Programowanie Grafiki Komputerowej

Microsoft DirectX cz. 3 - oświetlenie i teksturowanie

Grzegorz Łukawski

Katedra Informatyki, Politechnika Świętokrzyska w Kielcach



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



Politechnika Świętokrzyska

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

" Program Rozwojowy Potencjału Dydaktycznego Politechniki Świętokrzyskiej w Kielcach: *kształcenie w nowoczesnych obszarach techniki* "

Program Operacyjny Kapitał Ludzki, Priorytet IV Szkolnictwo wyższe i nauka, Działanie 4.1. Wzmocnienie i rozwój potencjału dydaktycznego uczelni oraz zwiększenie liczby absolwentów kierunków o kluczowym znaczeniu dla gospodarki opartej na wiedzy, Poddziałanie 4.1.1. Wzmocnienie potencjału dydaktycznego uczelni.

Umowa UDA–POKL.04.01.01-00395/09-00

Spis treści

1. Oświetlenie
   - Wektor normalny
   - Modele oświetlenia
2. Oświetlenie w DirectX
   - Wektory normalne
   - Materiały
   - Parametry oświetlenia
   - Przykładowy program (dx-p5)
3. Teksturowanie
   - Parametry tekstury
   - Mipmapping
   - Teksturowanie w Direct3D
   - Obsługa tekstur w Direct3D
   - Przykładowy program (dx-p6)
4. Koniec

Uwagi

- Animacje są uruchamiane, gdy plik PDF jest wyświetlany na pełnym ekranie.
- Animacja jest uruchamiana dopiero po naciśnięciu spacji (program Adobe Reader).
- Klatki animacji wyświetlane są co 2 sekundy.
- Symulacje są uruchamiane po kliknięciu w ikonę z odpowiednim systemem operacyjnym. Przeglądarka wyświetli okno z ostrzeżeniem, należy go zaakceptować. W systemie Linux należy dodatkowo kliknąć dwukrotnie w nazwę programu.

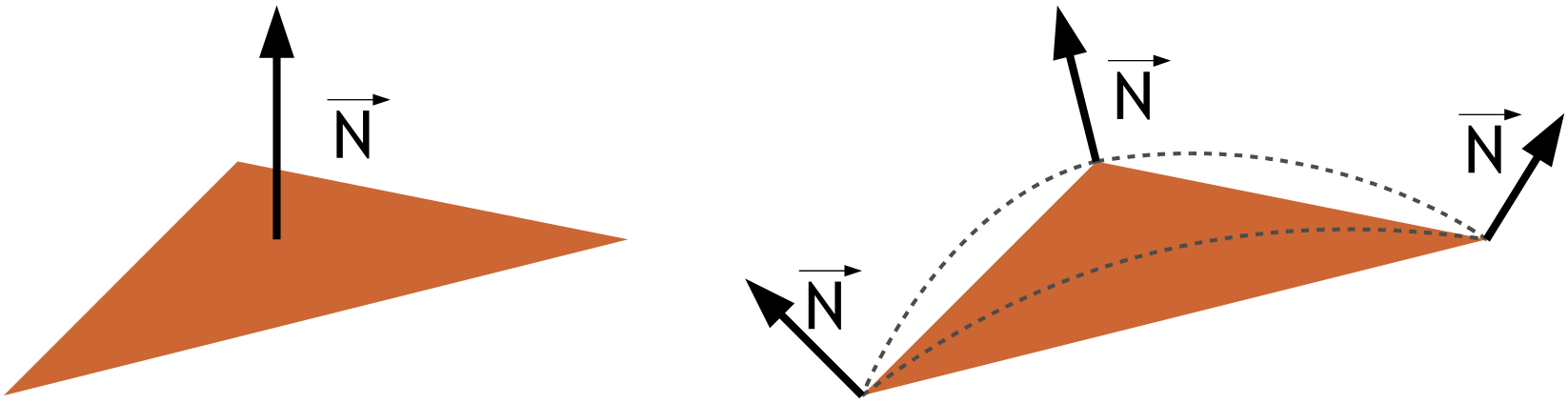
Oświetlenie

Oświetlenie to jeden z podstawowych elementów realistycznej sceny 3D. Sceny generowane bez oświetlenia wydają się płaskie i pozbawione szczegółów.

Modele oświetlenia symulują oświetlenie o różnej charakterystyce pochodzące z różnych źródeł.

