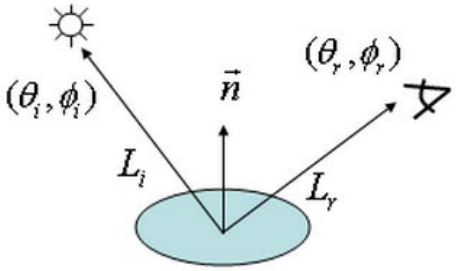
$$I_d = \mathrm{f}(d_L) I_p k_d (\mathbf{N} \cdot \mathbf{L})$$

gdzie współczynnik tłumienia $\mathrm{f}(d_L)$ jest liczony z zależności empirycznej:

$$\mathrm{f}(d_L) = min\left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right)$$

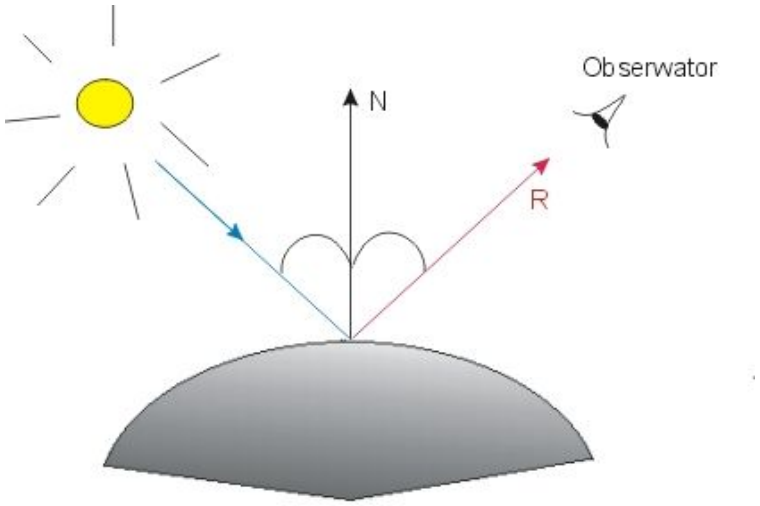
Wzór ten zapewnia, że współczynnik tłumienia nie przekroczy wartości 1, niezależnie od sposobu liczenia wielkości $d_L$. Wielkość ta jest odległością powierzchni od źródła światła; najlepiej jest liczyć ją w sposób względny, tak by na powierzchni znajdującej się najbliżej źródła światła tłumienie nie następowało.

Uogólnieniem modelu Lamberta, lepiej oddającym własności odbijające mikronierówności, jest model odbicia rozproszonego Orena-Nayara http://en.wikipedia.org/wiki/Oren%E2%80%93Nayar_reflectance_model

## 2.2 Odbicie lustrzane

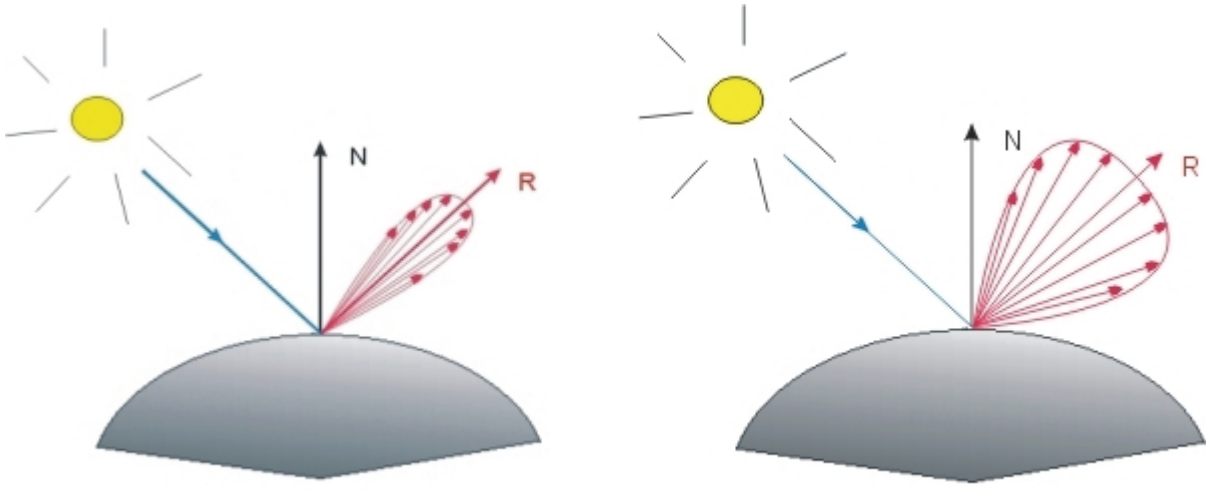
Przeciwieństwem powierzchni matowych, rozpraszających światło, są powierzchnie idealnie lustrzane (zwierciadlane). Odbijają one promień światła tylko w jednym kierunku, zgodnie z powszechnie znanym prawem fizyki:




*Rysunek 19. Powierzchnia idealnie błyszcząca: kąt padania równa się kątowi odbicia. R jest wersorem promienia odbitego.*

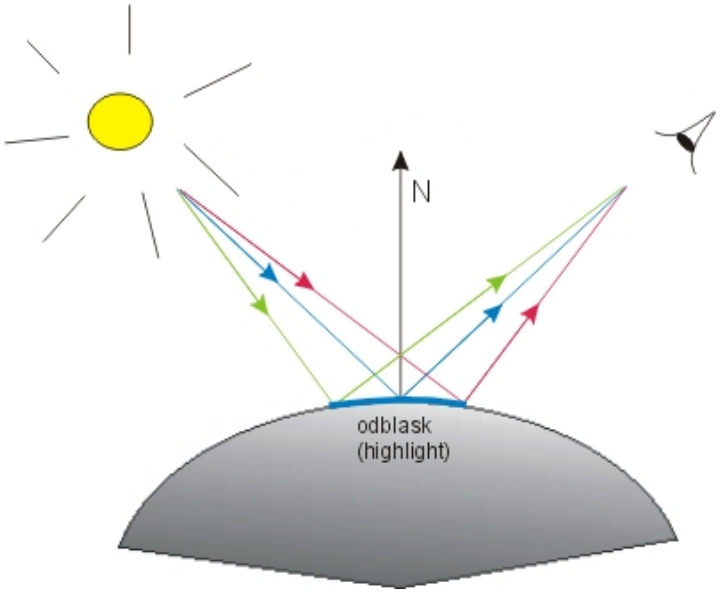
Jeśli więc powierzchnia jest idealnym lustrem, to spośród wielu promieni wychodzących ze źródła światła do obserwatora stojącego w wybranym punkcie trafia tylko jeden z nich; obserwator widzi więc na takiej powierzchni tylko jeden (!) jasny punkt - ten, w którym spełniona jest równość kątów padania i odbicia (Rys. 10.11). Jeśli obserwator się przesunie, ten jedyny jasny punkt przesunie się wraz z nim.

Powierzchnie rzeczywiste nie są nigdy idealnymi lustrami i odbijają cały stożek promieni wokół promienia idealnie odbitego: im mniej błyszcząca powierzchnia, tym większy kąt rozwarcia tego stożka:




*Rysunek 20.Powierzchnie lustrzane o różnym stopniu połysku: a) powierzchnia bardziej błyszcząca; b) powierzchnia mniej błyszcząca*

Na powierzchni błyszczącej, ale nie będacej idealnym lustrem obserwator widzi więc nie jeden jasny punkt, ale błyszczącą plamkę świetlną, tzw. odblask (ang. *highlight*), który przesuwa się wraz z obserwatorem i daje wrażenie połysku na powierzchni:




*Rysunek 21. Odblask na powierzchni ma wielkość zależną od stopnia połysku i przesuwa się wraz obserwatorem*

### 2.2.1 Metoda Phonga

Intensywność odbicia lustrzanego jest więc maksymalna wówczas, gdy obserwator stoi na kierunku promienia odbitego, zaś bardzo szybko maleje, gdy obserwator oddala się od tego kierunku. Zależność tę modeluje się zgodnie ze wzorem empirycznym zaproponowanym przez Bui-Tuong Phonga w 1975 r.:

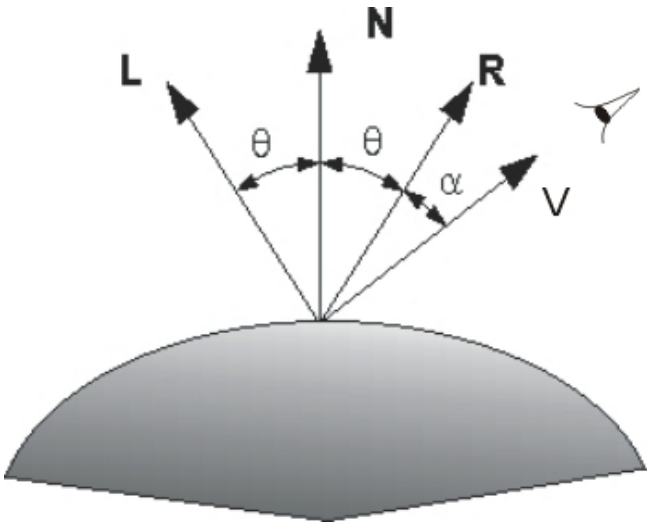
$$I_s = I_p k_s cos^n \alpha$$

gdzie:
$I_p$ - intensywność światła padającego
$I_s$ - intensywność odbicia lustrzanego (ang. *specular*)
$k_s$ - współczynnik odbicia lustrzanego, zawarty w przedziale <0, 1>, określający, jaka część światła padającego jest odbijana
$n$ - stopień połysku powierzchni.
Zamiast cosinusa kąta pomiędzy wersorem promienia odbitego **R** a wersorem skierowanym do obserwatora **V** można obliczyć ich iloczyn skalarny, analogicznie jak poprzednio:

$$I_s = I_p k_s (\mathbf{V} \cdot \mathbf{R})^n$$

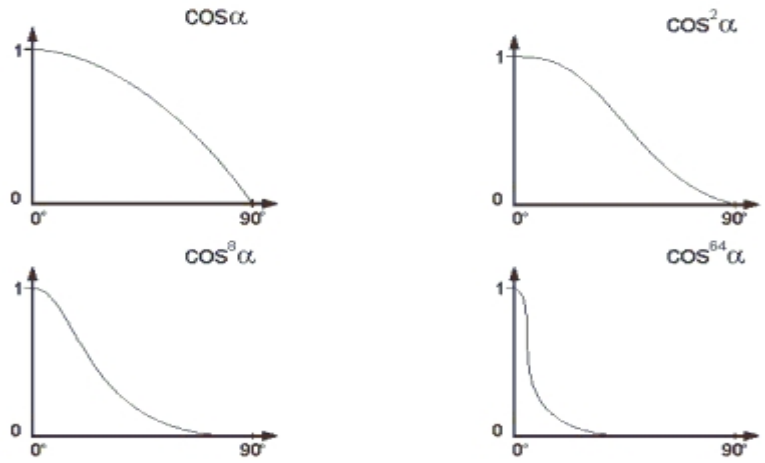
Sam zaś wersor promienia odbitego **R** wyznacza się z zależności:

$$\mathbf{R} = 2\mathbf{N}(\mathbf{N} \cdot \mathbf{L}) - \mathbf{L}$$



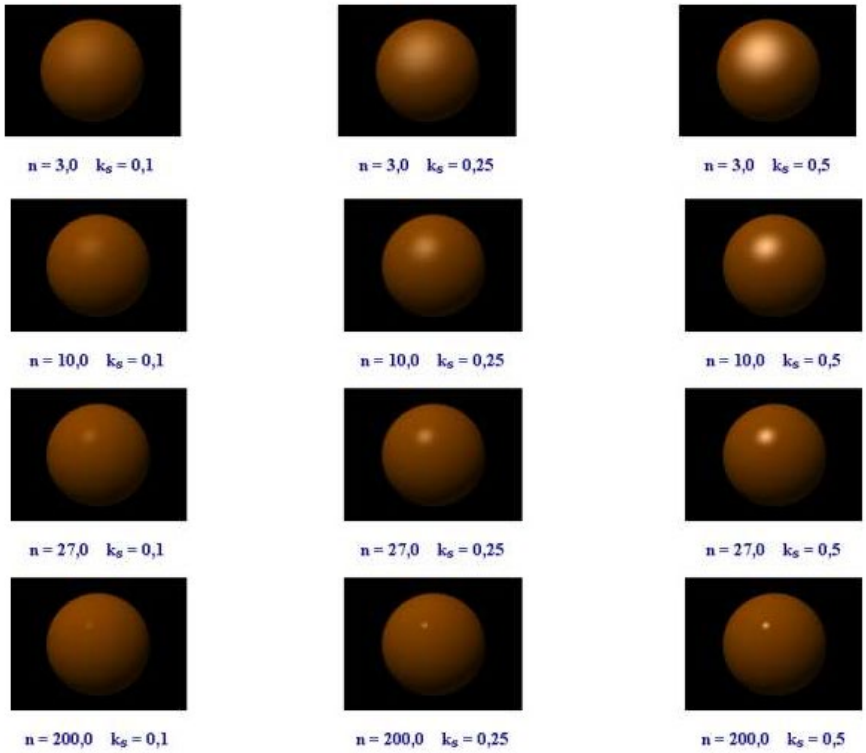

*Rysunek 22. Wersory niezbędne do wyznaczenia intensywności odbicia lustrzanego metodą Phonga*

Wykładnik potęgi w tym wzorze nazywany jest współczynnikiem połysku - im większa jego wartość, tym powierzchnia bardziej zbliżona jest do idealnego lustra, któremu odpowiada nieskończenie wielka wartość tego współczynnika:




*Rysunek 23. Wpływ współczynnika połysku na wartość potęgi funkcji cosinus*

Przykłady powierzchni o różnym stopniu połysku, zamodelowanych zgodnie z tymi zależnościami, obrazuje rysunek 24:




*Rysunek 24. Wpływ współczynnika połysku i współczynnika odbicia lustrzanego na wygląd oświetlonej powierzchni*

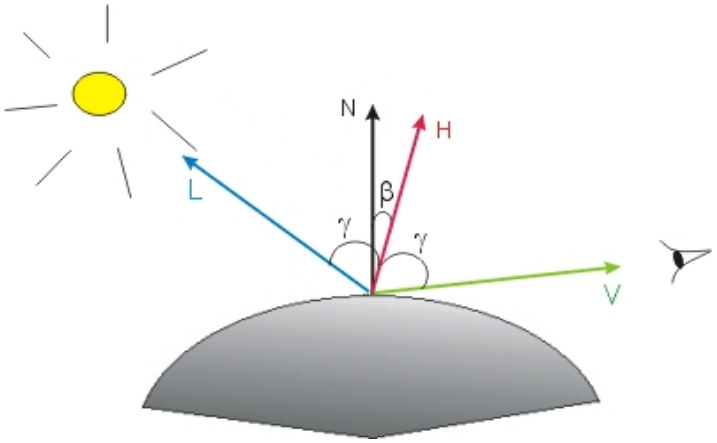
### 2.2.2 Ulepszona metoda Phonga

Tzw. ulepszona metoda Phonga polega na wykorzystaniu wersora **H** leżącego na dwusiecznej kąta pomiędzy wersorami skierowanymi do źródła światła i do obserwatora:

$I_s = I_p k_s (\mathbf{N} \cdot \mathbf{H})^n$

gdzie:

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|}$$




*Rysunek 25, Wersory niezbędne do wyznaczenia intensywności odbicia lustrzanego ulepszoną metodą Phonga*

Metoda nie jest dokładnym odpowiednikiem poprzedniej (kąt między wersorami **N** i **H** jest na ogół połową kąta między wersorami **V**i **R**) ale równie dobrze symuluje powierzchnię lustrzaną, a jej ulepszenie odczuwalne jest wówczas, gdy źródło światła i obserwator są w nieskończoności - wtedy wersor **H** jest stały. Dlatego w tej właśnie postaci najczęściej modeluje się połysk na powierzchni.

Modelując oświetlenie rzeczywistej powierzchni należy uwzględnić zarówno odbicie rozproszone, jak i lustrzane. Aby zaś obiekt dał się odróżnić od tła, uwzględnia się dodatkowo tzw. światło otoczenia:

$$I_0 = I_a k_a$$

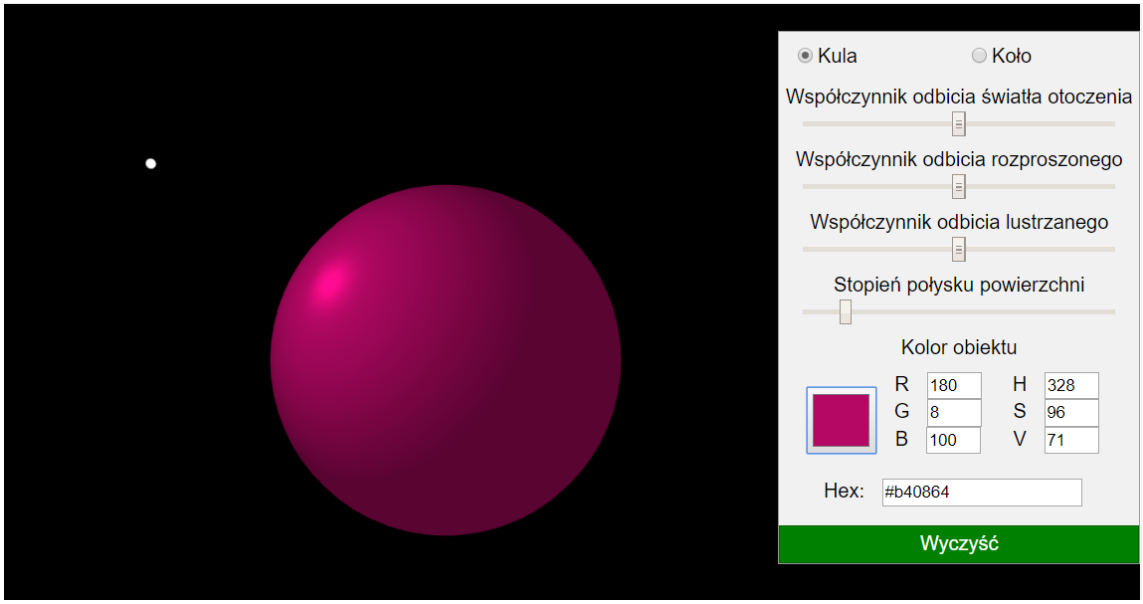
gdzie:

$I_a$ - intensywność światła otoczenia (ang. *ambient*), stała dla wszystkich obiektów modelowanej sceny

$k_a$ - współczynnik odbicia światła otoczenia, zawarty w przedziale <0, 1>, określający, jaka część światła otoczenia jest odbijana.