Materiał określa sposób zachowania się powierzchni obiektu oświetlonej światłem. Oprócz koloru obejmuje również inne właściwości, jak chropowatość i współczynniki odbicia światła różnego rodzaju.

Do obliczenia oświetlenia niezbędny jest **wektor normalny powierzchni (normalna)**. Normalna jest wektorem prostopadłym do powierzchni, określa jej orientację względem źródła światła.



Jeżeli wielokąt jest aproksymacją powierzchni krzywoliniowej (np. fragmentu sfery), to dla każdego wierzchołka można zdefiniować **inną normalną** (rysunek z prawej strony).

W komputerowej grafice 3D stosuje się trzy podstawowe modele oświetlenia:

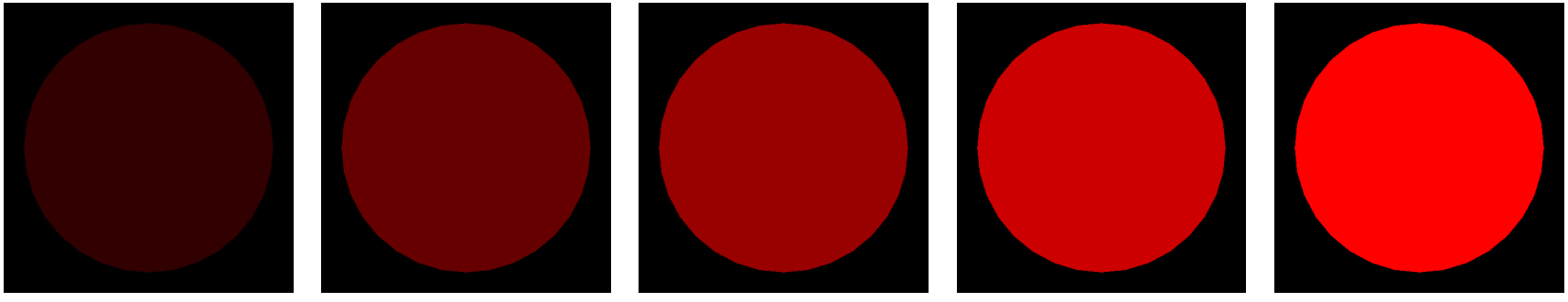
1. **Światło otoczenia (ang. ambient)** – model symulujący oświetlenie globalne. Światło nie posiada określonego źródła, oświetla obiekty równomiernie ze wszystkich stron. Stosowane jest do rozjaśniania zacienionych części obiektów.
2. **Światło rozproszenia (ang. diffuse)** – podstawowy model symulujący kierunkowe i punktowe źródła światła. Na powierzchni obiektów symulowane jest tzw. odbicie rozproszone (lambertowskie).
3. **Odbicie zwierciadlane (ang. specular)** – model symulujący światło odbijające się od gładkich powierzchni, z zastosowaniem prawa odbicia. Realizowane zazwyczaj z pomocą modelu Phonga lub Blinna-Phonga.

Światło otoczenia

Natężenie **światła otoczenia (ang. ambient)** zależy tylko od natężenia źródła światła I_a i współczynnika odbicia światła tego typu przez obiekt k_a (właściwość materiału).

$$I = I_a * k_a$$

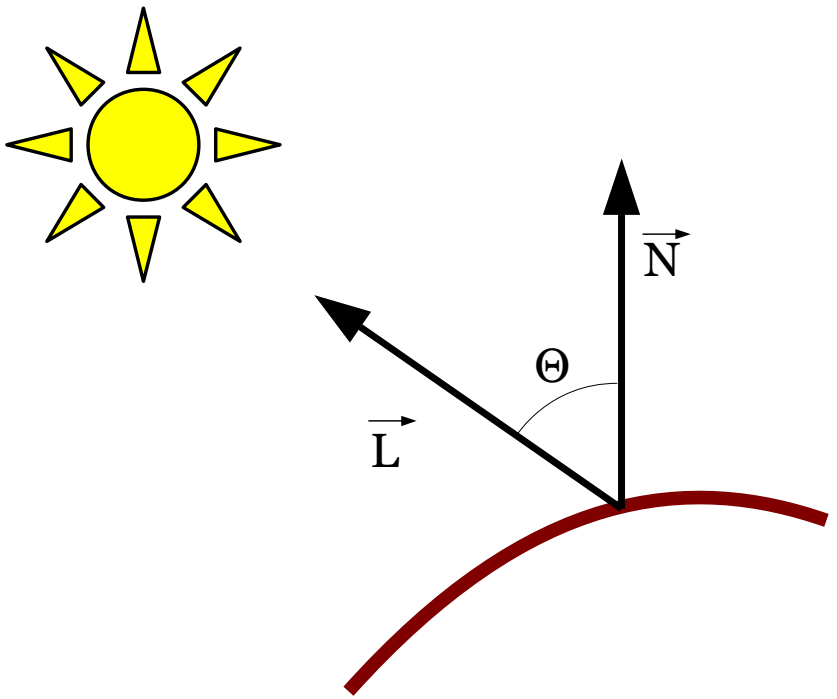
Model sfery oświetlonej światłem otoczenia o różnym natężeniu:



$I_a = 0.2$ $I_a = 0.4$ $I_a = 0.6$ $I_a = 0.8$ $I_a = 1.0$

Światło rozproszenia

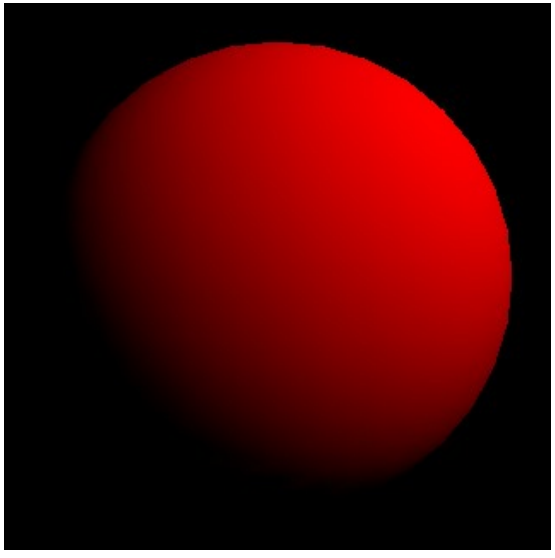
Natężenie **światła rozproszenia (ang. diffuse)** zależy bezpośrednio od cosinusa kąta θ między wektorem normalnym a wektorem padania światła $\overrightarrow{L}$, natężenie źródła światła I_p, oraz współczynnika odbicia światła rozproszenia przez obiekt k_d (właściwość materiału).



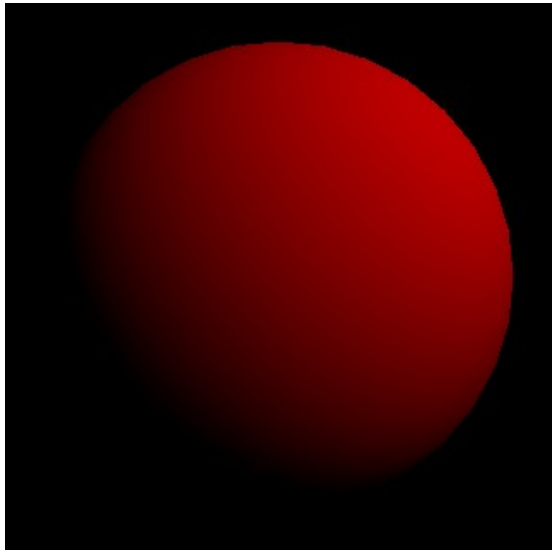
$$I = I_p * k_d * cos\theta = I_p * k_d * (\overrightarrow{N} \cdot \overrightarrow{L})$$

Światło rozproszenia

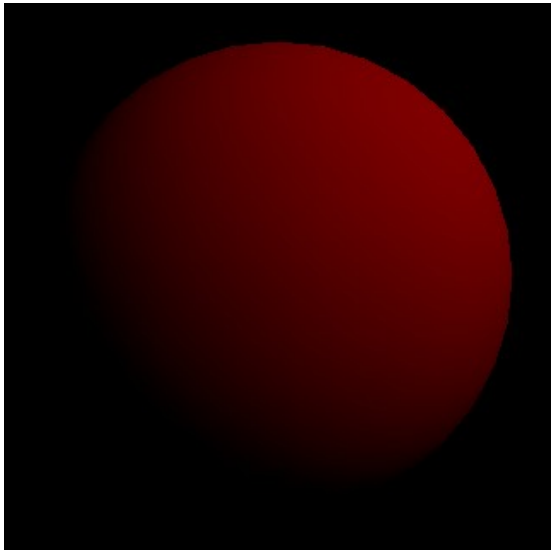
Model sfery oświetlonej światłem rozproszenia o różnym natężeniu:



$I_p = 1.0$



$I_p = 0.75$



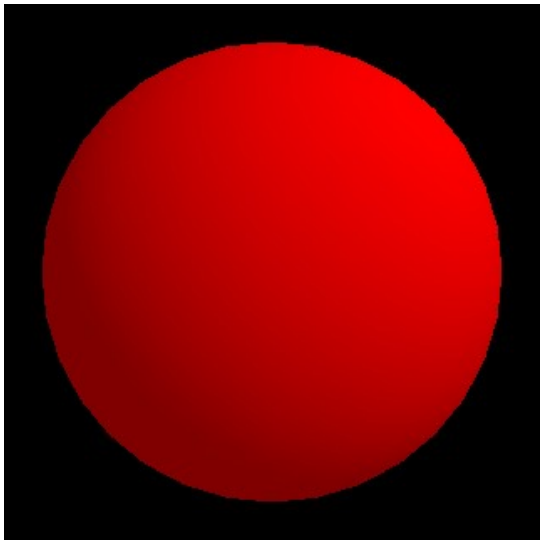
$I_p = 0.5$

Światło mieszane

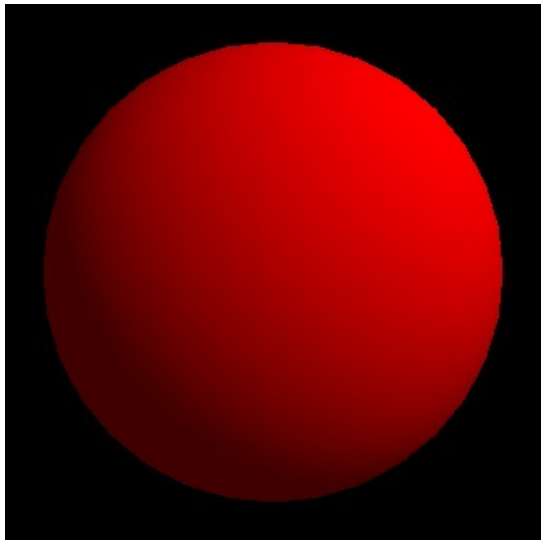
Najlepsze efekty daje jednoczesne zastosowanie **dwóch modeli oświetlenia**, gdzie światło rozproszenia odpowiada za oświetlone części obiektów, a światło otoczenia za zaciemnione:

$$I = I_a * k_a + I_p * k_d * (\overrightarrow{N} \cdot \overrightarrow{L})$$

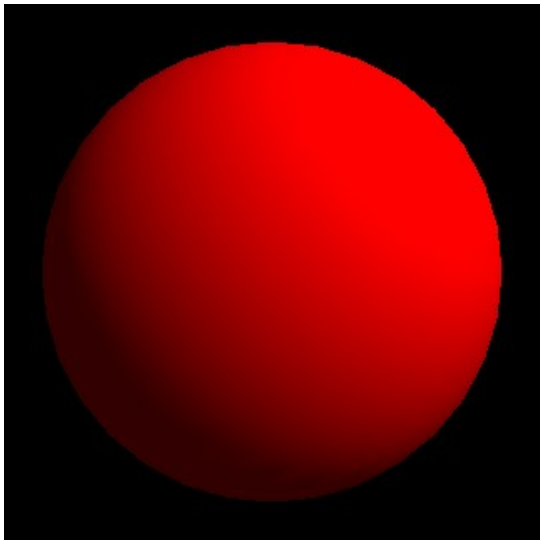
Model sfery oświetlonej światłem mieszanym o różnych proporcjach:



$I_a = 0.5, I_p = 0.5$



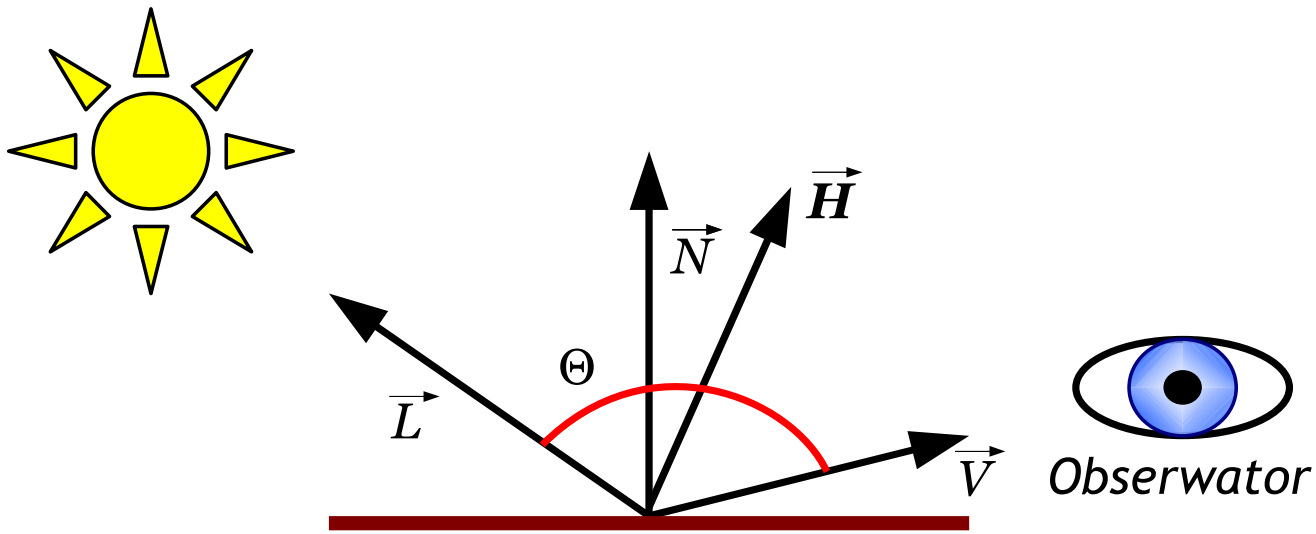
$I_a = 0.25, I_p = 0.75$



$I_a = 0.2, I_p = 1.0$

Odbicie zwierciadlane

Odbicie zwierciadlane (ang. specular) umożliwia uzyskanie rozbłysków na powierzchniach przedmiotów, które mają być widoczne jako gładkie, odbijające światło. Odbicie zwierciadlane uzupełnia pozostałe modele oświetlenia, tzn. światło otoczenia i rozproszenia. Najczęściej realizowane jest jako algorytm **Blinna-Phonga**.



Podstawą do wyliczenia jasności rozbłysku jest **wektor połówkowy** $\vec{H}$, wyznaczany jako połowa sumy wektorów padania światła $\vec{L}$ i wektora wskazującego położenie obserwatora $\vec{V}$.

Odbicie zwierciadlane

Jasność rozbłysku, w przeciwieństwie do światła otoczenia i rozproszenia, zależy od położenia obserwatora względem sceny.

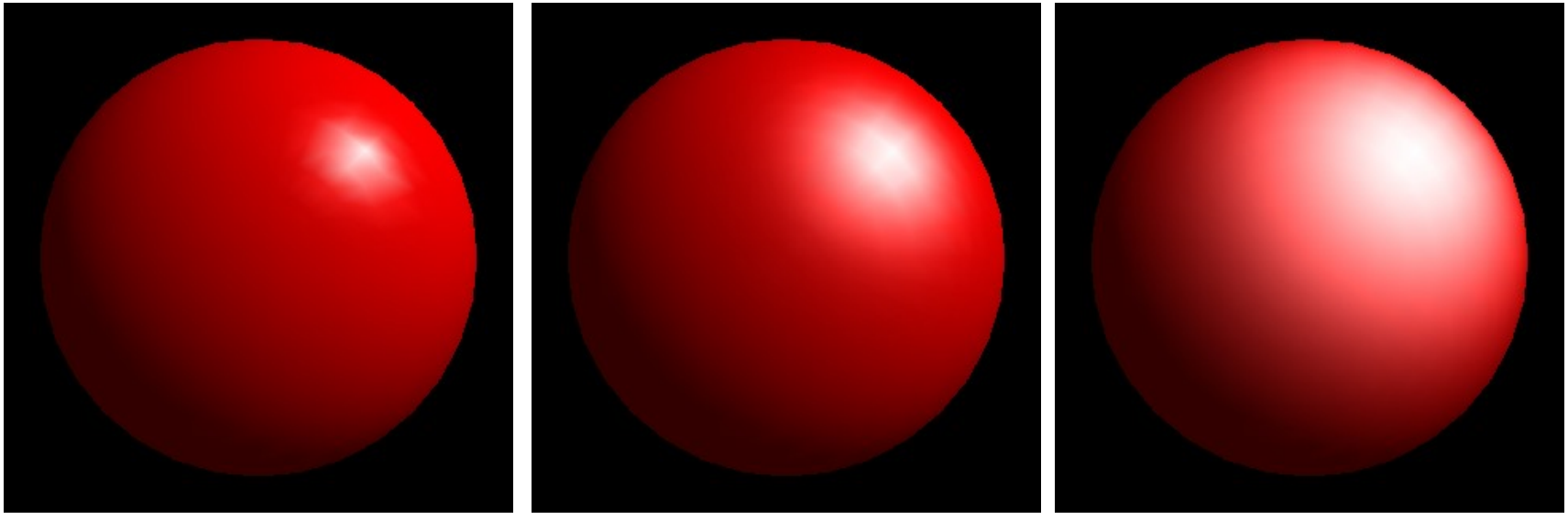
$$\overrightarrow{H} = \frac{\overrightarrow{L} + \overrightarrow{V}}{2} \qquad I_s = k_s * (\overrightarrow{H} \cdot \overrightarrow{N})^n$$