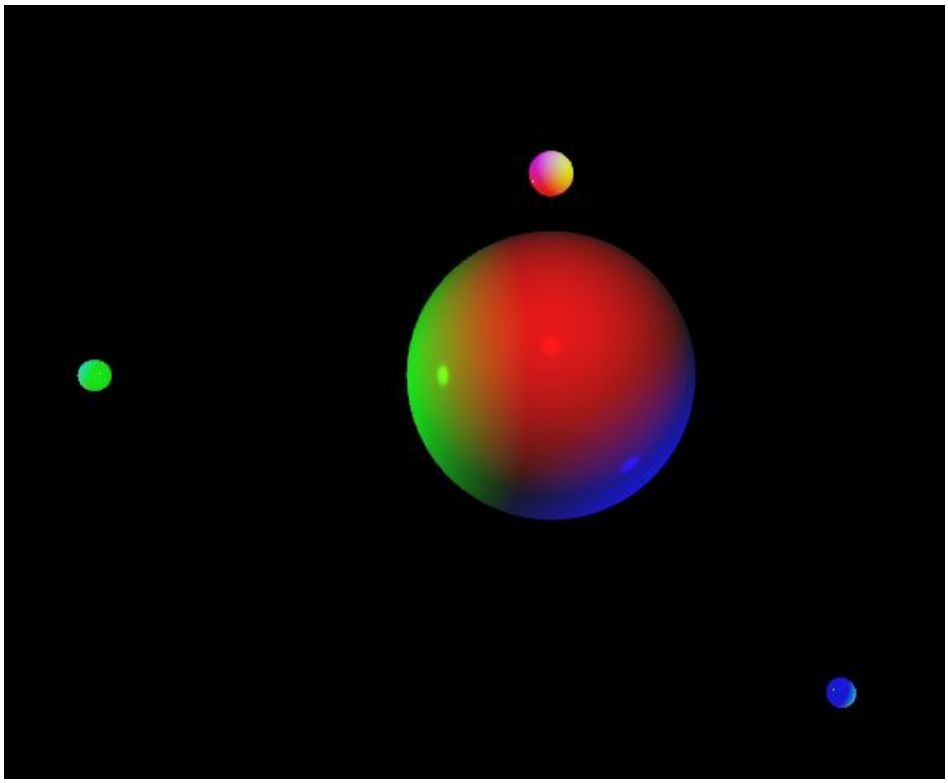
Łączny model odbicia światła zawiera wszystkie te trzy składowe:

$$I = I_a k_a + I_p k_d (\mathbf{N} \cdot \mathbf{L}) + I_p k_s (\mathbf{N} \cdot \mathbf{H})^n$$




*Rysunek 26 – **Aplikacja nr 2**. Odbicie rozproszone, lustrzane i otoczenia na powierzchni kuli o wybranej barwie. Powierzchnię można oświetlać z różnych kierunków, przesuwając myszką symboliczne białe słoneczko.*

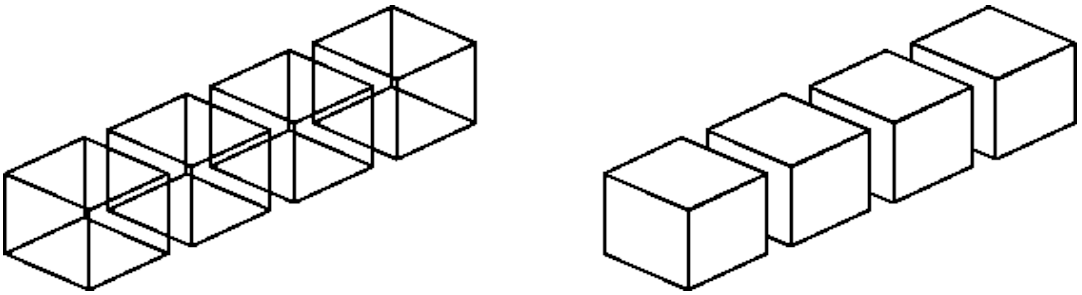
Należy podkreślić, że symulując odbicie rozproszone bierze się pod uwagę barwę obiektu, zatem obliczenia składowej rozproszonej wykonuje się dla każdej składowej barwy osobno. Natomiast odbicie zwierciadlane przyjmuje na ogół barwę światła padającego. Powyższe zasady można też zastosować do kilku źródeł światła - uzyskuje się wówczas dodatkowe efekty, ale trzeba dbać (wykonując odpowiednie przeskalowania), aby wynikowa jasność światła odbitego od powierzchni nie przekroczyła maksymalnej możliwej jasności piksela. Poniższa animacja, napisana w WebGL, obrazuje te możliwości:



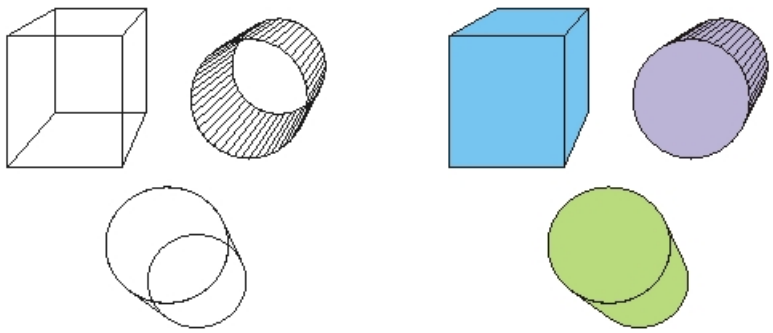
*Rysunek 27 – **Aplikacja nr 3.** Obiekt oświetlony trzema różnobarwnymi źrodłami światła. Odblask na powierzchni ma barwę światła padającego.*

# 3 Usuwanie punktów i ścian niewidocznych

Usuwanie punktów i ścian niewidocznych (lub odwrotnie: określanie, które są widoczne) jest fundamentalną potrzebą w grafice komputerowej, wystarczająco wyjaśnioną na poniższych rysunkach:



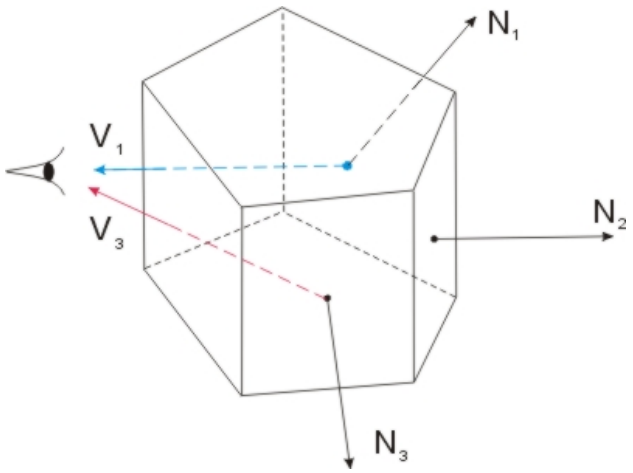
*Rysunek 28. Określanie powierzchni widocznych*



*Rysunek 29. Kolejny przykład określania powierzchni widocznych*

## 3.1 Usuwanie ścian tylnych

Jeśli obiekt jest wielościanem lub siatką wielościanową, nie ma potrzeby wykonywania obliczeń ani renderowania ścian niewidocznych dla obserwatora.

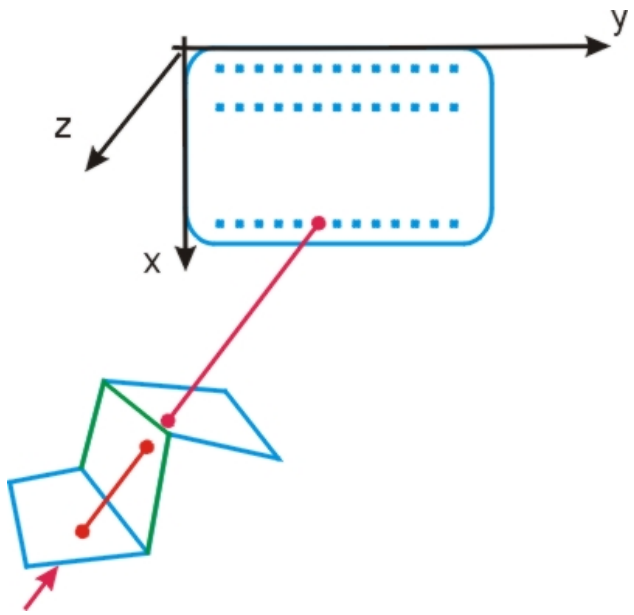



*Rysunek 30. Zasada usuwania ścian tylnych - wszystkie ściany z oznaczonymi wersorami normalnymi są zwrócone tyłem do obserwatora, więc mogą być od razu usunięte.*

Najprostszą metodą identyfikacji tylnej ściany jest sprawdzenie warunku określonego przez iloczyn skalarny:
$\mathbf{V} \bullet \mathbf{N} < 0$
W przypadku gdy ten warunek jest spełniony, ściana jest zwrócona tyłem do obserwatora, jest więc dla obserwatora niewidoczna.

## 3.2 Algorytm z z-buforem

Algorytm z z-buforem, nazywany też często buforem głębokości (ang. *depth buffer*), opracowany przez Catmulla, jest najprostszym algorytmem usuwania punktów niewidocznych, toteż jest powszechnie stosowany w realizacji sprzętowej (i również bardzo prosty w implementacji programowej). Może być używany do renderingu dowolnego obiektu, jeżeli dla każdego punktu rzutu można obliczyć barwę i wartość *z*. Algorytm z z-buforem nie wymaga, aby obiekty były wielokątami.
Algorytm z z-buforem wymaga korzystania obok pamięci obrazu z **z-bufora**, czyli tablicy o takiej samej liczbie pozycji, jak pamięc obrazu; w z-buforze pamiętana jest wartość z dla każdego piksela.
W fazie początkowej z-bufor jest wypełniany minimalną głębokością, co odpowiada zapamiętaniu wartości z dla tylnej ściany obcinającej scenę, czyli dla najdalej od obserwatora położonego elementu sceny, a do bufora obrazu jest wpisywana barwa tła. Największa wartość, jaka może być wpisana w z-buforze, reprezentuje z dla przedniej ściany obcinającej. Obiekty są przeglądane punkt po punkcie, np. wielokąty są przeglądane wierszami, powierzchnie są przeglądane wzdłuż linii stałego parametru. Kolejność przeglądania obiektów, wielokątów jest dowolna. Jeżeli rozpatrywany punkt (x, y) nie leży dalej od obserwatora niż punkt, którego barwa i głębokość są zapisane w buforach, to nowa barwa i głębokość zastępują stare wartości.




*Rysunek 31. Zasada działania z-bufora; wszystkie punkty zaznaczone na czerwono zamalowują ten sam piksel ekranu.*

W pamięci obrazu i w z-buforze są zapisywane informacje związane z największą wartością *z* napotkaną dotychczas dla każdego (x, y). Dlatego obiekty (wielokąty) pojawiają się na ekranie w kolejności, w jakiej są przetwarzane.
Warto podkreślić, że jeśli przyjmiemy inny niż przyjęty na Rys. 31 układ współrzędnych, w którym oś *z* będzie skierowana w głąb ekranu, wówczas należy odpowiednio zmodyfikować algorytm: z- bufor wypełnić największą możliwą wartością z i relację "większy niż " zamienić na relację "mniejszy niż".

Prześledźmy działanie z-bufora na poniższym rysunku, przy założeniu, że przetwarzane obiekty są trójkątami. W górnej części rysunku z lewej strony mamy pusty z-bufor. Pośrodku znajduje się tablica z określonymi wartościami *z* trójkąta. Po prawej stronie otrzymamy z-bufor po przeprowadzeniu operacji dodania trójkąta do z-bufora. W dolnej części z lewej strony mamy wynikowy z-bufor z górnej części. Pośrodku tablica z określonymi wartościami *z* drugiego trójkąta. Po prawej stronie mamy wynik dodania go do z-bufora:

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

+

| | | | |
|---|---|---|---|
| 5 | 5 | 5 | 5 |
| 5 | 5 | 5 | 0 |
| 5 | 5 | 0 | 0 |
| 5 | 0 | 0 | 0 |

=

| | | | |
|---|---|---|---|
| 5 | 5 | 5 | 5 |
| 5 | 5 | 5 | 0 |
| 5 | 5 | 0 | 0 |
| 5 | 0 | 0 | 0 |

| | | | |
|---|---|---|---|
| 5 | 5 | 5 | 5 |
| 5 | 5 | 5 | 0 |
| 5 | 5 | 0 | 0 |
| 5 | 0 | 0 | 0 |

+

| | | | |
|---|---|---|---|
| 3 | 0 | 0 | 0 |
| 4 | 3 | 0 | 0 |
| 7 | 6 | 3 | 0 |
| 8 | 7 | 6 | 5 |

=

| | | | |
|---|---|---|---|
| 5 | 5 | 5 | 5 |
| 5 | 5 | 5 | 0 |
| 7 | 6 | 3 | 0 |
| 8 | 7 | 6 | 5 |

*Rysunek 32. Zasada działania z-bufora na przykładzie dwóch trójkątów*

Jeżeli przetwarzamy wielokąty, to można uprościć obliczanie *z* dla każdego punktu w przeglądanym wierszu korzystając z faktu, że wielokąt jest płaski. Zazwyczaj w celu obliczenia *z* powinniśmy rozwiązać równanie płaszczyzny:

$Ax + By + Cz + D = 0$

dla zmiennej z ma postać

$$z = \frac{-D - Ax - Bz}{C}$$

Jeżeli w punkcie (x, y) z powyższego równania otrzymamy $z_1$, to w punkcie

$(x + \Delta x, z)$

wartość z wynosi

$$z_1 - \frac{A}{C}(\Delta x)$$

W przypadku, gdy znamy z(x, y), to do obliczenia

$z(x + 1, y)$

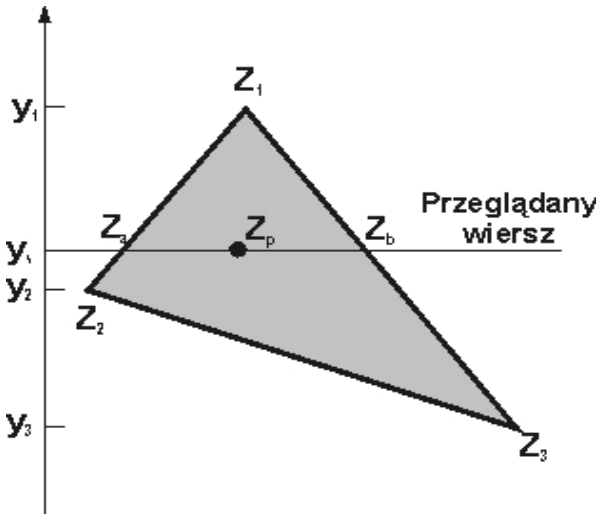
będziemy musieli wykonać tylko jedno odejmowanie, ponieważ iloraz

$\frac{\mathrm{A}}{\mathrm{C}} = const \quad i \qquad \Delta x = 1 \qquad \Delta$

W celu obliczenia pierwszej wartości *z* w następnym przeglądanym wierszu należy wykonać podobne obliczenie przyrostowe, zmniejszając *z* o

$\frac{B}{C}$ dla każdego $\Delta y$

W przypadku, gdy płaszczyzna nie została określona, albo gdy wielokąt nie jest płaski, to wówczas z(x, y) można określić interpolując współrzędne *z* wierzchołków wielokąta wzdłuż par krawędzi, a potem wzdłuż każdego przeglądanego wiersza.




*Rysunek 33.. Interpolacja wartości z wzdłuż krawędzi wielokąta i wierszy przeglądania.*

W powyższej sytuacji $z_a$ jest interpolowane między $z_1$ i $z_2$, $z_b$ między $z_1$ i $z_3$ a $z_p$ między $z_a$ i $z_b$.
Poszczególne punkty wyznaczymy z poniższych zależności:

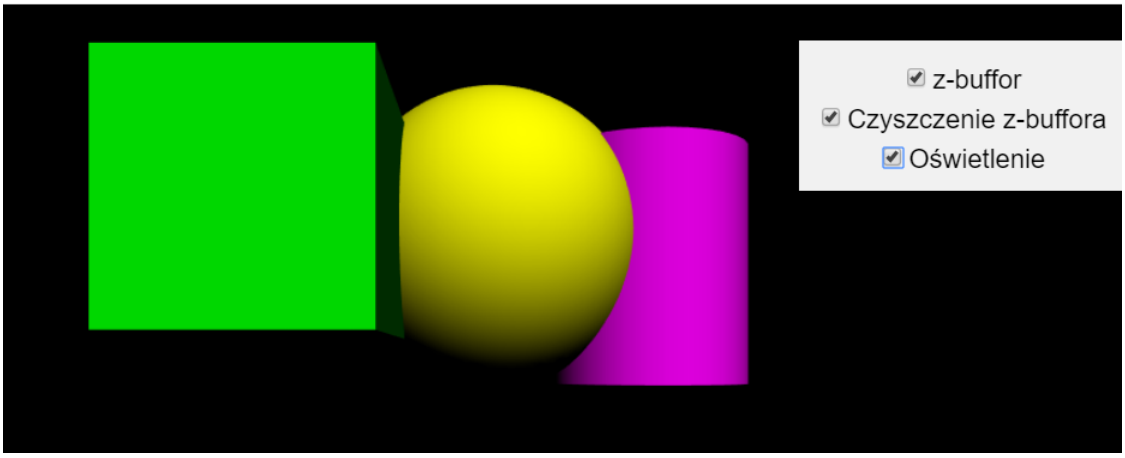
$$z_a = z_1 - (z_1 - z_2)\frac{y_1 - y_s}{y_1 - y_2}$$

$$z_b = z_1 - (z_1 - z_3)\frac{y_1 - y_s}{y_1 - y_3}$$

$$z_p = z_b - (z_b - z_a)\frac{x_b - x_p}{x_b - x_a}$$

Możemy tu także wykorzystać obliczenia przyrostowe. Zauważmy, że barwa piksela nie musi być obliczana, jeżeli warunkowe określanie widoczności piksela nie jest spełnione. Zatem jeżeli obliczenia związane z cieniowaniem są czasochłonne, można zwiększyć efektywność wykonując zgrubne sortowanie obiektów. Sortowanie pozwoli posegregować je według kryterium od najbliższego do najdalszego, a następnie będzie można najpierw wyświetlać najbliższe obiekty. Czas zajmowany przez obliczenia z użyciem z-bufora jest w przybliżeniu niezależny od liczby wielokątów w obiektach. Dzieje się tak, ponieważ średnio liczba pikseli pokrytych przez każdy wielokąt zmniejsza się wraz ze wzrostem liczby wielokątów w bryle widzenia.

Na zakończenie pouczająca aplikacja (Rysunek 34), która pokazuje efekty wyświetlania obiektów z z-buforem i bez:




*Rysunek 34 - **Aplikacja nr 4** ilustrująca efekty działania algorytmu z z-buforem. Efekty 3D zobaczymy po włączeniu oświetlenia i przemieszczeniu obiektów.*

Warto też zaobserwować, co się stanie, gdy z-bufor nie będzie inicjalizowany, i jakie będą efekty, gdy wyłączymy odświeżanie bufora obrazu.

# 4 Cieniowanie siatek wielościanowych

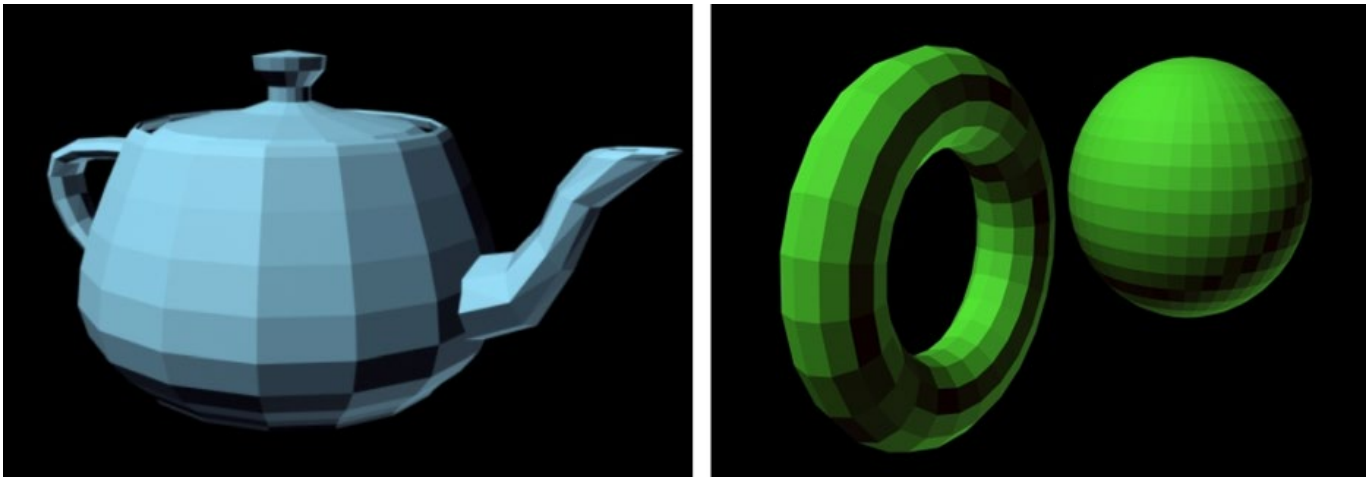
Z rozdziału 1 wynika, że dowolną pocieniowaną powierzchnię możemy uzyskać obliczając wersor normalny do powierzchni w każdym widocznym punkcie i stosując odpowiedni model oświetlenia w tym punkcie. Uzyskanie takiego efektu może być jednak bardzo kosztowne, zwłaszcza jeśli chcemy pocieniowaną powierzchnię poddawać przekształceniom w czasie rzeczywistym (obroty itp.). Jeżeli powierzchnia złożona jest z wielokątów, cieniowanie jej można uprościć i przyśpieszyć stosując metody interpolacji.

## 4.1 Cieniowanie płaskie

Najprostszy model cieniowania wielokąta, określany często jako **cieniowanie płaskie** (ang. *flat*) polega na wyświetlaniu go ze stałą wartością intensywności. Rozwiązanie to jest poprawne przy spełnieniu poniższych warunków:

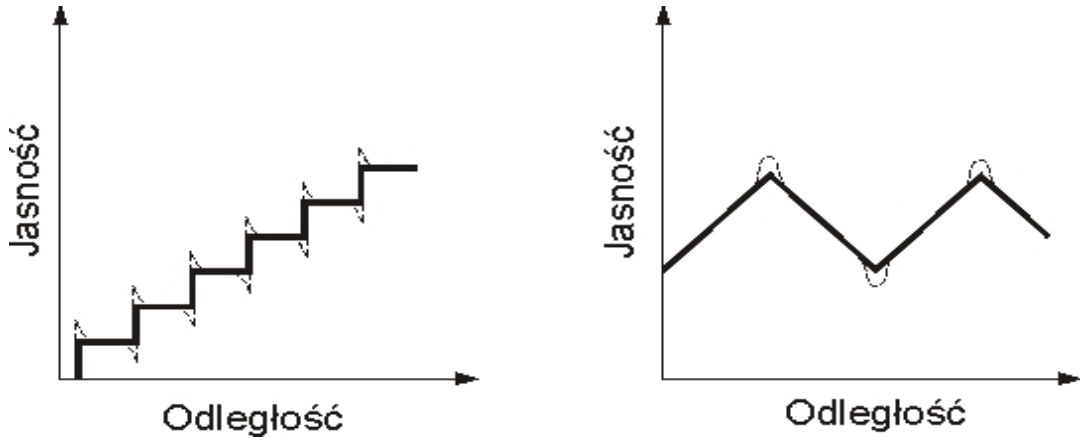
- Źródło światła jest w nieskończoności, zatem iloczyn **N·L** jest stały na całej powierzchni wielokąta.
- Obserwator jest w nieskończoności, zatem **N·V** jest stałe na całej powierzchni.
- Wielokąt reprezentuje powierzchnię modelowaną i nie jest aproksymacją powierzchni krzywoliniowej.

W przypadku aproksymacji powierzchni krzywoliniowej za pomocą siatki wielokątowej cieniowanie płaskie ukazuje wyraźnie strukturę tej siatki; co więcej, można odnieść wrażenie, że wielokąty nie są płaskie, a wgłębione pośrodku (Rys. 35).



*Rysunek 35. Cieniowanie płaskie.*

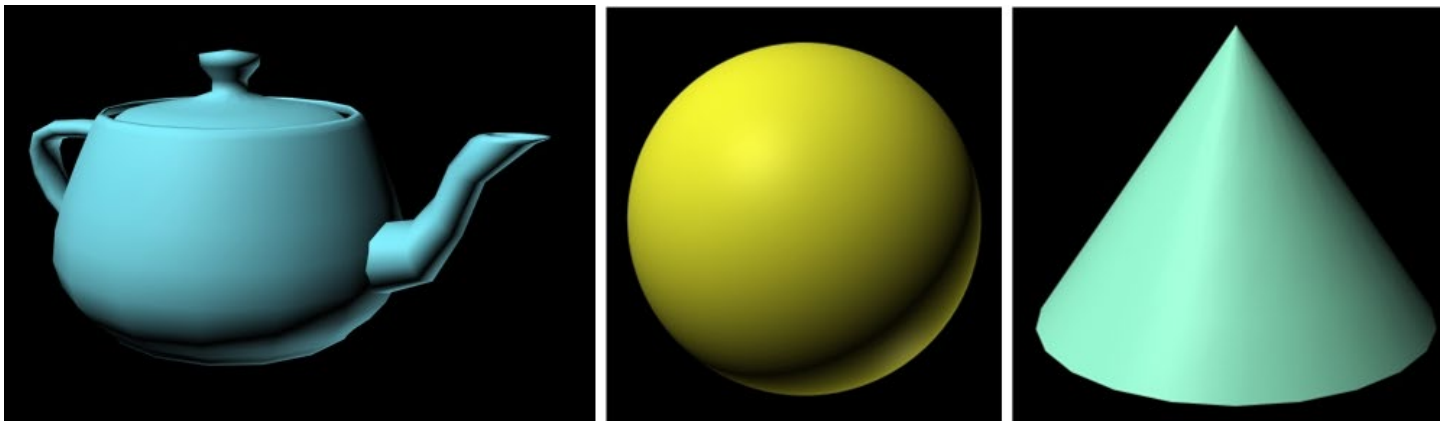
Ten tzw. efekt muszelkowatości jest wynikiem występowania *zjawiska Macha* , które uwypukla zmianę jasności na każdej krawędzi, w miejscu występowania nieciągłości amplitudy lub pochodnej. Efekt ten uwidacznia się na krawędzi między dwiema ścianami, tzn. ciemna ściana wygląda ciemniej, a jasna jaśniej.




*Rysunek 36. Przykład rzeczywistej i postrzeganej jasności dla efektu pasm Macha.*

## 4.2 Cieniowanie Gourada

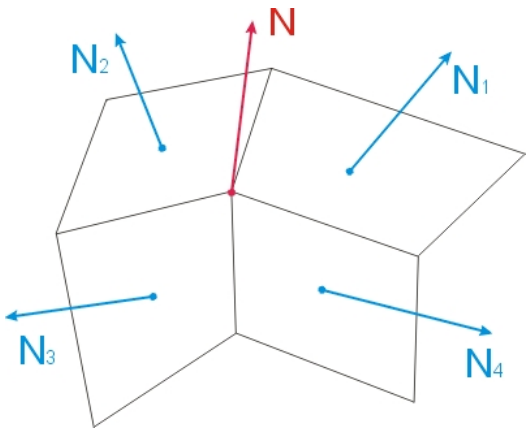
Jeżeli zależy nam, by ukryć siatkową strukturę powierzchni, można zastosować prostą zasadę interpolowania jasności, znaną jako **cieniowanie Gourauda** (Rys. 37).



*Rysunek 37. Cieniowanie Gourauda.*

Obiekty wyglądają na gładkie, chociaż można zauważyć jasne, mało widoczne pręgi na stożku; są one pasmami Macha powodowanymi przez szybkie, chociaż nie nieciągłe zmiany nachylenia krzywej jasności. cieniowanie Gourauda nie eliminuje więc całkowicie takich zmian jasności, ale wydatnie je zmniejsza.

W procesie cieniowania wymagana jest znajomość normalnej dla każdego wierzchołka siatki wielokątowej, które możemy obliczyć bezpośrednio z analitycznego opisu powierzchni. W przypadku, gdy normalne dla wierzchołków nie są zapisane z siatką i nie mogą być określone bezpośrednio dla bieżącej powierzchni, Gouraud zaproponował, aby je aproksymować na zasadzie uśredniania normalnych do powierzchni wszystkich ścian wielokątowych, dla których rozpatrywany wierzchołek jest wspólny.

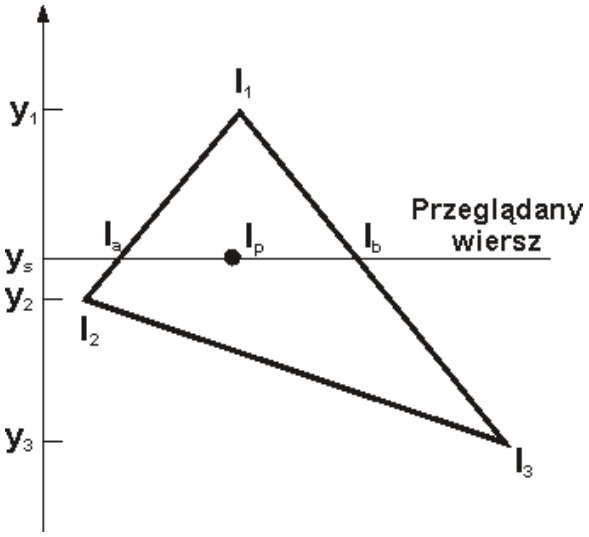



*Rysunek 38. Uśredniona normalna Nv*

Zatem uśredniona normalna $\mathbf{N_v}$ jest równa (dla przedstawionego przypadku):

$$\mathbf{N} = \frac{\mathbf{N_1} + \mathbf{N_2} + \mathbf{N_3} + \mathbf{N_4}}{|\mathbf{N_1} + \mathbf{N_2} + \mathbf{N_3} + \mathbf{N_4}|}$$

W kolejnym kroku cieniowania Gourauda należy znaleźć jasności w wierzchołkach wykorzystując do tego normalne w wierzchołkach za pomocą wybranego modelu oświetlenia. Ostatecznie każdy wielokąt jest cieniowany na zasadzie interpolacji liniowej między wierzchołkami wzdłuż każdej krawędzi, a następnie między krawędziami wzdłuż każdego przeglądanego wiersza, w taki sam sposób jak przy interpolowaniu wartości *z* w algorytmie z z-buforem.




*Rysunek 39. Interpolowanie jasności wzdłuż krawędzi wielokąta i przeglądanych wierszy.*

Poszczególne jasności wyznacza się za pomocą równań:

$$I_a = I_1 - (I_1 - I_2)\frac{y_1 - y_s}{y_1 - y_2}$$

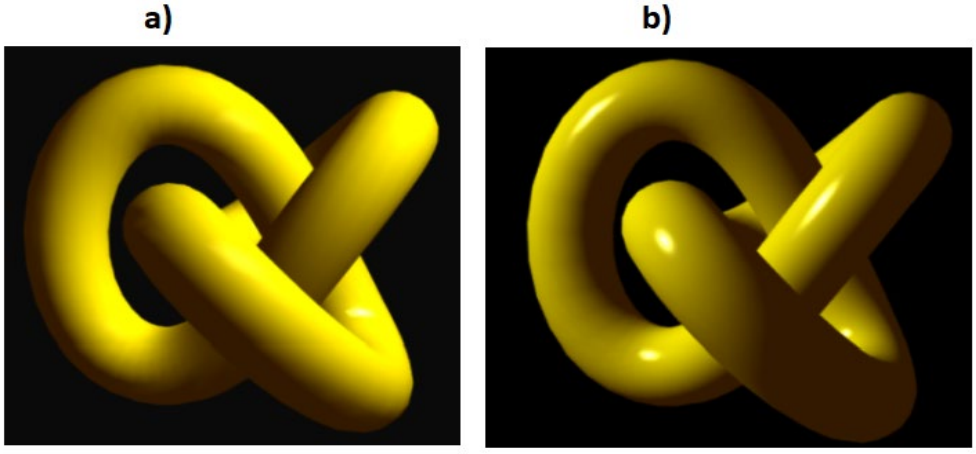
$$I_b = I_1 - (I_1 - I_3)\frac{y_1 - y_s}{y_1 - y_3}$$

$$I_p = I_b - (I_b - I_a)\frac{x_b - x_p}{x_b - x_a}$$

## 4.3 Cieniowanie Phonga

Gdybyśmy wykorzystali cieniowanie Gourauda w połączeniu z modelem odbicia lustrzanego, mogłoby dojść do gubienia odblasków pochodzących od źródeł światła. Dopiero metoda Phonga zapewnia poprawne odwzorowanie odblasków oraz eliminację efektu Macha.

Cieniowanie Phonga określane jest również jako cieniowanie **z interpolacją wektora normalnego**. W metodzie tej zamiast jasności interpolowany jest wektor normalny **N** do powierzchni. Wektory normalne są




*Rysunek 40. a) Cieniowanie Gourauda; b) cieniowanie Phonga - dla tego samego modelu oświetlenia (model Phonga, czyli powierzchnia błyszcząca)*

interpolowane wzdłuż krawędzi wielokąta na podstawie normalnych w wierzchołkach, obliczonych w razie potrzeby w sposób omówiony dla przypadku cieniowania Gourauda.

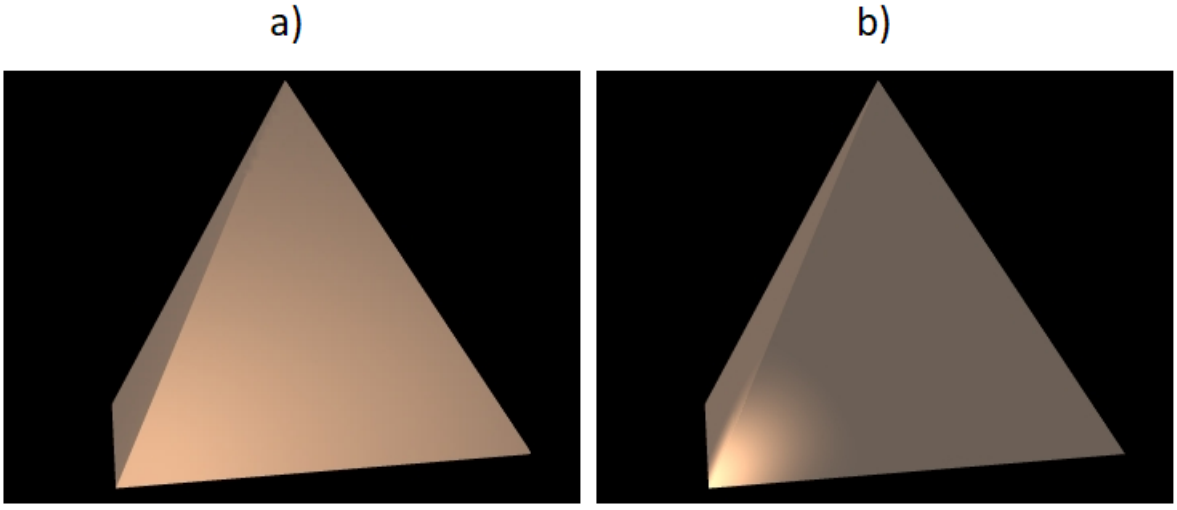
Interpolacja wzdłuż krawędzi może być dokonana za pomocą obliczeń przyrostowych. Wszystkie trzy składowe wektora normalnego są inkrementowane przy przejściu od przeglądanego wiersza do następnego przeglądanego wiersza. Dla każdego piksela wzdłuż przeglądanego wiersza interpolowana normalna jest normalizowana. Na koniec wykonuje się nowe obliczenie jasności za pomocą dowolnego modelu oświetlenia. Cieniowanie Phonga daje istotne polepszenie w stosunku do cieniowania Gourauda. Wynika to z lepszego reprodukowania odblasków na błyszczącej powierzchni.

## 4.4 Porównanie metod

Rozpatrzmy sytuację, w której n występującemu w członie $\cos^n\alpha$ cieniowania Phonga oświetlenia przypiszemy dużą wartość i dla jednego wierzchołka określimy bardzo mały kąt α, natomiast dla każdego z sąsiednich wierzchołków kąt α jest duży. Wynikają z tego następujące wnioski:

- Jasność związana z wierzchołkiem, dla którego kąt α określiliśmy jako niewielki, będzie odpowiednia dla odblasku, a dla innych jasności pojawią się wartości, nie odpowiadające odblaskowi.
- Po zastosowaniu metody cieniowania Gourauda jasność wzdłuż wielokąta jest liniowo interpolowana między jasnością odblasku a mniejszymi jasnościami sąsiednich wierzchołków, rozprzestrzeniając odblask po powierzchni wielokąta.

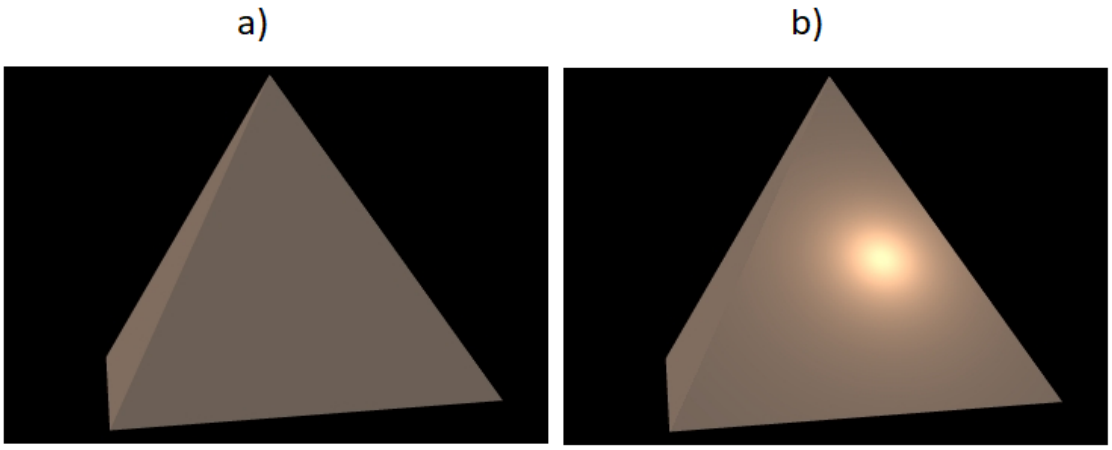
Przy wykorzystaniu normalnych interpolowanych liniowo do obliczenia czynnika $\cos^n\alpha$ w każdym pikselu, mamy do czynienia z ostrym spadkiem jasności odblasku (Rys. 41 b).




*Rysunek 41. Odblask w lewym wierzchołku: a) Cieniowanie Gourauda, b) cieniowanie Phonga*

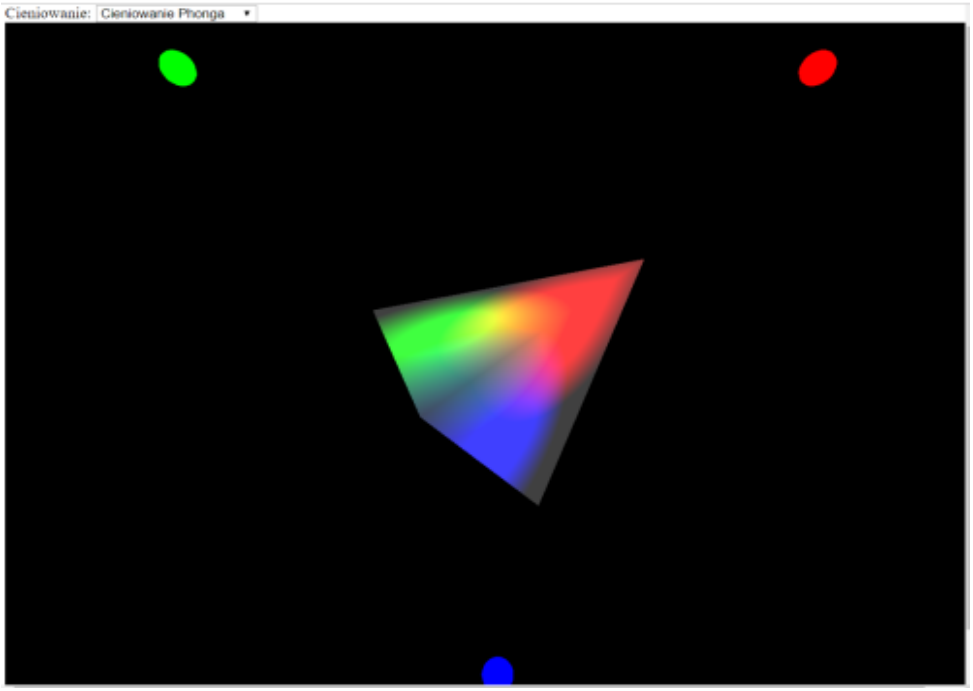
W przypadku, gdy odblask nie trafi na wierzchołek, cieniowanie Gourauda może je całkowicie pominąć. Dzieje się tak, ponieważ żaden punkt wewnętrzny nie może być jaśniejszy od najjaśniejszego wierzchołka, od którego zaczyna się interpolacja.

- Natomiast w metodzie cieniowania Phonga możliwe jest prawidłowe odwzorowanie odblasku wewnątrz wielokąta.




*Rysunek 42. Odblask wewnątrz wielokąta: a) Cieniowanie Gourauda, b) cieniowanie Phonga.*

Innym przykładem jest Aplikacja nr 5 (Rysunek 43). Każda ściana wirującego czworościanu jest wyświetlana stopniowo przechodzącymi w siebie barwami, wynikającymi z interpolacji barw w wierzchołkach.

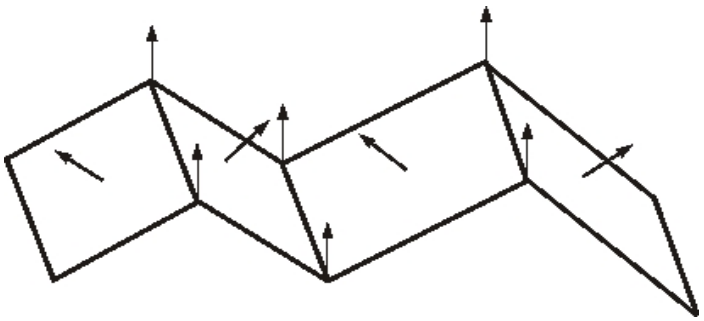



*Rysunek 43. **Aplikacja nr 5.** Cieniowanie czworokąta oświetlonego trzema kolorowymi źródłami światła metodą Gourauda i Phonga - do wyboru.*

Wykorzystanie dla modelu oświetlenia interpolacji wektora normalnego daje większe korzyści wizualne niż wykorzystanie interpolacji jasności, przy pominięciu współczynnika odbicia zwierciadlanego. Zaletą zastosowania interpolacji wektora normalnego jest w większości przypadków redukcja problemu pasm Macha. Jednakże w wyniku takiego postępowania znacznie zwiększa się koszt cieniowania w bezpośredniej implementacji, ponieważ interpolowana normalna musi być normalizowana za każdym razem, gdy jest używana w modelu oświetlenia.

Wyznaczone normalne związane z wierzchołkami mogą nie reprezentować dokładnie geometrii powierzchni. Rozpatrzmy poniższy książkowy przykład. Jeżeli obliczymy normalne związane z wierzchołkami na zasadzie uśredniania normalnych do powierzchni mających wspólny wierzchołek, to wszystkie normalne związane z

wierzchołkiem będą do siebie równoległe. W efekcie otrzymamy niewielką zmianę jasności sąsiednich powierzchni albo jej brak w przypadku cieniowania dla odległego źródła światła. Rozwiązaniem problemu może być dalsza dekompozycja wielokątów przed obliczeniem normalnej związanej z wierzchołkiem.