Parametr n (właściwość materiału) określa wielkość (skupienie) plamy rozbłysku na powierzchni obiektu. Parametr ten może określać stopień "chropowatości" materiału pokrywającego obiekt:

- Dla małych wartości n (np. 1), plama rozbłysku ma dużą powierzchnię i obejmuje duży kąt widzenia.
- Dla dużych wartości n (np. 100), plama jest niewielka i widoczna tylko pod małym kątem.

Odbicie zwierciadlane

Model sfery oświetlonej światłem otoczenia, rozproszenia i z rozbłyskami dla różnych wartości parametru n:



$n = 128$ $n = 32$ $n = 8$

Parametry oświetlenia

Kierunek z którego pada światło może być zdefiniowany na dwa sposoby:

1. **Światło punktowe** – promienie światła wychodzą promieniście z punktu w którym znajduje się źródło światła (przykład: żarówka).
2. **Światło kierunkowe** – promienie światła są równoległe, padają zawsze z tego samego kierunku (przykład: światło słoneczne).

Inne parametry oświetlenia:

W symulacji oświetlenia mogą brać udział również inne parametry, jak tłumienie światła w funkcji odległości lub symulacja reflektorów.

Wielokrotne źródło światła:

Jeżeli scena oświetlona jest przez kilka źródeł światła, ich efekty sumują się. Składowa światła otoczenia nie podlega sumowaniu – jest stała dla całej sceny.

Aby oświetlenie mogło być poprawnie symulowane, definicja obiektu 3D musi zostać uzupełniona o **wektor normalny**.

Sześcian:

```
#define MOJ_FVF (D3DFVF_XYZ | D3DFVF_NORMAL)

struct MOJ_VERTEX {
    FLOAT x, y, z; // Pozycja wierzchołka
    D3DVECTOR normal; // Wektor normalny
};

// Sześcian opisany z pomocą 24 wierzchołków
MOJ_VERTEX cube_vert[] = {
    { -1.0f, -1.0f, 1.0f,   0.0f, 0.0f, 1.0f},
    { 1.0f, -1.0f, 1.0f,   0.0f, 0.0f, 1.0f},
    { -1.0f, 1.0f, 1.0f,   0.0f, 0.0f, 1.0f},
    { 1.0f, 1.0f, 1.0f,   0.0f, 0.0f, 1.0f},
    /* ... */
```

Jeżeli scena 3D jest oświetlona, do wyznaczania wynikowego koloru obiektów niezbędny jest **materiał** (zmienna typu `D3DMATERIAL9`). Pola zmiennej typu `D3DMATERIAL9` to między innymi:

- `D3DXCOLOR Ambient` – współczynnik (kolor) odbicia światła otoczenia.
- `D3DXCOLOR Diffuse` – współczynnik (kolor) odbicia światła rozproszenia.
- `D3DXCOLOR Specular` – współczynnik (kolor) odbicia światła powodującego rozbłyski.
- `float Power` – współczynnik określający skupienie plamy rozbłysków, zgodnie z modelem Blinna-Phonga.

Użycie materiału:

Wybranie przygotowanego materiału odbywa się przez wywołanie metody **SetMaterial()**. Wszystkie tworzone obiekty i prymitywy pokrywane są ostatnio wybranym materiałem.