*Rysunek 44. Wszystkie normalne związane z wierzchołkami są do siebie równoległe*

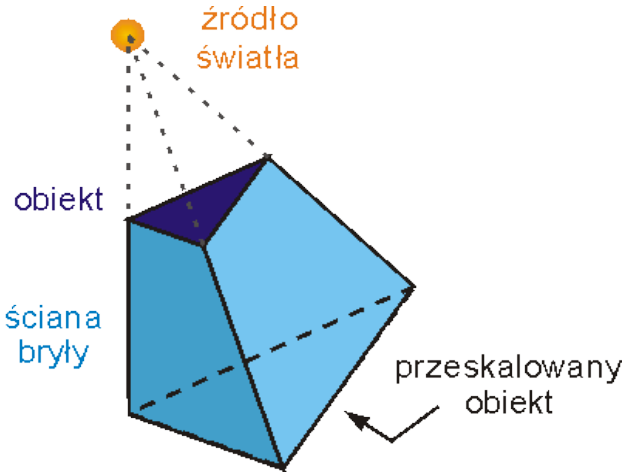
# 5 Nakładanie cieni

1. Wyznaczenie cieni rzucanych przez obiekt (poprzez rzutowanie powierzchni zwróconych przodem do źródła światła) oraz zacienionych fragmentów obiektów - zależne tylko od położenia źródła światła. Zacienione wieloboki wyznacza się jako zwrócone tyłem do źródła światła. Są osobno oznaczane i dołączane do struktury opisującej obiekt.
2. Wyznaczenie obrazu obiektu widzianego z pozycji obserwatora, z usunięciem wieloboków niewidocznych.

**DEFINICJA 1**

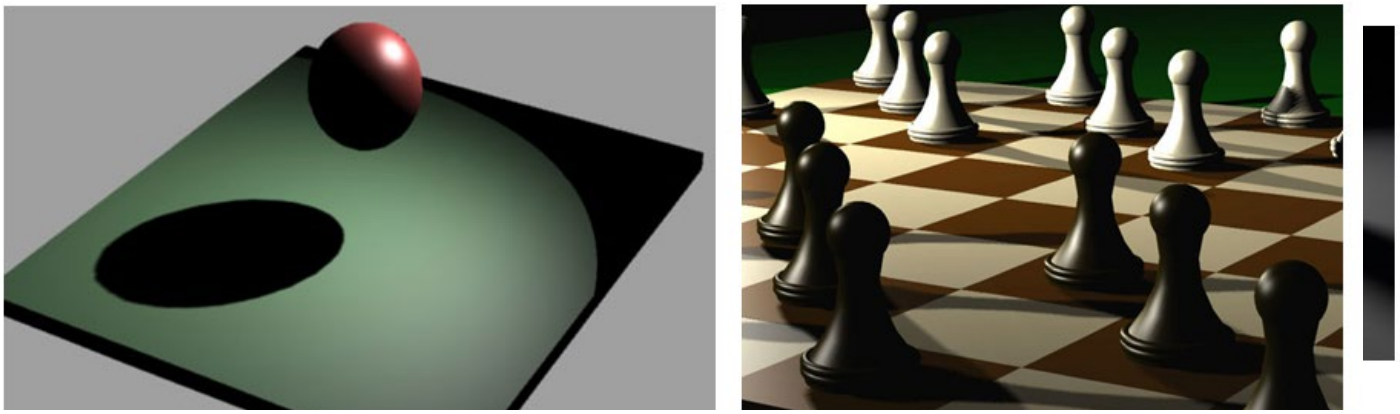
Zasada ogólna: powierzchnie widoczne ze źródła światła nie są w cieniu, natomiast te, które nie są widoczne, znajdują się w cieniu.

.




*Rysunek 45. Metoda z bryłą cienia*

+2.



*Rysunek 46. Przykłady generowania cieni w  Rhinoceros.*

# 6 Odwzorowywanie tekstur

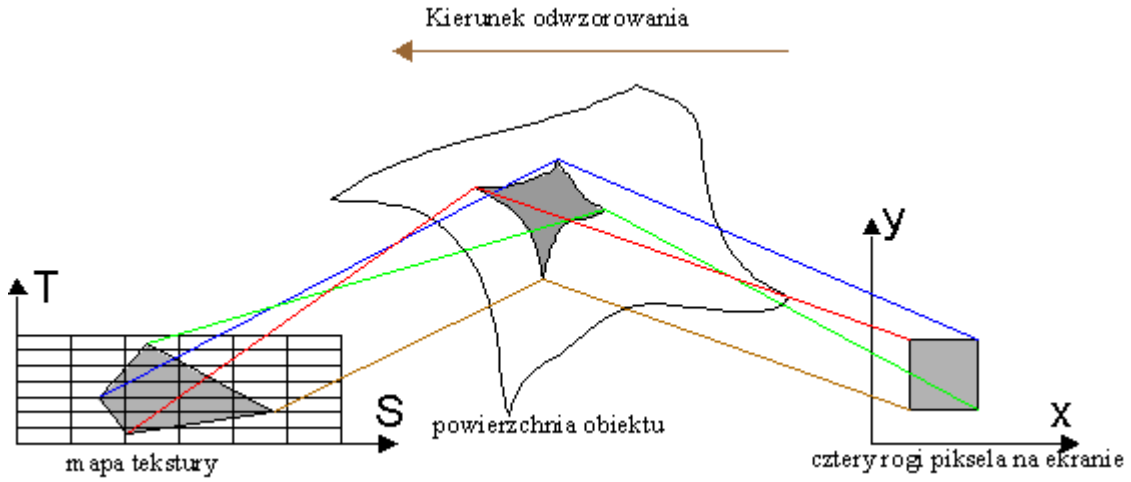
Odwzorowanie tekstury może służyć do modyfikacji pewnych właściwości obiektu, takich jak:

- Kolor powierzchni
- Wektor normalny powierzchni
- Współczynnik połysku powierzchni
- Współczynnik przezroczystości
- Cienie, przesunięcie powierzchni
- Lokalne układy współrzędnych

Współrzędne tekstury są w takich przypadkach związane ze współrzędnymi obiektu.
Tekstura może byc też wykorzystywana do opisu oświetlenia sceny. Nie jest wówczas związana z żadnym obiektem sceny, ale z umowną sferą o nieskończonym promieniu, w której centrum znajduje się scena. Ten rodzaj tekstury nazywany jest odwzorowaniem środowiska.
Przy wykorzystywaniu tekstur w systemach komputerowej generacji obrazu pojawiają się następujące problemy, które dalej zostaną omówione:

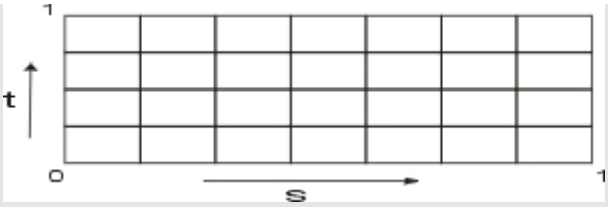
- Problem generowania wzorca tekstury
- Problem odwzorowania geometrycznego tekstury
- Problem filtrowania tekstury

Istnieją dwa sposoby opisu tekstury w systemach komputerowej generacji obrazu, tzn.:
proceduralny - w postaci funkcji matematycznej lub zestawu parametrów dla określonej klasy funkcji
jawny - w postaci tablicy wartości funkcji tekstury
Szeroko rozpowszechnionymi klasami funkcji stosowanymi do proceduralnego generowania tekstur są szeregi fal sinusoidalnych, funkcje fraktalne czy procesy stochastyczne. Każda funkcja opisana proceduralnie może być stablicowana dla określonej części dziedziny funkcji i z określoną precyzją (tzw. gęstością próbek).
Po wygenerowaniu tekstury powstaje problem jej poprawnego odwzorowania na obraz obiektu na ekranie. W procesie odwzorowania tekstury na obraz obiektu możemy wyróżnić dwa etapy:
*Parametryzację* - etap polegający na określeniu przekształcenia przestrzeni tekstury w przestrzeń obiektu
*Projekcję* - etap polegający na odwzorowaniu obiektów na przestrzeń ekranu
Powszechnie stosowane odwzorowanie przestrzeni tekstury 2D na powierzchnię obiektu wykonywane jest najczęściej zgodnie ze schematem:



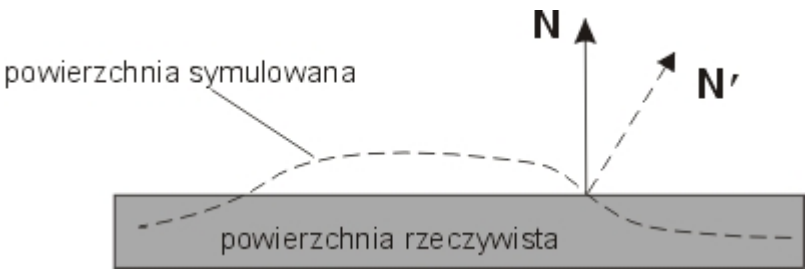

*Rysunek 48. Odwzorowanie tekstury od piksela do mapy (tablicy) tekstury*

Przeglądanie w przestrzeni ekranu, nazywane także *odwzorowaniem odwrotnym*, jest najczęściej stosowaną metodą, w której dla każdego piksela w przestrzeni ekranu jest wyznaczany jego obraz źródłowy w przestrzeni tekstury. Jest on następnie filtrowany dając wartość tekstury dla danego piksela. Przy stosowaniu tej metody wymaga się znajomości odwzorowania odwrotnego, tzn. z przestrzeni ekranu do przestrzeni tekstury, niezbędnego do wyznaczenia obrazu źródłowego piksela. Ponadto niezbędny jest bezpośredni dostęp do tablicy tekstury.
Odwzorowanie odwrotne można również wykonać na etapie parametryzacji, przechodząc z przestrzeni tekstury do przestrzeni obiektu.




**Rys. 11.7. Tablica opisująca teksturę, złożona z tzw. tekseli (elementów tekstury) o współrzędnych (s,t)**

Ten rodzaj odwzorowania, powszechnie znany pod nazwą "bump mapping" jest szczególnie efektowny, a zarazem bardzo prosty od strony algorytmicznej. Efekt pofałdowań na powierzchni uzyskuje się wyłącznie poprzez zaburzenia wersora normalnego do powierzchni - wynikająca stąd zmiana odbicia rozproszonego i ew. lustrzanego wystarcza do uzyskania wrażenia wklęsłych i wypukłych miejsc na powierzchni.




*Rysunek 49. Zasada symulowania wypukłości na płaskiej powierzchni*

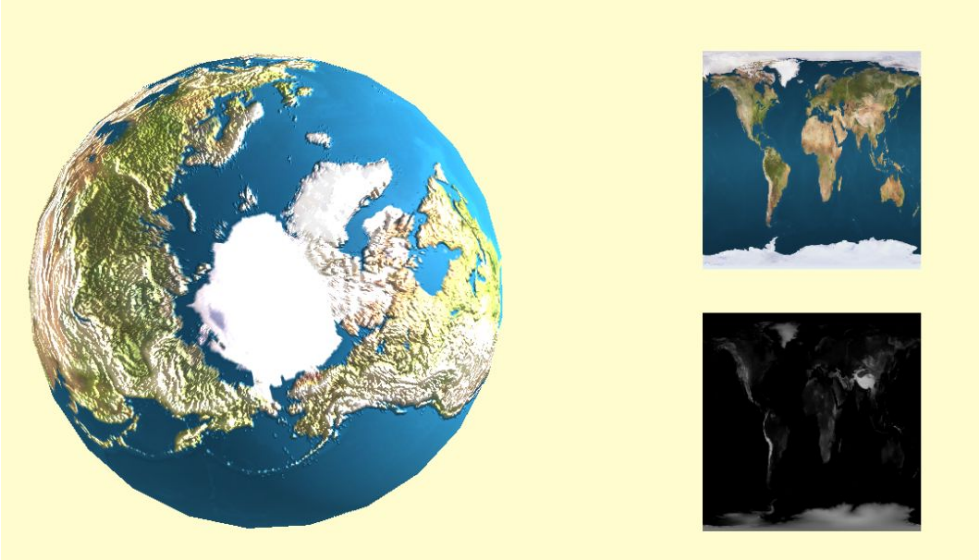
Dla powierzchni $\mathbf{P}(\boldsymbol{u},\boldsymbol{v})$ definiuje się funkcję zaburzającą $\mathbf{T}(\boldsymbol{u},\boldsymbol{v})$, tzw. mapę wypukłości. Funkcja ta określa zaburzenia powierzchni w kierunku wektora normalnego:

$$\mathbf{P}'(\boldsymbol{u},\boldsymbol{v}) = \mathbf{P}(\boldsymbol{u},\boldsymbol{v}) + \mathbf{T}(\boldsymbol{u},\boldsymbol{v})\frac{\mathbf{N}}{|\mathbf{N}|}$$

Po wyznaczeniu wektora normalnego dla tak zaburzonej powierzchni i pominięciu mało znaczących członów otrzymuje się prostą zależność:

$$\mathbf{N}'(u,v) = \mathbf{N} + \frac{\partial T}{\partial u}\frac{\mathbf{N}\times\partial\mathbf{P}/\partial v}{|\mathbf{N}|} + \frac{\partial \mathbf{T}}{\partial v}\frac{\mathbf{N}\times\partial\mathbf{P}/\partial u}{|\mathbf{N}|}$$

Funkcją zaburzającą $\mathbf{T}(\boldsymbol{u},\boldsymbol{v})$ może być dowolna funkcja, dla której da się policzyć pochodne. Jeśli wzór wypukłości jest zdefiniowany niematematycznie, za pomocą tablicy, to wartości pośrednie wyznacza się stosując biliniową interpolację, a pochodne funkcji zaburzającej - metodą różnic dzielonych. W szczególności funkcją zaburzającą może być dowolna mapa bitowa, pełniąca podwójną rolę - jako wzór płaski i jako mapa zaburzeń. Działanie algorytmu bump mapping można przeanalizować w następującej aplikacji:

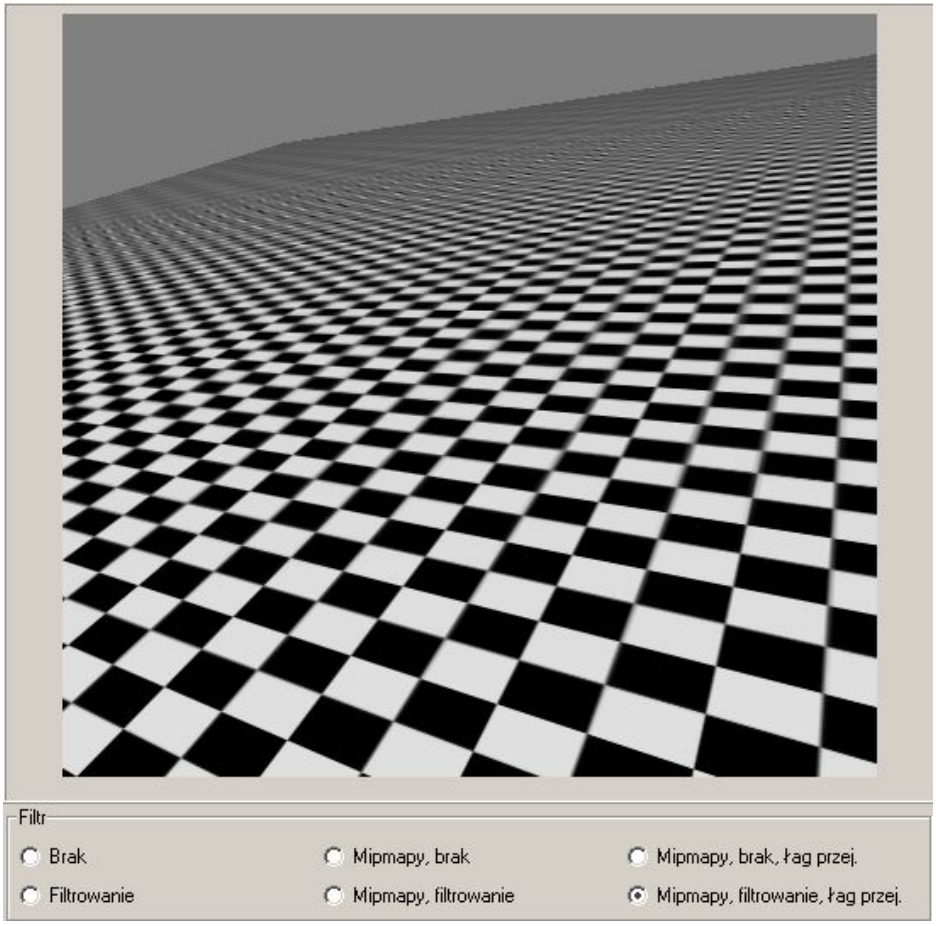


*Rysunek 50. Odwzorowywanie wypukłości na powierzchni kuli.*

### 6.1.1 Displacement mapping

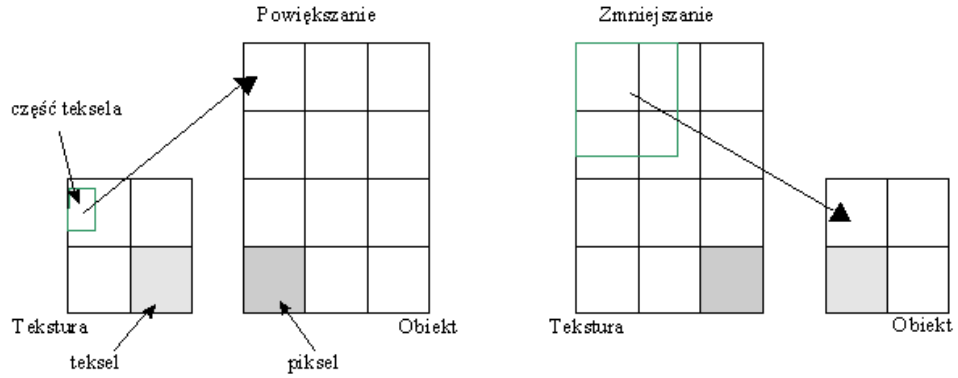
Bardziej zaawansowaną metodą odwzorowania wypukłości jest metoda displacement mapping, czyli odwzorowywania przemieszczeń. Wierzchołki renderowanej geometrii (siatki wierzchołków) podlegają modyfikacji zgodnie z mapą przemieszczeń – co pozwala kształtować rzeczywiste wypukłości, zasłaniające się wzajemnie, rzucające cienie itp. – jest to technika znacznie kosztowniejsza niż bump mapping, ale znacznie podnosząca realistyczny wygląd sceny.


Oba zaprezentowane powyżej aplety dawały możliwość tzw. filtrowania tekstur. Bez tej możliwości trudno uzyskać poprawne efekty w sytuacji, gdy wiele tekseli (elementów mapy tekstury) trafia w wyniku odwzorowania w jeden i ten sam obszar obiektu. Dzieje sie tak na przykład przy rzutowaniu perspektywicznym; Kolejny aplet ukazuje najbardziej typową sytuację:



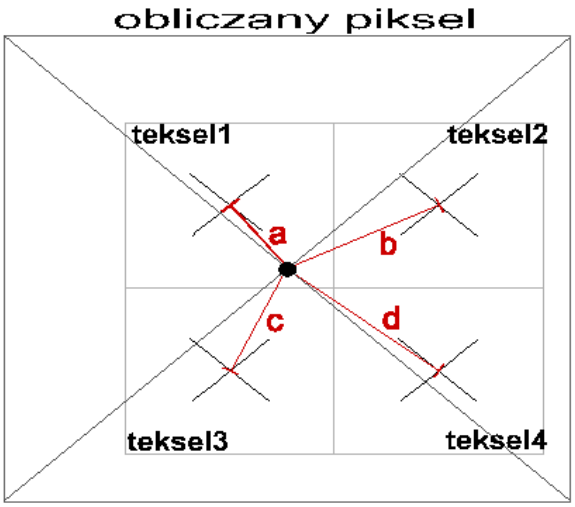

*Rysunek 51. Różne rodzaje filtrowania tekstur dostępne w OpenGL*

Może też zdarzyć się sytuacja odwrotna: gdy jeden teksel pokrywa kilka pikseli na ekranie Powyższe efekty odwzorowywania nazywają się, odpowiednio, zmniejszaniem (minifikacją) i powiększaniem (magnifikacją) tekstur:




*Rysunek 53, Powiększanie (magnifikacja) i zmniejszanie (minifikacja) tekstur*

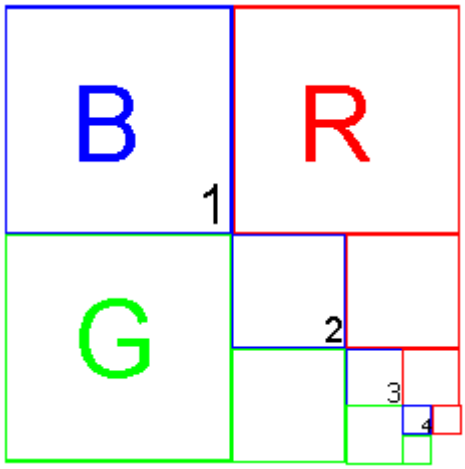
Istnieją też sytuacje, dla których zastosowanie pojedynczej tekstury dla wszystkich obiektów w scenie może powodować pewne problemy jej renderowania. Przykładem takiej sceny może być scena animowana, gdzie rozmiary i kształty obiektów zmieniają się wraz ze zmianą pozycji obserwatora w scenie (jego odległości i kąta




*Rysunek 52. Kolor teksturowanego piksela jest wynikiem obliczenia wartości średniej ważonej kolorów czterech elementów tekstury , które znajdują się najbliżej środka teksturowanego piksela. Współczynniki wagowe poszczególnych tekseli wyznaczane są w zależności od ich odległości od analizowanego piksela.*

patrzenia). Ponieważ tekstury są powiązane z obiektami na które są nakładane, to efekt teksturowania musi być zgodny z zachowaniem obiektów sceny. I tak w miarę zmniejszania się obiektu na ekranie, mapowana tekstura staje się mniej wyraźna i widoczna. Zatem obiekt oddalony od obserwatora, nie musi być wyświetlany z dużą dokładnością (rozdzielczością), ponieważ i tak nie zostanie to zauważone. Z drugiej strony rysowanie odległych obiektów teksturą o dużej rozdzielczości zmniejsza szybkość renderowania sceny.

Aby umożliwić równie szybkie teksturowanie wszystkich obiektów sceny niezależnie od ich rozmiaru i położenia (w stosunku do obserwatora), stosowane są tzw. mipmapy. Termin *mip*pochodzi z łacińskiego "multum in parvo", co oznacza "wiele w małym", w tym wypadku - wiele danych w małej przestrzeni. Mipmapy tworzą zbiór tekstur charakteryzujących się tym, że każda następna (o mniejszym rozmiarze) ma mniejszą rozdzielczość tzn. zawiera mniej informacji o obrazie tekstury. Określają one zestaw tablic przedstawiających ten sam obraz. Mipmapy są tak skonstruowane, że informacje o obrazie zostają upakowane w jak najmniejszym obszarze pamięci, tworząc tzw. piramidę mipmap:

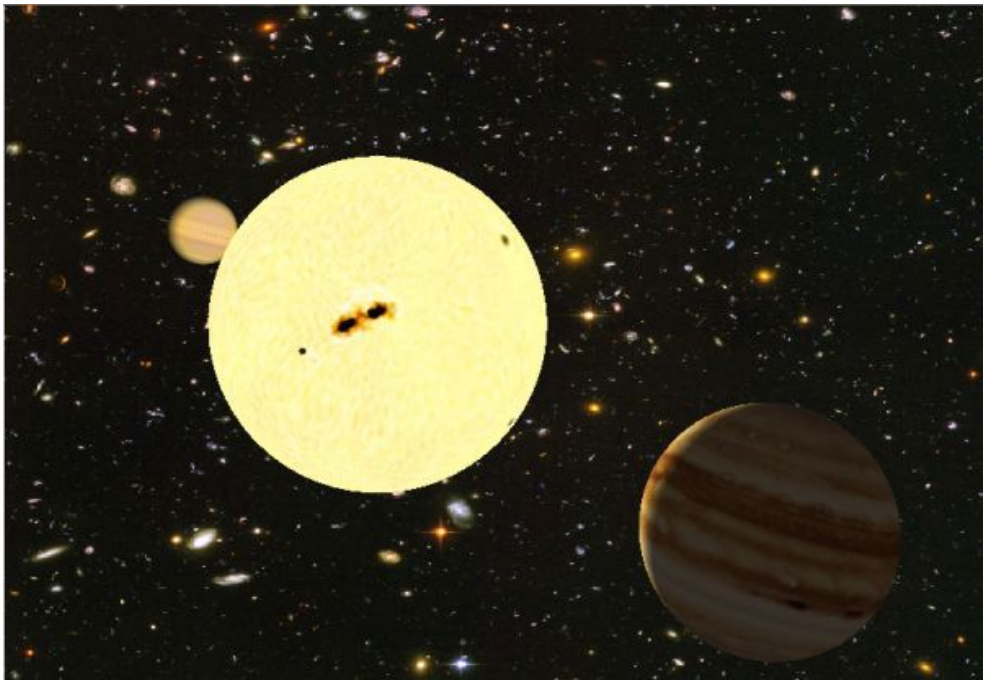



*Rysunek 54. Piramida mipmap.*

Podejście takie realizowane jest przez bibliotekę OpenGL, która automatycznie wybiera mniejszą teksturę z piramidy, jesli tylko obraz obiektu się zmniejsza.

Zaawansowany efekt korzystania z tekstur w bibliotece OpenGL przedstawia ostatni, najbardziej rozbudowany aplet przedstawiający układ słoneczny. Aplet prezentuje wiele z poprzednio omówionych zagadnień:

- Oświetlenie (słońce jest źródłem punktowym światła)
- Teksturowanie
- Filtrowanie (mipmapy)
- Przezroczystość (pierścienie planet)

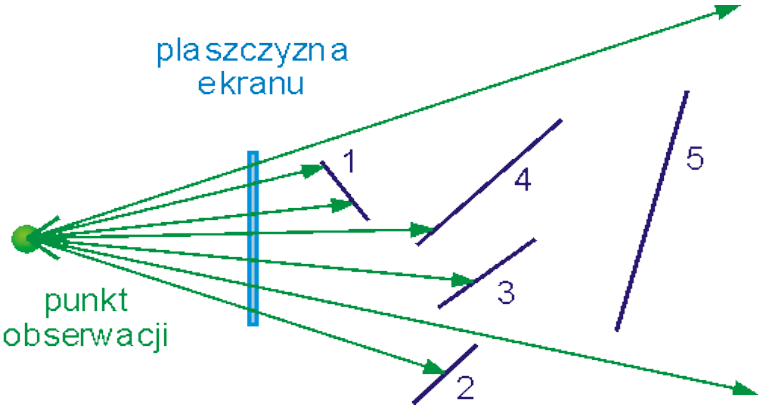


*Rysunek 55 Układ słoneczny.*

# 7 Oświetlenie globalne

## 7.1 Ray casting, czyli rzucanie promieni

Ray casting, czyli metoda "rzucania promieni" to jeszcze jedna, dość oczywista metoda pozwalająca na usuwanie punktów niewidocznych, czyli wyświetlanie tylko widocznych punktów danej sceny 3D. Metoda polega na tym, że od obserwatora, umieszczonego w wybranym punkcie, prowadzi się w kierunku sceny promienie przechodzące przez każdy piksel ekranu. Dla każdego promienia wyznaczany jest najbliższy obserwatorowi punkt przecięcia tego promienia z jednym z obiektów sceny, które ten promień przebija, i punkt ten jest wyświetlany w odpowiedniej barwie, właściwej dla tego obiektu i zależnej od wersora normalnego do powierzchni w danym punkcie przebicia jej promieniem. Jeśli promień przechodzący przez jakis piksel nie przetnie żadnego obiektu, piksel ten wyświetlany jest w kolorze tła.

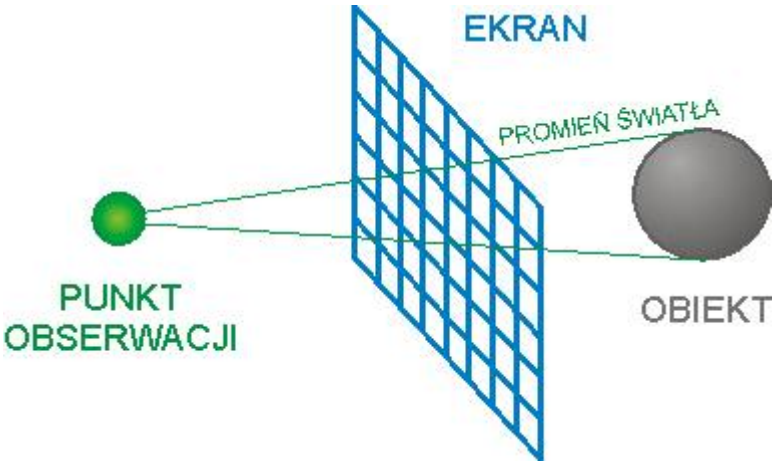



*Rysunek 56. Metoda rzucania promieni; obiekty są ponumerowane i zaznaczone schematycznie.*

Metoda wymaga stosowania algorytmów przecięcia obiektów z prostoliniowym promieniem. Jest więc oczywiste, że złożoność obliczeniowa algorytmu rośnie wraz ze złożonością opisu matematycznego obiektów renderowanej sceny. Najprostsze w obliczeniach jest szukanie punktów przecięcia prostej z płaszczyzną czy kulą, bardziej pracochłonne jest poszukiwanie przecięć z wielościanami i siatkami wieloboków, zaś w przypadku powierzchni parametrycznych konieczne jest stosowanie iteracyjnych metod rozwiązywania nieliniowych równań określających współrzędne punktu przebicia. Ponieważ metoda musi być wykonywana dla każdego piksela ekranu, wymaga stosowania najbardziej efektywnych metod. Możliwość wykorzystania pracy wieloprocesorowej jest tu szczególnie istotna: każdy procesor wykonuje obliczenia tylko dla wybranego fragmentu ekranu. Algorytmiczne metody przyspieszenia obliczeń zostaną podane w ogólnym zarysie w następnym rozdziale.
Metoda rzucania promieni nadaje się również do wyświetlania brył CSG: punkt najbliższy obserwatorowi wyznacza się porządkując punkty przebicia brył składowych wzdłuż promienia i rozwiązując wzdłuż tego promienia jednowymiarowe operacje boolowskie.

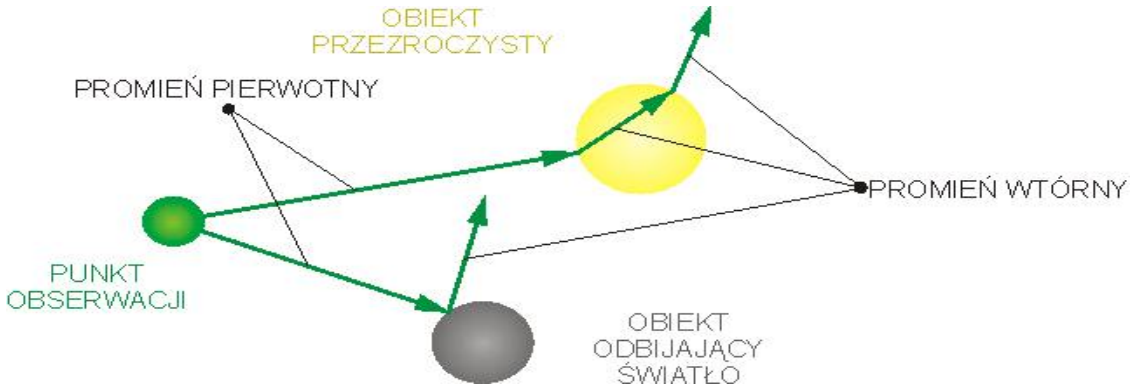
## 7.2 Ray tracing, czyli śledzenie promieni

**Metoda śledzenia promieni** jest uogólnieniem metody rzucania promieni. Również i tu promienie prowadzone są od obserwatora przez każdy piksel ekranu, ale śledzenie ich biegu nie kończy się w momencie przebicia pierwszego napotkanego po drodze obiektu, tylko trwa nadal, bo promień odbity od danego obiektu może trafić w kolejny obiekt, odbić się znowu itd. Dzięki odwrotnemu śledzeniu promieni, od oka do źródła światła, a nie ze źródła światła do oka, rozpatrywane są tylko te promienie światła, które docierają do oka odbiorcy i tworzą obraz na ekranie komputera. Unika się w ten sposób niemożliwej do zrealizowania w praktyce analizy wszystkich promieni wychodzących ze źródła światła, z których tylko bardzo nieliczne docierają do obserwatora.
Rozpatrzmy działanie algorytmu po kolei.
Pierwszym etapem w algorytmie śledzenia promieni jest generacja promieni w punkcie obserwacji i poprowadzenie ich przez każdy piksel ekranu w kierunku sceny, analogicznie jak w metodzie ray casting. Te promienie są nazywane **promieniami pierwotnymi** (*ang.: primary ray*).




*Rysunek 57. Generowanie promieni pierwotnych.*

Kolejnym etapem, podobnie jak w metodzie poprzedniej, jest znajdowanie punktów zderzenia się promienia z obiektami na scenie - testowany jest każdy obiekt. W przypadku gdy do przecięcia nie dochodzi, promień wychodzi poza scenę i piksel na ekranie przybiera kolor tła. Jeżeli zaś przecięcie nastąpi, spośród możliwych punktów przecięcia wybierany jest punkt najbliższy obserwatorowi. W miejscu zderzenia się promienia z obiektem powstają **promienie wtórne** (*ang.: secondary rays*), które w zależności od właściwości obiektu (przezroczystość, współczynnik odbicia) biegną dalej zgodnie z zachowaniem praw odbicia i załamania.
Promienie wtórne traktowane są jak promienie pierwotne: również są testowane z innymi obiektami znajdującymi się na scenie i podobnie jak promienie pierwotne mogą być źródłem kolejnych promieni wtórnych. Proces generowania promieni wtórnych trwa, dopóki wszystkie promienie nie wyjdą poza scenę (trafią w umowne tło) lub algorytm nie osiągnie odpowiedniego poziomu rekurencji (określonego uprzednio w algorytmie). Ta metoda, polegająca na tworzeniu i śledzeniu promieni wtórnych, jest nazywana **rekursywną metodą śledzenia promieni** (*ang.: recursive raytracing*), lecz w skrócie mówi się o niej po prostu jako o metodzie śledzenia promieni.




*Rysunek 58. Promienie pierwotne i wtórne.*

Rekurencję zawartą w algorytmie śledzenia promieni ilustruje poniższy schemat algorytmu: funkcja TraceRay wywołuje samą siebie aż do osiągnięcia maksymalnej głębokości rekurencji. Pokazano tu najważniejsze kroki obliczeniowe metody:

**DEFINICJA 2 – Algorytm ray tracing**

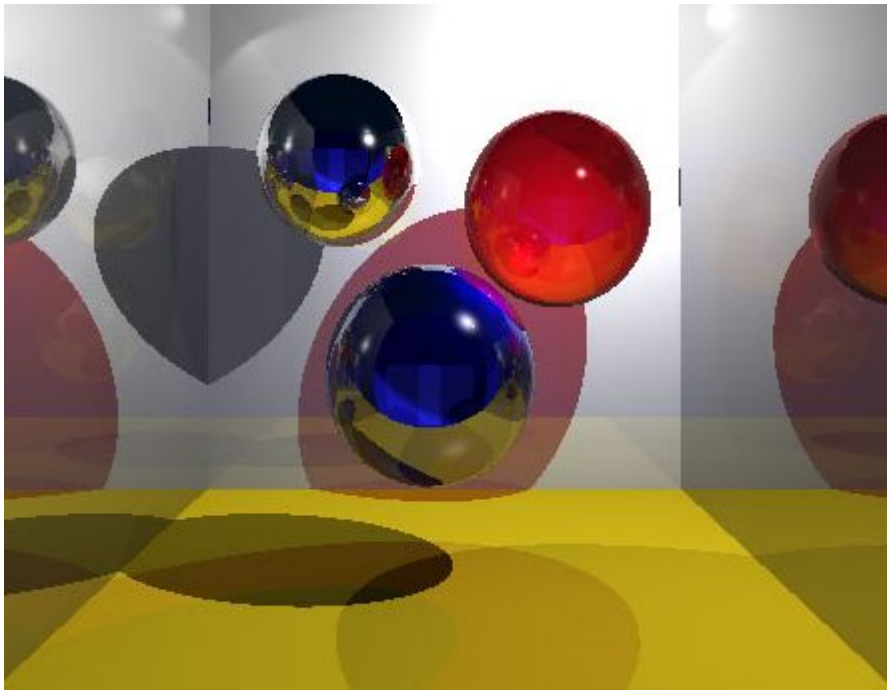
```
function TraceRay(R: ray; depth: integer): kolor;
...
begin
wyznacz P - najbliższy punkt przecięcia promienia R z obiektem;
Jeśli nie ma żadnego punktu przebicia:
TraceRay :=kolor_tła
else begin
 oblicz color jako światło otoczenia;
 znajdź wersor normalny do powierzchni obiektu w tym punkcie;
 oblicz lokalne oświetlenie punktu P przez wszystkie źródła światła i dodaj do color;

 if depth<maxDepth then begin   {depth określa głębokość rekurencji}
  ...
  ...
  {jeśli obiekt jest choćby częściowo lustrzany, wykonaj poniższe operacje:}
  wyznacz R_ray jako promień odbity w punkcie P;
  R_color:= TraceRay(R_ray, depth+1);
  color:=color + ksR_color;
  ...
  {jeśli obiekt jest choćby częściowo przeroczysty, wykonaj poniższe operacje:}
  wyznacz T_ray jako promień załamany w punkcie P;
  T_color:= TraceRay(T_ray, depth+1);
  color:=color + ktT_color;
  ...
 end;
 TraceRay :=color;
end;
```

Zatem oświetlenie obiektu w danym punkcie pochodzi nie tylko z lokalnego oświetlenia tego punktu przez źródła światła (które jak pamiętamy z poprzedniego rozdziału wyznacza się jako odbicie rozproszone i lustrzane), ale dodaje się do niego oświetlenie pochodzące z promieni wtórnych, które znowu jest sumą oświetlenia lokalnego i oświetlenia pochodzącego z kolejnych promieni wtórnych itd. W ten sposób uzyskuje się **efekty wzajemnych, wielokrotnych odbić** obiektów, czego nie da się uzyskać bez rekurencyjnego algorytmu.

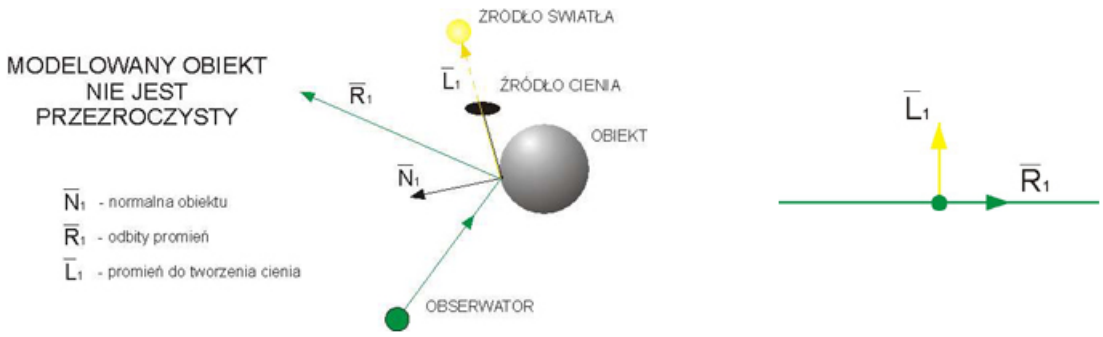
Przy obliczaniu oświetlenia lokalnego w danym punkcie prowadzi się promienie testujące i wyznacza ich wersory do każdego źródła światła. Jeśli promień taki natrafi na obiekt całkowicie nieprzezroczysty, dane źródło światła nie bierze udziału w obliczeniach oświetlenia lokalnego. Jeśli natrafi na obiekt częściowo przezroczysty, udział danego źródła światła w lokalnym oświetleniu powierzchni jest odpowiednio zmniejszany. W ten sposób

metoda śledzenia promieni prowadzi do wyznaczania **cieni na obiektach**, wynikających z zasłaniania ich przez inne obiekty.
Ostatecznie więc uzyskuje się więc tą metodą efektowne obrazy obiektów o różnym stopniu przezroczystości, wzajemnie i wielokrotnie się w sobie odbijających, wraz z cieniami, które na siebie nawzajem rzucają:

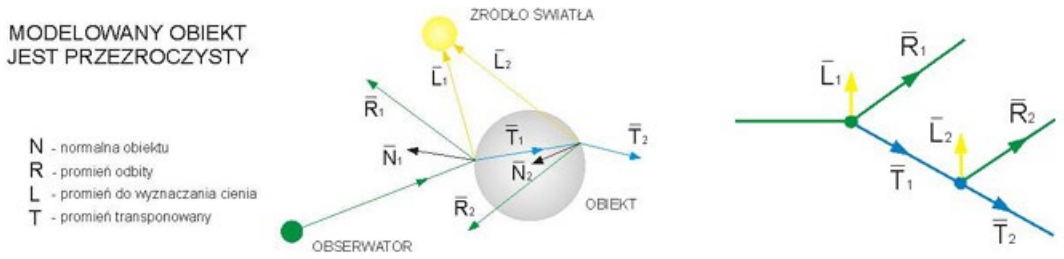


**Przykład 1 - seria ilustracji pokazująca zmianę obrazu wraz ze zwiększeniem poziomu rekurencji**

Metoda śledzenia promieni pozwala też na rendering obiektów CSG: przy przecięciach promienia śledzącego z obiektem CSG problem operacji logicznych redukuje się do jednego wymiaru. Ray tracing możliwia też nakładanie tekstur i odwzorowywanie otoczenia na powierzchni, a także dodawanie do sceny efektów specjalnych, takich jak tłumienie atmosferyczne, mgła i wiele innych. Pozwala wprowadzić różnego rodzaju wirtualne kamery (sferyczne, typu "rybie oko") i rozmaite źródła światła (reflektory, walcowe, powierzchniowe). Modyfikacje i uzupełnienia wprowadzane do algorytmu pozwalają nie tylko na generowanie ostrych cieni, ale i tych tzw. "miękkich", rozmytych, bardziej realistycznych.
Rekurencyjny proces zachodzący podczas śledzenia promieni można zilustrować tzw. drzewem promieni, czyli schematem pokazującym drogę promienia i jego późniejsze rozgałęzienia, aż do końca procesu rekurencyjnego. Poniżej schematy dla obiektów przezroczystych i nieprzezroczystych.