Przykład

Przygotowanie i użycie materiału:

```
void ustaw_mat(float r, float g, float b) {
    D3DMATERIAL9 material;
    ZeroMemory(&material, sizeof(D3DMATERIAL9));

    // Składowa otoczenia:
    material.Ambient = D3DXCOLOR(r, g, b, 1.0f);
    // Składowa rozproszenia:
    material.Diffuse = D3DXCOLOR(r, g, b, 1.0f);
    // Jasność rozbłysków:
    material.Specular = D3DXCOLOR(1.0f, 1.0f, 1.0f, 1.0f);
    // Skupienie plamy rozbłysków:
    material.Power = 16.0f;

    d3ddev->SetMaterial(&material);
    d3ddev->SetRenderState(D3DRS_SPECULARMATERIALSOURCE,
                           D3DMCS_MATERIAL);
}
```

Globalne parametry oświetlenia ustawiane są z pomocą metody **SetRenderState()**. Są to między innymi:

- `D3DRS_LIGHTING (TRUE, FALSE)` – globalne włączenie/wyłączenie oświetlenia.
- `D3DRS_AMBIENT (D3DCOLOR)` – kolor światła otoczenia.
- `D3DRS_SPECULARENABLE (TRUE, FALSE)` – globalne włączenie/wyłączenie symulacji rozbłysków.
- `D3DRS_SPECULARMATERIALSOURCE (D3DMATERIALCOLORSOURCE)` – aby właściwości rozbłysków były powiązane z materiałem, należy ustawić na `D3DMCS_MATERIAL`.

Parametry źródła światła

Parametry jednego źródła światła opisane są z pomocą zmiennej typu `D3DLIGHT9`. Pola zmiennej typu `D3DLIGHT9` to między innymi:

- `D3DLIGHTTYPE Type` – określa rodzaj źródła światła:
  - `D3DLIGHT_DIRECTIONAL` – kierunkowe,
  - `D3DLIGHT_POINT` – punktowe,
  - `D3DLIGHT_SPOT` – reflektor (spot).
- `D3DCOLORVALUE Diffuse` – natężenie (kolor) światła rozproszenia.
- `D3DCOLORVALUE Specular` – natężenie (kolor) światła generującego rozbłyski.
- `D3DCOLORVALUE Ambient` – natężenie (kolor) światła otoczenia.
- `D3DVECTOR Position` – pozycja źródła światła dla światła punktowego.
- `D3DVECTOR Direction` – kierunek padania światła dla światła kierunkowego.

Włączenie źródła światła

Źródła światła identyfikowane są przez numery (począwszy od zera). Każde źródło światła należy niezależnie włączyć. Odbywa się to w dwóch krokach:

1. Ustawienie parametrów źródła światła, przygotowanych w zmiennej typu `D3DLIGHT9`, z pomocą metody **SetLight()**.
2. Indywidualne włączenie każdego źródła światła z pomocą metody **LightEnable()**.