*Rysunek 59. Śledzenie promieni dla obiektów nieprzezroczystych; obok - drzewo promieni; na żółto zaznaczono promienie testujące, skierowane do źródła światła.*

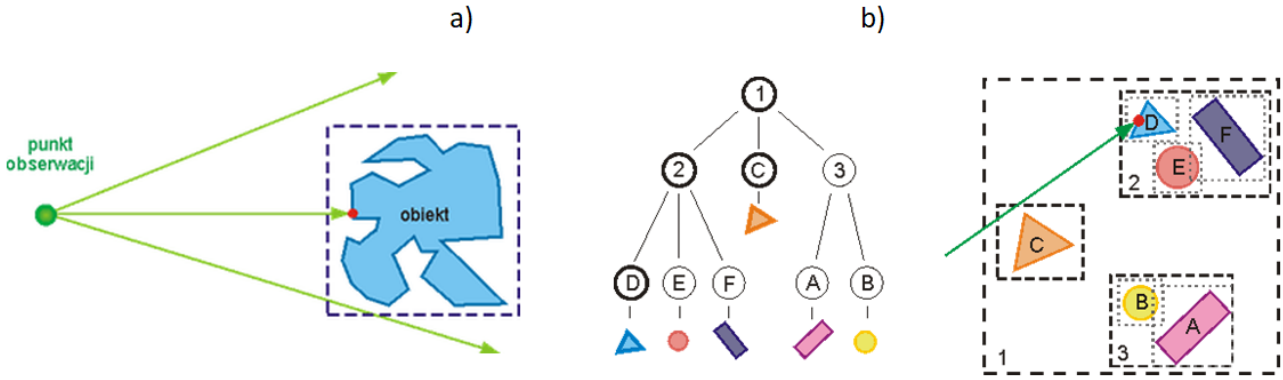



*Rysunek 60, Śledzenie promieni dla obiektów przezroczystych; obok - drzewo promieni; na żółto zaznaczono promienie testujące, skierowane do źródła światła.*

### 7.2.1 Metody przyspieszania obliczeń

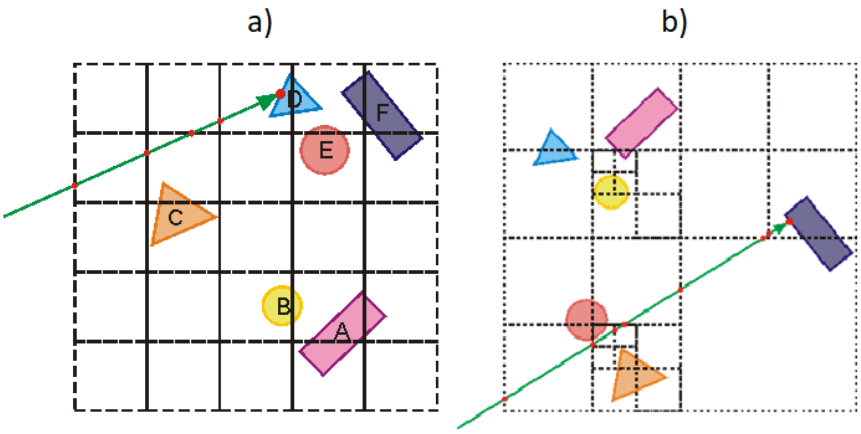
Z metodą śledzenia promieni w sposób naturalny związane są zagadnienia zwiększania szybkości obliczeń. Najbardziej klasyczne z nich (poza wspomnianą juz wcześniej pracą wieloprocesorową) są następujące:

- zwiększanie efektywności algorytmów przecięcia obiektów (tych o złożonej reprezentacji) z promieniem;
- metoda brył ograniczających (ang. *bounding volumes*) - skomplikowane obiekty otacza się prostymi otoczkami (np. sferami), których przecięcia z promieniem można łatwo obliczyć; jeżeli promień nie przetnie kształtu otaczającego obiekt, to możemy jednoznacznie stwierdzić, że sam obiekt również nie został przez ten promień przecięty. Dopiero kiedy obiekt otaczający zostanie trafiony przez promień, należy sprawdzić, czy nastąpiło przecięcie z obiektem właściwym. Prawdopodobieństwo, że obiekt zostanie trafiony przez promień, zależy od tego, jak ściśle jest on otoczony przez otoczkę ograniczającą. Przypadkiem szczególnym są powierzchnie NURBS, dla których znalezienie otoczek wypukłych wykonuje się w prosty sposób na podstawie siatki wierzchołków kontrolnych.
- Metodę tę można dodatkowo usprawnić wprowadzając hierarchię obiektów; obiekty leżące blisko siebie tworzą grupę obiektów, mającą wspólną otoczkę ograniczającą.




*Rysunek 61.a) zastosowanie otoczek ograniczających; b) zastosowanie hierarchii obiektów i ich otoczek.*

- metoda podziału przestrzeni sceny - tworzenie struktury złożonej z voxeli (czyli "trójwymiarowych pikseli", będących elementami przestrzeni 3D). Scena może być podzielona w sposób równomierny, ale bardziej efektywny jest adaptacyjny podział, który prowadzi się tak, by każdy voxel przecinał mniej niż pewną maksymalną liczbę obiektów sceny.




*Rysunek 62. Metoda podziału przestrzeni sceny: a) podział równomierny; b) podział przystosowany do sceny.*

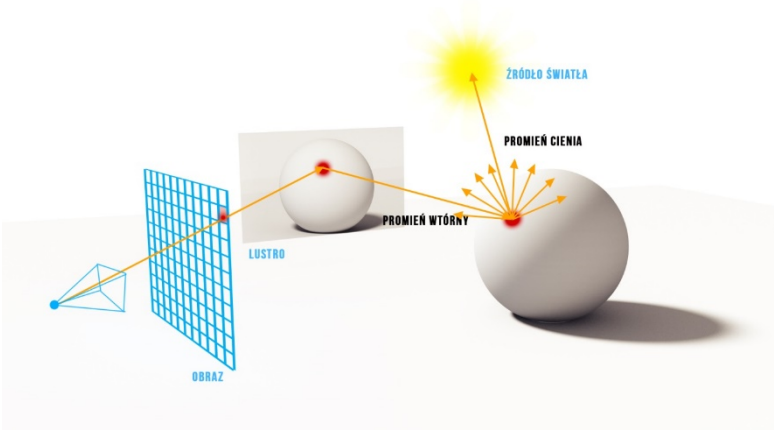
Metoda ray tracing należy do grupy metod tzw. fotorealistycznego renderingu. Przy dzisiejszych szybkościach komputerów pozwala na uzyskanie obrazów najwyższej jakości w czasie zbliżonym do rzeczywistego. Dlatego obecnie stosuje się ją nie tylko w systemach grafiki komputerowej, ale i w systemach CAD/CAM do realistycznego obrazowania zaprojektowanych modeli.

Zupełnie inną filozofię rozwiązania prezentuje **POV-Ray**, program - podobnie jak Blender - z grupy Freeware i Open Source, generujący obrazy w oparciu o metodę śledzenia promieni i scenę zapisaną w języku skryptowym. Ze względu na całkowicie odmienne podejście do modelowania sceny nie będziemy się nim tutaj zajmować, ale jest on ze wszech miar wart polecenia do samodzielnej pracy. Przykłady obrazów wygenerowanych przy jego użyciu były już zamieszczone w niniejszym rozdziale, a także w poprzedniej lekcji, zaś kolejne znajdują się w rozdziale o animacji.
Na zakończenie - animacja ilustrująca metodę śledzenia promieni; nie jest ona dokładnym odwzorowaniem działania raytracera - ilustruje działanie algorytmu przy najprostszych założeniach i jednym poziomie rekurencji.

Rys. 11.27. Animacja ilustrująca metodę śledzenia promieni. Osobno pokazano przypadki:
a) gdy promień testujący nie napotyka obiektu na swej drodze;
b) gdy napotyka obiekt matowy, o małym współczynniku odbicia lustrzanego - promienie wtórne mają małe znaczenie;
c) gdy napotyka obiekt lustrzany - udział promieni wtórnych jest bardzo istotny;
d) gdy napotyka obiekt przezroczysty - promienie wtórne wynikają z praw załamania światła i odbicia wewnętrznego.
Obsługa animacji:
- strzałka w lewo - odtwarzanie

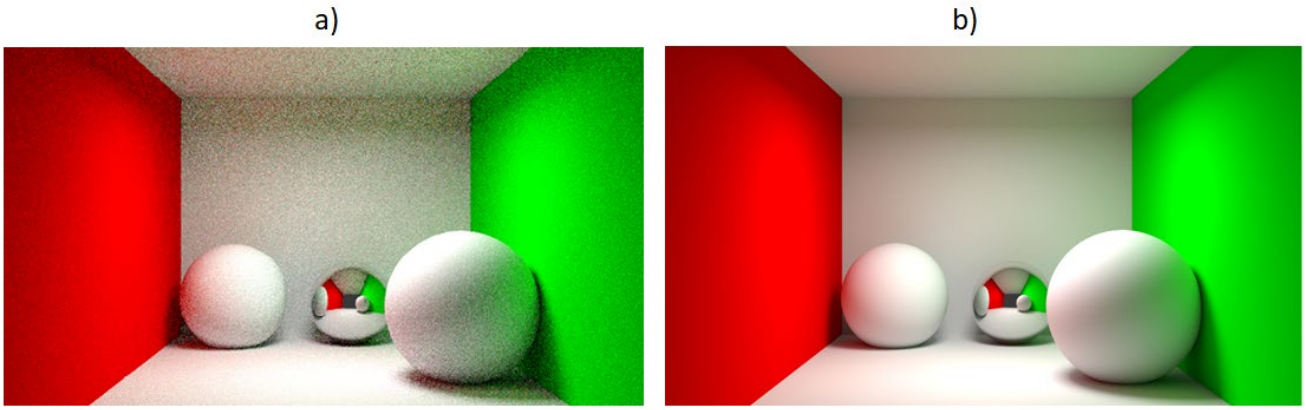
- podwójna kreska - pauza
- podwójna strzałka - przeskok do końca animacji

## 7.3 Path tracing, czyli śledzenie ścieżek




*Rysunek 63.*

Path tracing to rekursywne śledzenie metodą Monte Carlo ścieżek wychodzących od obserwatora i odbitych od obiektów w kierunkach losowych i wyznaczanie na tej podstawie wynikowego natężenia światła. Pozwala modelować globalne odbicia lustrzane (jak ray-tracing) oraz rozproszone (jak radiosity).



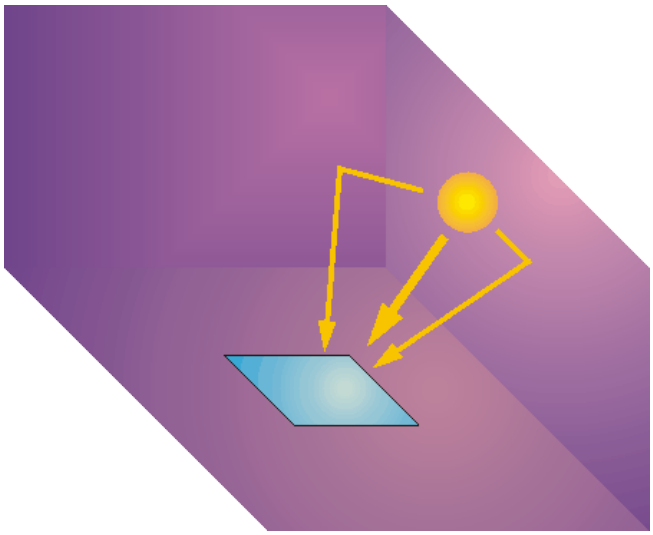

*Rysunek 64. Wynik renderowania sceny testowej: a} dla 5 próbek; b) dla 500 próbek.*

# 8 Radiosity, czyli metoda energetyczna

Metoda energetyczna, bardziej ściśle nazywana **metodą bilansu energetycznego** lub bilansu promieniowania, opisuje równowagę energii światła w zamkniętej przestrzeni, przy założeniu, że procesy emisji i odbicia są idealnie rozpraszające. Metoda pozwala wygenerować bardzo realistyczne obrazy powierzchni rozpraszających, zawierających subtelne cienie i półcienie i delikatnie oświetlone szczegóły, a więc uzyskać efekty niemożliwe do otrzymania metodą śledzenia promieni. Nie nadaje się natomiast do symulacji wzajemnych odbić między obiektami błyszczącymi. Dlatego dopiero połączenie obu tych metod prowadzi do uzyskania prawdziwie fotorealistycznych obrazów.

Metoda bilansu energetycznego zakłada zachowanie energii światła w zamkniętych środowiskach, zatem energia emitowana lub odbijana od każdej powierzchni jest w całości odbijana bądź pochłaniana przez inne

powierzchnie. Na oświetlenie składa się suma świateł bezpośredniego oraz odbitego. Światło bezpośrednie można zdefiniować jako pochodzące bezpośrednio ze źródła światła, zmodyfikowane wyłącznie przez czynniki takie, jak dym, mgła lub kurz. Za odbite uznaje się światło, które na swej drodze od źródła zostało odbite od jednej lub więcej powierzchni. Zamiast rozpatrywania oświetlenia otoczenia prowadzona jest więc dokładna symulacja odbić międzyobiektowych.



*Rysunek 65. Oświetlenie powierzchni światłem bezpośrednim i odbitym.*

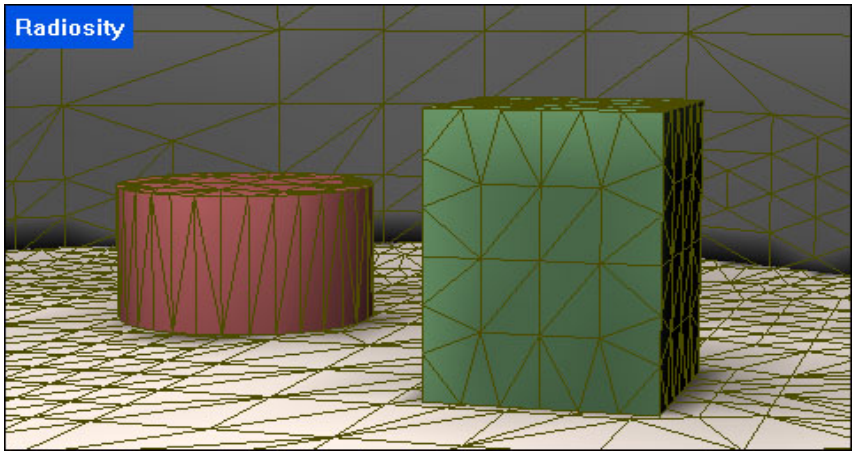
Energia opuszczająca powierzchnię jest zmieniana w zależności od właściwości odbijających powierzchni, które określają, jaka część światła zostaje odbita, a jaka pochłonięta (lub przetransmitowana) przez obiekt. Przeważnie jedyną modyfikacją światła odbijanego jest zmiana jego barwy. Energia wychodząca z danej powierzchni jest sumą energii emitowanej przez powierzchnię, odbijanej lub transmitowanej przez tę powierzchnię.


*Rysunek 66. Składniki energii opuszczającej powierzchnię.*

Pierwszym krokiem w algorytmie metody energetycznej jest ustalenie interakcji światła między obiektami sceny niezależnie od punktu obserwacji. Aby tego dokonać, musimy podzielić scenę wraz ze znajdującymi się w niej obiektami na elementarne płaty (najczęściej wieloboki), o których zakłada się, że mają stałą energię świetlną (promienistość). Otrzymana w ten sposób siatka płatów jest bardzo gęsta, znacznie gęstsza niż początkowa siatka obiektów definiujących powierzchnię (o ile jest ona zadana siatką 3D). Każdy z płatów wynikłych z takiego podziału w określony sposób odbija, przepuszcza (transmituje) lub emituje światło (jedna możliwość nie wyklucza innych). Na każdy z płatów mają wpływ wszystkie pozostałe płaty znajdujące się w danej scenie. W wyniku dokonanego bilansu energii otrzymuje się otrzymuje się **promienistości każdego płata**, a zatem intensywność każdego z nich. Wartości te są **niezależne od obserwatora**.

Następnym krokiem jest renderowanie obrazu dla wybranego przez nas rzutu, co wiąże się z dodatkowymi obliczeniami powierzchni widocznych z pozycji obserwatora oraz interpolacją intensywności. To zasadniczo różni tę metodę od metody śledzenia promieni, która pozwala wyłącznie na stworzenie obrazu zależnego od uprzednio obranego punktu obserwacji. Tutaj mając obliczoną emisję światła dla każdej powierzchni renderujemy od razu całą scenę określając tylko punkt obserwacji.

W metodzie energetycznej każda powierzchnia może emitować światło, w związku z czym wszystkie źródła światła są modelowane jako mające powierzchnię. Aby poprowadzić dalej rozważania, musimy najpierw podzielić stworzone przez nas środowisko - czyli obiekty i źródła światła - na skończoną liczbę *n* dyskretnych płatów o skończonej powierzchni, zakładając, że każdy z płatów jednolicie odbija i emituje światło na całej swojej powierzchni.




*Rysunek 67. Podział środowiska na n płatów.*

Po przyjęciu założenia, że każdy płat jest nieprzezroczystym, rozpraszającym emiterem i reflektorem, otrzymuje się następujące **równanie bilansu energetycznego**:

$$B_i = E_i + p_i \sum_{1 \le j \le n} B_j F_{i-j}$$

gdzie:

**$B_i$** - promienistość płata i,

**$B_j$** - promienistość płata j.

Promienistość jest mierzona w W/m$^2$ - czyli energia/jednostka czasu/jednostka powierzchni.

**$E_i$** - energia świetlna emitowana samoistnie przez płat i (ma taką sama jednostkę jak promienistość),

**$\rho_i$** - współczynnik rozproszenia światła (zdolność odbijania) dla płata i - współczynnik bezwymiarowy <1.

**$F_{j-i}$** - współczynnik sprzężenia (bezwymiarowy <1) - określa, jaka część energii opuszczająca płat j trafia do płata i (uwzględnia kształt i względną orientację płatów oraz obecność wszelkich zakłócających płatów).

Podane wyżej równanie bilansu energetycznego, opisujące oddziaływanie światła między płatami w zamkniętym środowisku, można zapisać w postaci następującego układu równań, w którym **niewiadomymi są promienistości $B_i$ płatów**:

$$\begin{bmatrix} 1-p_1F_{1-1} & -p_1F_{1-2} & \cdots & -p_1F_{1-n} \\ -p_2F_{2-1} & 1-p_2F_{2-2} & \cdots & -p_2F_{2-n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ -p_nF_{n-1} & -p_2F_{n-2} & \cdots & 1-p_nF_{n-n} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \cdot \\ \cdot \\ \cdot \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \cdot \\ \cdot \\ \cdot \\ E_n \end{bmatrix}$$

Macierz tego układu jest diagonalnie dominująca i najczęściej rozwiązuje się go korzystając z metody iteracyjnej Gaussa - Seidela.

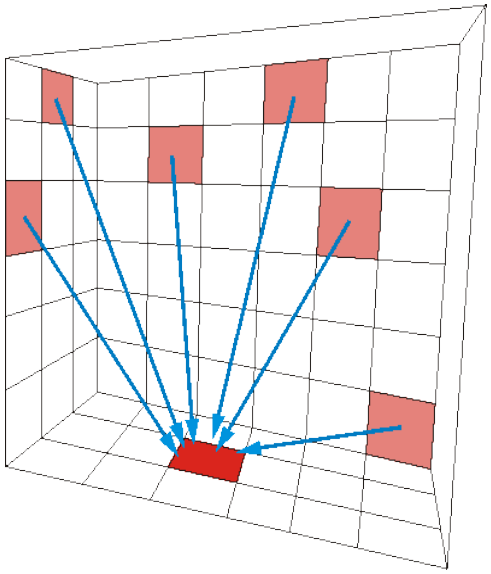
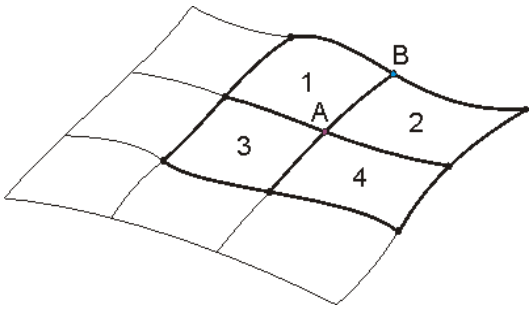


*Tabela 1. Graficzne przedstawienie obliczania promienistości.*

Po rozwiązaniu układu równań otrzymujemy promienistość (a stąd intensywność) dla każdego płata. Wartości te są stałe i niezależne od punktu patrzenia, co umożliwia - w kolejnym etapie - rendering płatów z dowolnego punktu obserwacji przy wykorzystaniu algorytmu wyznaczania powierzchni widocznych.

Podczas odtwarzania obrazu intensywność w pikselach wewnątrz każdego płata wyznacza się interpolując intensywności w wierzchołkach analogicznie jak w metodzie Gourauda. W tym celu należy najpierw wyznaczyć promienistość wierzchołków - oblicza się ją jako średnią promienistość sąsiednich płatów.
W przypadku gdy wierzchołek jest wewnętrzny dla powierzchni, to przypisuje mu się średnią wartość promienistości płatów, pomiędzy którymi leży. Promienistość wierzchołków leżących na krawędzi wyznacza się korzystając z promienistości płatów sąsiadujących z punktem bądź znajdujących się w jego bliskim otoczeniu, np. wg wzoru (Rysunek 68).

$$B = \max(0, (3B_1 + 3B_2 - B_3 - B_4))$$




*Rysunek 68. Wyznaczanie romienistości w wierzchołku*

Największą trudność w metodzie bilansu promieniowania stanowi obliczanie współczynników sprzężenia geometrycznego $F_{i-j}$ (zwanych też współczynnikami kształtu) między płatami. Współczynniki te zależą wyłącznie od geometrii powierzchni.

Aby uzyskać współczynnik sprzężenia płata $A_i$ do płata $A_j$, należy wykonać całkowanie po powierzchni obu płatów:

$$F_{i-j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} H_{ij} dA_j dA_i$$

gdzie:

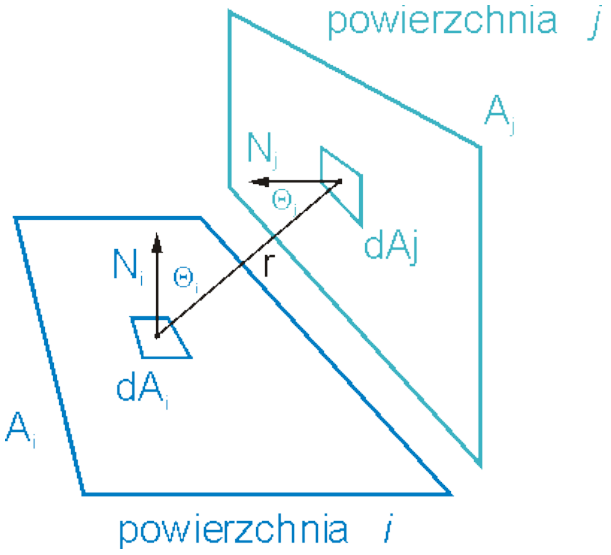
$dA_i$, $dA_j$ - powierzchnie elementarne,

r - długość promienia łączącego płaty,

$\Theta_i$ - kąt między promieniem a normalną do $A_i$,

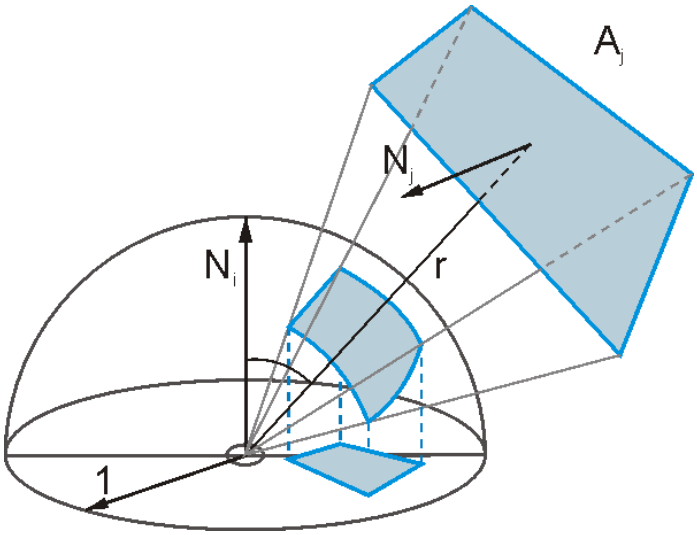
$\Theta_j$ - kąt między promieniem i normalną do $A_j$.

$H_{ij}$ - współczynnik przyjmujący wartość 1 lub 0 w zależności od tego, czy $dA_j$ jest widoczne z $dA_i$, czy też nie.

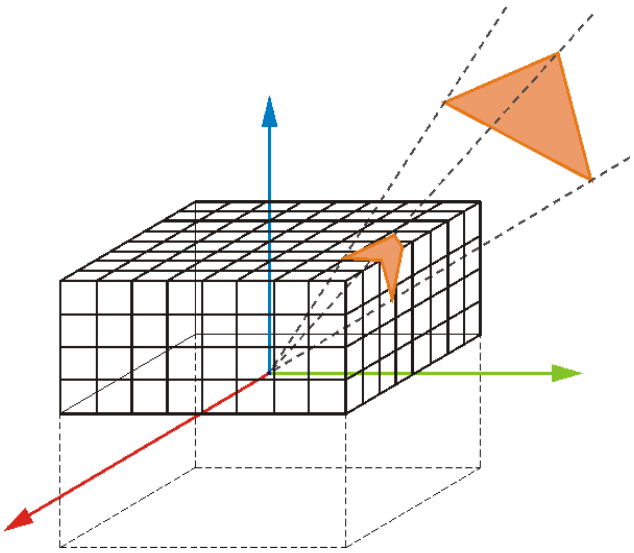



*Rysunek 69. Wielkości geometryczne potrzebne do obliczenia współczynnika sprzężenia między powierzchniami.*

Zgodnie z teorią Nusselta obliczanie współczynnika sprzężenia między powierzchnią $dA_i$, a powierzchnią $A_j$ jest równoważne rzutowaniu widocznych z $dA_i$ części płata $A_j$ na jednostkową półkulę ze środkiem w $dA_i$, a następnie dalsze rzutowanie tego rzutu na jednostkowe koło będące podstawą półkuli i dzielenie przez pole koła:




*Rysunek 71. Metoda hemisphere*



*Rysunek 70. Metoda "hemicube" - zasada obliczania współczynnika sprzężenia poprzez rzutowanie płata na półsześcian; elementarny płat dAi znajduje się w środku układu współrzędnych, płat Aj jest tu trójkątem.*

Zamiast tej analitycznej metody można zastosować metodę numeryczną, wykorzystującą rzutowanie płatów na odpowiednią ścianę półsześcianu (**"hemicube"**), stanowiącego przybliżenie półkuli będącej obszarem całkowania; współczynnik sprzężenia jest aproksymowany poprzez sumowanie wartości współczynników sprzężenia przypisanych do wszystkich elementarnych kwadratów na ścianach półsześcianu zakrytych przez płat $A_j$. Rzutowanie wszytkich płatów na powierzchnie półsześcianu odbywa się z wykorzystaniem z-bufora, tak by wyznaczyć wielobok najbliższy dla każdego piksela półsześcianu.

W celu zwiększenia efektywności zamiast jednokrotnego rozwiązania układu równań opisującego bilans energii i wyznaczania naraz wszystkich współczynników sprzężenia i promienistości wszystkich płatów, stosuje się metodę tzw. **progresywnych ulepszeń**, w której do tego końcowego rozwiązania dochodzi się, z pewnym przybliżeniem, w kolejnych krokach. Na początku wybiera się płaty pochodzące od źródeł światła i z każdego z nich kolejno wysyła się promieniowanie do wszystkich pozostałych płatów sceny, wyznaczając współczynniki sprzężenia ich z danym płatem, a na tej podstawie przyrost ich promienistości. Potem udział w oświetleniu całej sceny biorą kolejno te płaty, które odebrały od innych najwięcej energii, i promieniowanie od nich pochodzące jest akumulowane przez pozostałe płaty. W ten sposób następuje stopniowe rozjaśnianie sceny i zbliżanie się do końcowego bilansu promieniowania; proces przerywany jest, gdy zmiany promieniowania płatów po kolejnej iteracji są niezauważalne lub po przekroczeniu zadanej liczby iteracji (np. 100, jak w Rhino).

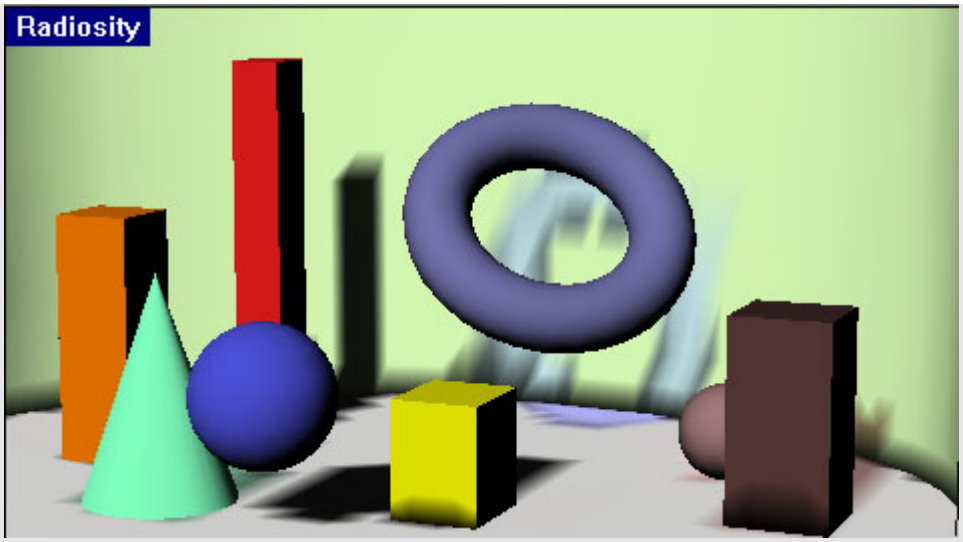
Jest oczywiste, że jakość obrazów generowanych metodą radiosity zależy nie tylko od liczby iteracji, ale też od sposobu podziału sceny na płaty (wieloboki). Podział taki zwykle dokonywany jest adaptacyjnie, w różny sposób dla płatów emitujących światło i dla absorberów, przy zadanych ustawieniach granicznych wielkości płatów.
Metodę energetyczną bardzo często **łączy się z metodą śledzenia promieni**. Obliczenia wykonywane są wówczas w dwu fazach:

- Faza niezależna od położenia obserwatora - wyznaczanie składowej rozproszonej (radiosity). Następuje tu podział powierzchni obiektów modelujących scenę na powierzchnie elementarne i wyznaczanie rozkładu oświetlenia rozproszonego;
- Faza zależna od położenia obserwatora - wyznaczanie składowej lustrzanej i przepuszczonej (ray tracing) i sumowanie jej z informacją o rozkładzie oświetlenia rozproszonego otrzymaną w fazie pierwszej.

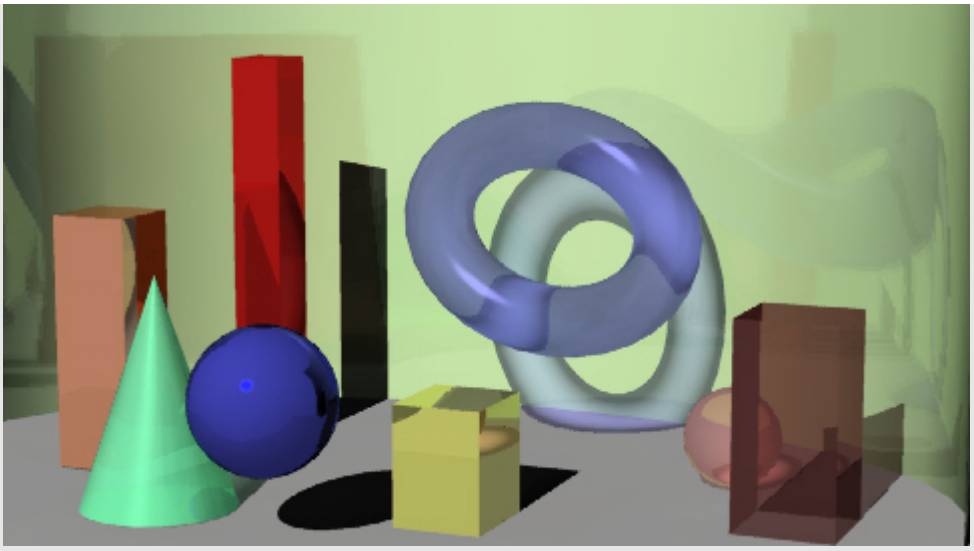
W ten sposób uzyskuje się obrazy o najwyższym stopniu realizmu - ale trzeba pamiętać, że jest to metoda bazująca na bilansie energii w zamkniętej przestrzeni, czyli dobrze nadaje się np. do wnętrz architektonicznych czy nawet do obrazowania zaprojektowanych modeli usytuowanych wewnątrz pomieszczeń. Niekiedy stosuje się ją też do obrazowania modeli "jako takich " czy scen w otwartej przestrzeni, ale wtedy należy tę przestrzeń ograniczyć dodatkowo zamodelowaną kopułą nieboskłonu, będącą emiterem światła. Tego typu źródło światła dobrze symuluje światło dzienne, słoneczne, zastępuje więc wprowadzanie punktowego źródła światła i pozwala uzyskać całkowicie odmienne, często bardziej realistyczne efekty.

Metoda radiosity, kiedyś czysto teoretyczny pomysł naukowców bez szans na implementację ze względu na złożoność obliczeń, wraz ze wzrostem mocy obliczeniowej komputerów staje się coraz bardziej powszechna i w ostatnich latach stała się dostępna nie tylko w najdroższych systemach grafiki komputerowej, ale również w systemach CAD/CAM i takich programach jak Rhinoceros, POV-Ray i Blender.

Poniżej przedstawiono parę obrazków wygenerowanych w Rhino. Pierwszy z nich pokazuje cienie i odbicia obliczone metodą energetyczną, drugi - po przejściu przez etap działania ray tracera. Warto zwrócić uwagę na różnice pomiędzy wyglądem obiektów przezroczystych i odbijających po i przed przeliczeniem obrazka przy pomocy ray tracingu.




**Rys. 11.36. Cienie i odbicia policzone przy pomocy metody energetycznej - tylko po barwie cieni można się domyślić, które obiekty są przezroczyste (bo cienie ich nie są czarne).**



Rys. 11.37. Ten sam rysunek dodatkowo policzony metodą Raytracingu - dokładnie widać, który obiekt jest przezroczysty, a także który ma powierzchnię metaliczną.

# 9 Wstęp do animacji

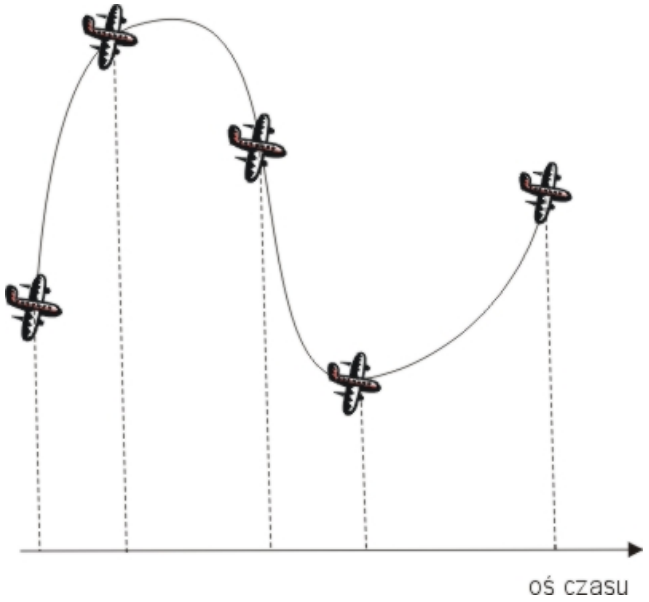
Animacja komputerowa, ktorą znamy na co dzień z reklam telewizyjnych, filmów animowanych i gier komputerowych znajduje szerokie zastosowanie również w nauce, edukacji, medycynie i przemyśle. Profesjonalna animacja najwyższej klasy (np. produkcja filmów animowanych) wymaga stosowania najdroższych high-endowych systemów wykorzystujących najnowsze rozwiązania algorytmiczne i techniki programistyczne, ale w wielu dziedzinach wystarczające może okazać się zastosowanie aplikacji mniej kosztownych, a nawet bezpłatnych, jak np. Blender. Niniejszy rozdział jest przeglądem najczęściej stosowanych rozwiązań, ze szczególnym uwzględnieniem możliwości najbardziej popularnych, dostępnych we wszystkich systemach.
Animacja jest to zmiana pewnej właściwości w czasie:

- położenia obiektu w przestrzeni 3D, wliczając w to również obroty
- kształtu obiektu, w tym np. skali obiektu
- jego barwy i przezroczystości
- jego tekstury
- położenia kamery i jej parametrów (np. ogniskowej)
- parametrów ruchu (np. prędkości i przyśpieszenia) obiektu
- parametrów ruchu kamery
- i wielu innych, mogących podlegać zmianom, atrybutów obiektu bądź śledzącej go kamery.

Poniżej omówione zostaną w bardzo dużym skrócie techniki, które służą do wizualizacji tych zmian we współczesnych systemach do animacji 3D.

Animacja kluczowa (ang. *key framing*) jest bezpośrednim przeniesieniem do animacji komputerowej zasad stosowanych w animacji tradycyjnej. Osoba zajmująca się animacją - animator musi mieć kontrolę nad całością zmian. Tylko w wyjątkowych sytuacjach animator musi tworzyć kolejne klatki (ramki) animacji krok po kroku, ręcznie wprowadzając wszelkie zmiany. Na ogół główny animator (*keyframer*) tworzy tylko tzw. **klatki kluczowe**, charakterystyczne dla sceny, zaś wszystkie klatki pomiędzy kluczowymi wykonują zwykli animatorzy. Dzięki komputerom i programom do animacji 3D, żmudny proces tworzenia klatek międzykluczowych nie musi być wykonywany przez człowieka, gdyż jest wynikiem interpolacji wykonywanej przez komputer.

Każda klatka kluczowa stworzona metodą komputerowego modelowania sceny przez animatora wnosi ze sobą informację o wartościach różnych atrybutów (zmiennych) w danej chwili czasowej, przypisanej do klatki (klatki projektuje się względem tzw. **osi czasu**, ang. *timeline*). Jeżeli pomiędzy kolejnymi klatkami kluczowymi pewien atrybut ulega zmianie, program musi dokonać interpolacji jego wartości w zadanym przedziale czasowym.
**Interpolacja** polega więc na wyznaczeniu krzywej przechodzącej przez zadane punkty określone przez wartości atrybutów w wybranych chwilach czasowych, gdyż na tej podstawie można wyznaczyć dowolną ilość klatek pośrednich. Zwykle chcemy, by zmiany następowały w sposób płynny, czyli krzywa taka powinna być gładka. Najczęściej do interpolacji stosuje się więc krzywe gładkie złożone z krzywych wielomianowych 3 stopnia (znane nam z lekcji 8), np. krzywe sklejane trzeciego stopnia:




*Rysunek 72. Przykład interpolacji międzykluczowej funkcją sklejaną. Przejście pomiędzy każdymi dwiema klatkami obiektu opisane jest funkcją wielomianową trzeciego stopnia. Na osi rzędnych mogą być wartości dowolnego atrybutu obiektu, nie tylko np. jego wysokość z, ale też np. jego kąt obrotu względem którejś z osi, jego skala lub kolor.*

Należy podkreślić, że krzywa, którą system generuje w wyniku interpolacji, nie może być dowolna - musi być jednoznaczną funkcją czasu: każdej chwili czasowej może odpowiadać tylko jedna wartość atrybutu.

Jeśli chcemy uzyskać lepszą kontrolę nad sposobem zmian atrybutów w czasie i nie dopuścić do powstania klatek pośrednich wyglądających mało realistycznie (co może być niekiedy efektem interpolacji), można "ręcznie" wpłynąć na przebieg krzywej interpolującej. Najczęściej użytkownik używając edytora wykresu może modyfikować poszczególne segmenty funkcji interpolującej, reprezentowane jako krzywe Béziera 3 stopnia, za pomocą punktów kontrolnych. Zwykle może też sobie zażyczyć, by interpolacja była prostsza, liniowa (funkcja interpolująca jest wówczas łamaną łączącą węzły interpolacji) albo dla uzyskania specjalnych efektów celowo wprowadzić ostrza na gładkiej funkcji interpolującej.

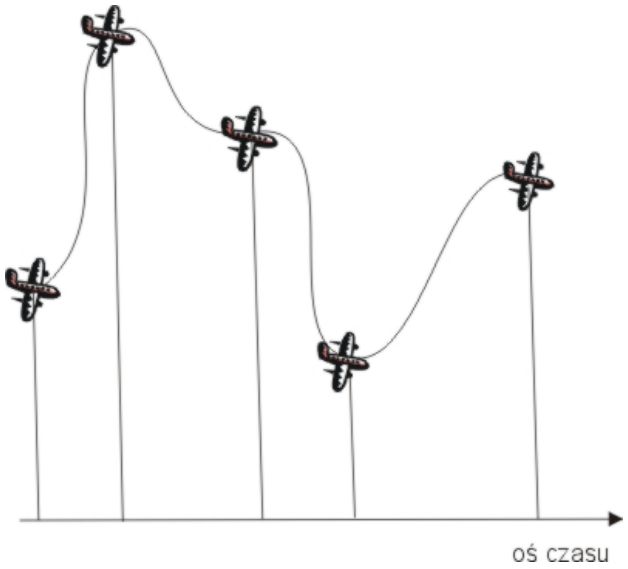
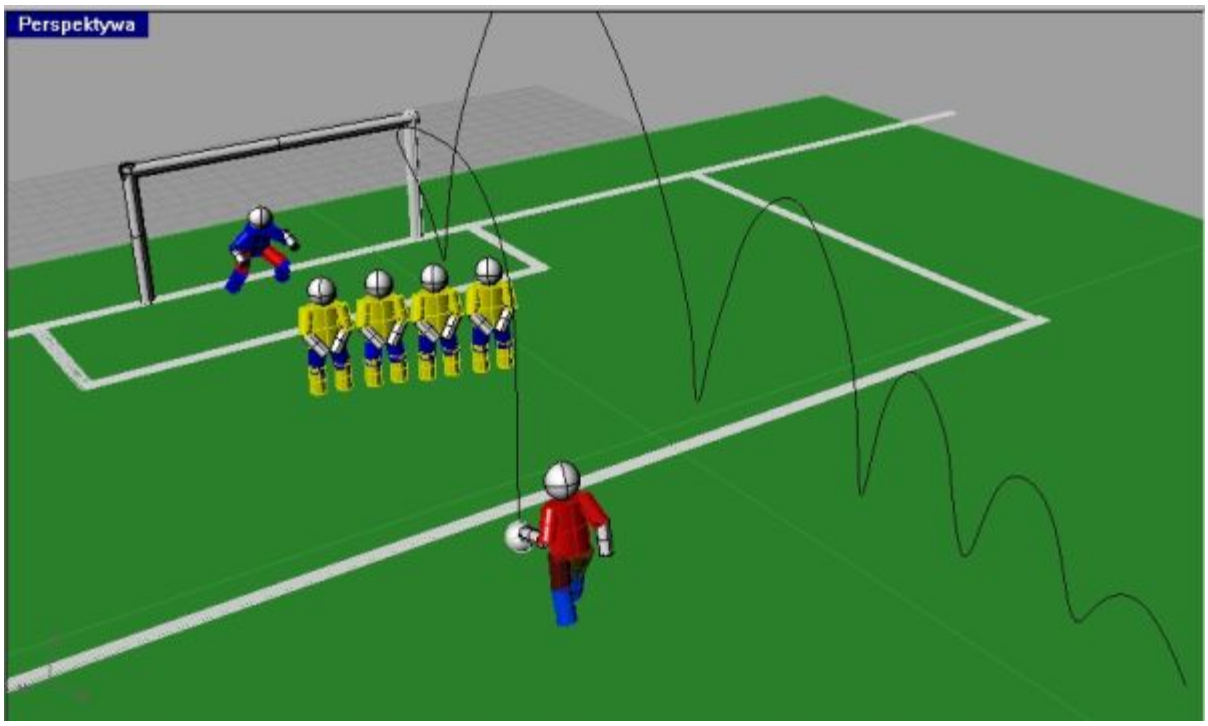



*Tabela 2. Przykład funkcji interpolującej po dokonaniu edycji - teraz zmiany atrybutu w pobliżu klatek kluczowych następują wolniej, natomiast szybsze są przejścia pomiędzy klatkami.*

Za pomocą edycji funkcji interpolujących (w Blenderze noszących nazwę krzywych IPO) można animować materiały, tekstury, można odzwierciedlić np. migotanie światła w scenie i wpływać na przyspieszanie i zwalnianie tempa animacji. Każda z funkcji interpolujących może być edytowana niezależnie od innych. Taki sposób wpływania na przebieg animacji nazywa się najczęściej **animacją wykresową**.

W najprostszych systemach interpolacja pomiędzy klatkami kluczowymi wykonywana jest automatycznie i użytkownik nie może ingerować w przebieg funkcji interpolującej.

Zupełnie innym sposobem wyznaczania kolejnych klatek animacji jest **animacja ścieżkowa**. Pozwala ona na animację tylko tych atrybutów, które związane są z przestrzenią 3D, czyli na animację położenia obiektu i/lub kamery. Tę najbardziej popularną opcję posiadają wszystkie systemy do animacji, niezależnie od ich wielkości. Animacja ścieżkowa polega na zdefiniowaniu krzywej będącej torem (ścieżką) poruszającego się obiektu bądź kamery. Krzywa taka może być dowolnie usytuowana w przestrzeni 3D. Ścieżki modeluje się w systemie tymi samymi technikami, co wszelkie inne krzywe, a więc jako krzywe przechodzące przez zadane punkty w przestrzeni 3D lub jako krzywe zadane wierzchołkami kontrolnymi.




*Rysunek 73. Przykład definicji ścieżki dla prostej animacji ścieżkowej - tor piłki zadany krzywą przestrzenną.*

Na ogół dodatkowo można ingerować w ścieżkę ruchu, na przykład nakazać obiektowi (lub kamerze) poruszanie się wzdłuż ścieżki, tak żeby wybrany wektor związany z obiektem był skierowany równolegle do stycznej do ścieżki ruchu. Będzie to przypominało efekt "patrzenia się" przedmiotu wzdłuż ścieżki ruchu. Można też usytuować obiekt względem ścieżki w inny sposób, np. zmusić, by w trakcie ruchu po ścieżce zawsze był zwrócony w stronę innego zdefiniowanego w scenie obiektu (ruchomego lub nieruchomego). Zwykle można też dodatkowo kontrolować prędkość i przyspieszenie obiektu poruszającego sie po trajektorii. W analogiczny sposób można animować ruch kamery względem ścieżki, uzyskując dodatkowe efekty filmowania sceny ruchomą kamerą.

Aby zdefiniować usytuowanie obiektu względem ścieżki, animator zwykle posługuje się lokalnym układem współrzędnych (zwykle określanym w systemach jako tzw. ***pivot***), związanym z obiektem. Względem osi tego układu definiuje się też obroty i skalę obiektu w danej chwili czasowej.

Animację ścieżkową zwykle łączy sie z animacją kluczową: w wybranych miejscach ścieżki można ustawić klatki kluczowe, które pozwalają na modyfikację wybranych atrybutów obiektu (skala, obrót, przezroczystość itp.) w trakcie jego ruchu po ścieżce.

Obiekty poddawane animacji ścieżkowej i/lub kluczowej mogą być tworzyć **strukturę hierarchiczną**: definiując jakiś obiekt jako podrzędny względem innego powodujemy, ze wszelkie transformacje dokonywane na obiekcie nadrzędnym będą dotyczyły również obiektu podrzędnego, który oprócz tego może mieć zdefiniowane swoje własne, lokalne transformacje. Przykładowo samochód może poruszać się po wybranym torze, a jego koła, jako obiekty podrzędne, oprócz ruchu po torze mogą wykonywać ruch obrotowy wokół swoich osi. Struktura hierarchiczna w przypadu złożonych scen i obiektów może być bardzo rozbudowana.

**Animacja hierarchiczna** często wiąże się z definiowaniem tzw. **obiektów proxy** (dostępnych np. w Rhino). Jeżeli chcemy animować całą grupę równorzędnych obiektów, to zamiast żmudnie każdy z nich np. przypisywać do ścieżki, możemy zdefiniować dla nich jeden wspólny punkt, leżący w ich pobliżu (tzw. punkt proxy) i do niego, jako podrzędne, przypisać wszystkie obiekty tej grupy. Wszelkie akcje odnoszące się do obiektu proxy (punkt proxy ma przypisany mu lokalny układ współrzędnych, więc pełni rolę obiektu) są wykonywane na wszystkich podłączonych do niego obiektach podrzędnych.

## 9.1 Animacja oparta na prawach fizyki

Animacja oparta na prawach fizyki należy do najbardziej zaawansowanych funkcji spotykanych w programach do animacji 3D i można by jej poświęcić niejeden wykład i niejeden podręcznik. Tutaj więc omówiona zostanie wyłącznie hasłowo.

Najpopularniejsze systemy oparte na prawach fizyki to **systemy cząsteczkowe** (ang. *particles* ). Znajdziemy je w postaci gotowych funkcji lub pluginów nie tylko w największych systemach graficznych, ale także w Blenderze. Pozwalają symulować ruch naturalnych, nieregularnych cząsteczek unoszących się w powietrzu lub w cieczy i zmieniających się w czasie. Można z ich pomocą modelować takie obiekty, jak ogień, dym czy wodospad, a także statyczne obiekty, ale o płynnej formie, np. trawę czy włosy. W trakcie animacji można dodawać obiekty, które będą wchodzić w interakcje ze strumieniem cząsteczek (np. zderzenie z obiektem spowoduje rozproszenie strumienia).

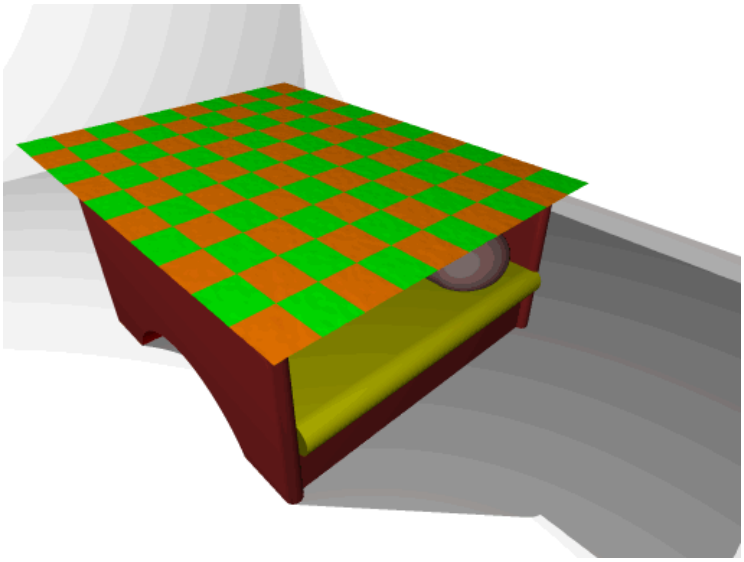
Emiter (generator cząsteczek) jest obiektem, który wyrzuca z siebie w sposób losowy (o predefiniowanych parametrach rozkładu prawdopodobieństwa) nieraz ogromne ilości cząsteczek. Cząsteczką może być obiekt o dowolnej (często bardzo prostej) geometrii, który można zamodelować w systemie. Cząsteczki generowane przez emiter mają nadawane pierwotne wartości atrybutów, takich, jak:

- położenie
- prędkość i przyspieszenie
- kształt
- rozmiar
- barwa
- przezroczystość
- itp.

i atrybuty te (stałe lub ustawiane losowo na początku) ulegają zmianom w czasie, wynikającym z praw kinematyki i dynamiki, np. z działania sił wiatru i grawitacji. Cząsteczki w momencie ich wygenerowania mają też przydzielony czas życia, po którym ulegają zanikowi. Mogą też zginąć przedwcześnie, np. spadając na ziemię. Renderowanie tysięcy cząsteczek odbywa się z dużym uproszczeniem; zwykle nie rozpatruje się tu zagadnień widoczności, cząsteczki do renderowania zastępuje się prostszymi obiektami, a barwa piksela jest sumą barw wszystkich cząsteczek odwzorowujących się na ten piksel.

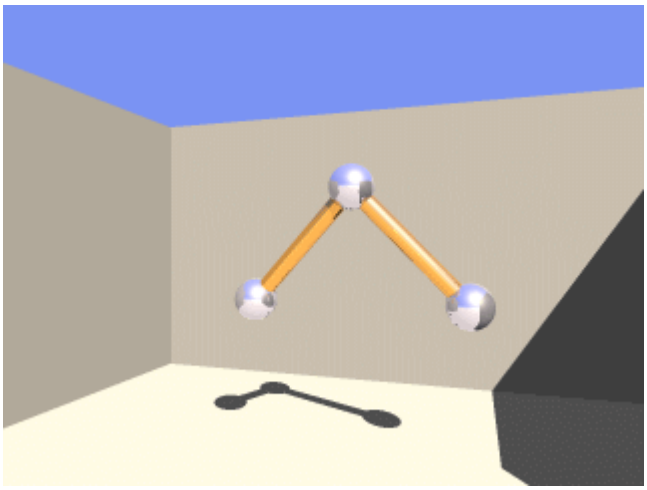
**Animacja tkanin** jest jedną z najbardziej zaawansowanych funkcji w systemach animacji 3D. Pierwotnie wrażenie poruszającej się tkaniny (np. łopoczącej na wietrze flagi) uzyskiwano przy użyciu deformatorów, które wprowadzały zaburzenia. Nie była to jednak dokładna symulacja - nie brano pod uwagę prędkości wiatru, siły grawitacji, własności tkaniny. Przy wykorzystaniu funkcji *Cloth* można już wprowadzić tego rodzaju parametry i uzyskać realistyczny efekt. Animację tkanin od strony programowej realizuje się różnymi metodami; jedna z możliwości polega na wykorzystaniu systemów cząsteczkowych, innym rozwiązaniem jest modelowanie tkanin jako systemu węzłów połączonych sprężynami.

Jeszcze do niedawna funkcja *Cloth* znajdowała się tylko w najdroższych pakietach, takich jak Maya, czy Softimage. Obecnie pojawia się coraz więcej plug-inów umożliwiających realistyczną animację tkanin w różnego rodzaju programach:



*Rysunek 74. Animowany GIF. Symulacja ruchu tkaniny z użyciem funkcji SimCloth z nakładki MegaPOV do programu POV-Ray*

Innym rodzajem zaawansowanej animacji opartej na prawach fizyki są **symulacje mechaniczne** - pozwalające na wizualizację zderzeń obiektów, drgań sprężyn, ruchu uwzględniającego siłę oporu w wodzie czy powietrzu. Im lepiej są w tych programach uwzględnione prawa kinematyki i dynamiki, i im lepiej użytkownik radzi sobie z ustawianiem parametrów, tym bardziej realistyczny efekt.



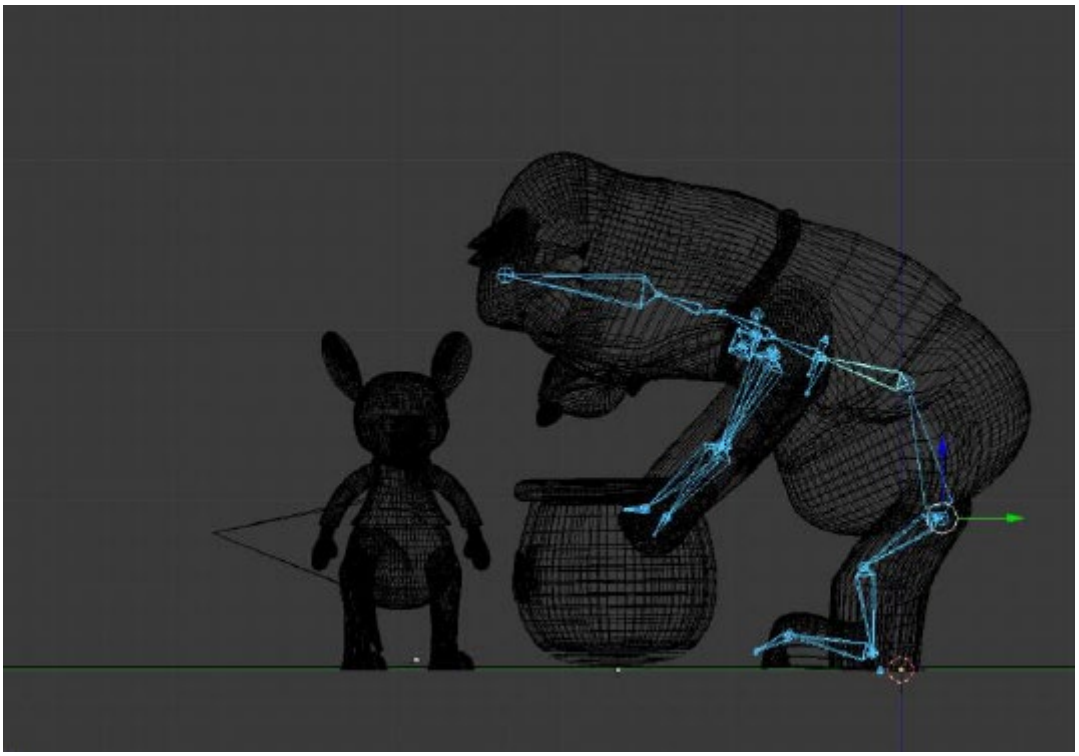
*Rysunek 75. Animowany GIF. Symulacja zderzeń z użyciem nakładki MegaPOV do programu POV-Ray*

## 9.2 Animacja z użyciem szkieletów

Najbardziej zaawansowanym zagadnieniem w animacji jest **animacja postaci**- niezwykle obszerna dziedzina wymagająca wielu wyspecjalizowanych funkcji. Jednym z nich jest modelowanie szkieletów, połączone z nakładaniem mięśni i skóry. Dzięki zaawansowanym narzędziom animując zginanie nogi nie musimy martwić sie o ułożenie skóry, czy ubrania - program uwzględni wszystkie zależności.

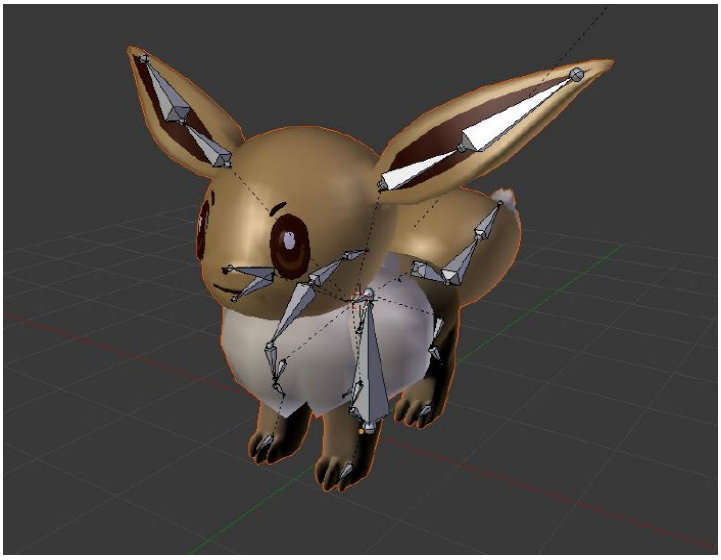
Szkielet jest **hierarchicznym systemem kości** (obiektów reprezentowanych w największym uproszczeniu odcinkami) połączonych ze sobą stawami (przegubami)) o określonej liczbie stopni swobody (np. możliwych przesunięć i obrotów). Szkielet po zdefiniowaniu systemu kości i ich wzajemnych zależności poddaje się oblekaniu skórą (tzw. *skining*), co polega na przypisaniu szkieletowi wierzchołków definiujących obiekty geometryczne związane z tymi szkieletami. W efekcie tego poruszenie kości wywołuje przemieszczenie tychże wierzchołków, dzięki czemu skóra ulega deformacji odpowiednio do ruchu szkieletu.

Animacja postaci wymaga zastosowania nie tylko systemu kinematyki prostej (FK - *Forward Kinematics*), w której ruch kości nadrzędnej jest przenoszony na ruch kości podrzędnych, ale przede wszystkim znacznie bardziej złożonego w obliczeniach systemu **kinematyki odwrotnej** (IK - *Inverse Kinematics*). Oznacza to, że w hierarchicznej strukturze kości ruchy kości podrzędnych są przeliczane na ruch kości stojących wyżej w hierarchii - np. ruchy stopy powodują odpowiednie ruchy kolan i kości udowych. Obliczenia muszą uwzględniać wszelkie ograniczenia nałożone na zakres ruchów poszczególnych elementów szkieletu, płaszczyzny ich ruchów itp.
Definiowanie szkieletów stosuje się w grafice komputerowej nie tylko w stosunku do postaci ludzi i zwierząt, ale także do animacji wszelkich przedmiotów, którym chce się nadać ruchy naśladujące ruchy istot żywych. Zasady kinematyki odwrotnej stosowane są też przy symulacji ruchów różnego rodzaju mechanizmów, np. robotów.

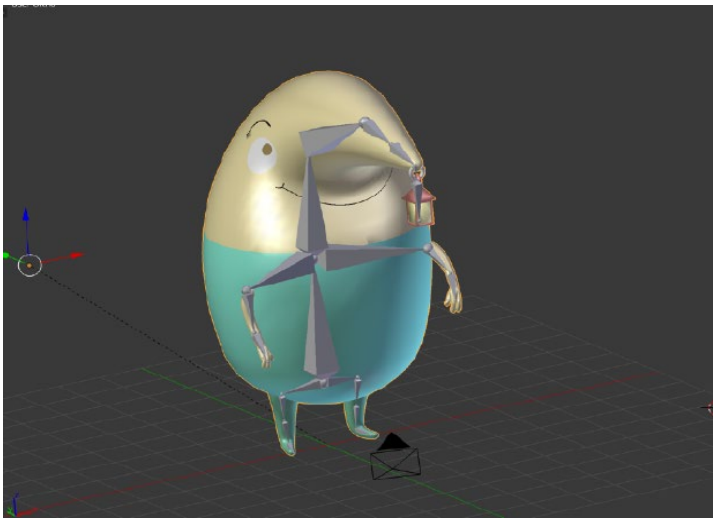


*Rysunek 76. Kubuś Puchatek z systemem kości - praca Pawła Pioruna*

Prezentowane tutaj przykłady animacji postaci z systemem kości zostały wykonane w Blenderze przez studentów Wydziału Mechatroniki; raporty wraz z filmami umieszczone są w osobnym katalogu dołączonym do Modułu 2.



*Rysunek 78. Pokemon Eevee z systemem kości - praca Aleksandry Zając*



*Rysunek 77. Postać z Rhino z systemem kości - praca Zuzanny Urbańskiej*