Przykład

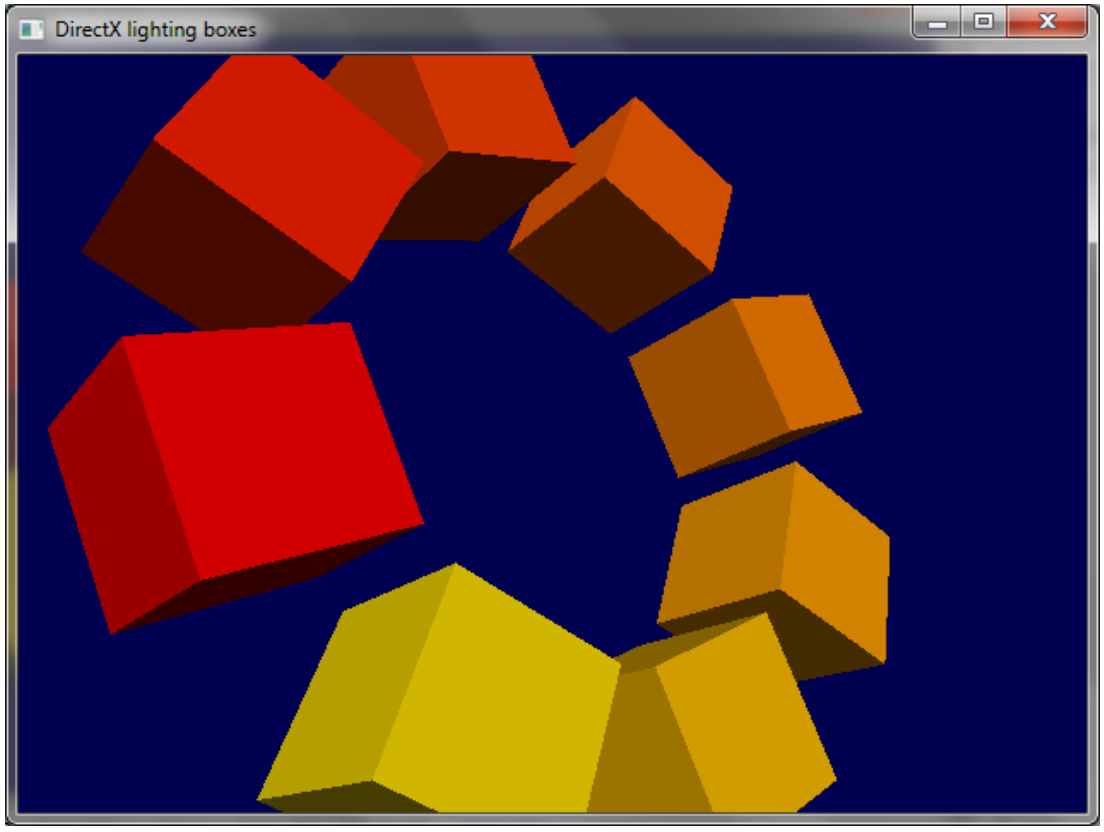
Ustawienie oświetlenia:

```
void initLight(HWND hWnd) {
    // WŁĄCZENIE oświetlenia:
    d3ddev->SetRenderState(D3DRS_LIGHTING, TRUE);

    // Parametry źródła światła nr 0:
    D3DLIGHT9 light;
    ZeroMemory(&light, sizeof(light));
    light.Type = D3DLIGHT_DIRECTIONAL;
    light.Ambient = D3DXCOLOR(0.2f, 0.2f, 0.2f, 1.0f);
    light.Diffuse = D3DXCOLOR(0.8f, 0.8f, 0.8f, 1.0f);
    light.Specular = D3DXCOLOR(1.0f, 1.0f, 1.0f, 1.0f);
    light.Direction = D3DXVECTOR3(0.0f, -1.0f, -1.0f);

    // Ustawione parametry będą dotyczyły źródła numer 0:
    d3ddev->SetLight(0, &light);

    // Włączenie światła numer 0:
    d3ddev->LightEnable(0, TRUE);
    d3ddev->SetRenderState(D3DRS_SPECULARENABLE, TRUE);
```



Przykładowy program

Windows

Teksturowanie

Teksturowanie:

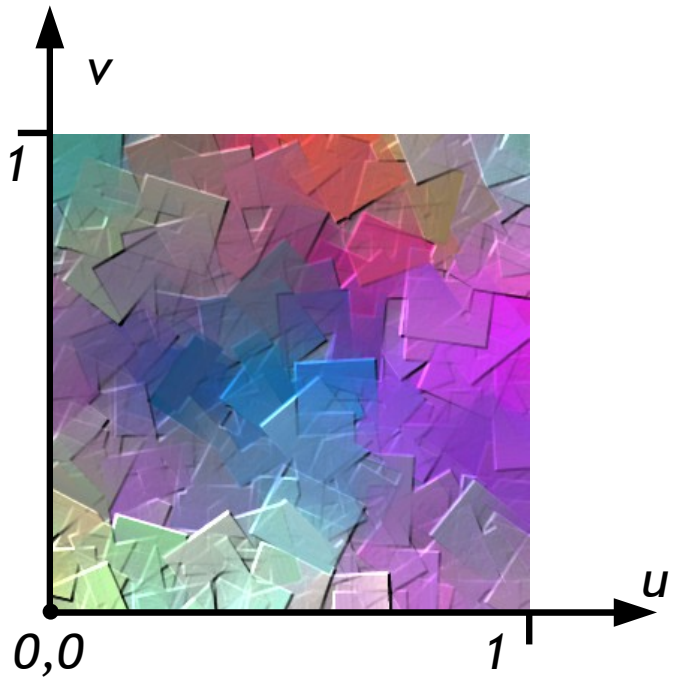
Teksturowanie, czyli metoda odwzorowania tekstury, stworzone zostało jako alternatywa dla bezpośredniego modelowania szczegółów za pomocą wielokątów i innych prymitywów geometrycznych. Metoda została opracowana przez *Catmulla* i ulepszona przez *Blinna* i *Newella*.

Tekstura:

Tekstura jest bitmapą nakładaną na powierzchnię. Poszczególne jej elementy nazywane są **tekselami**.

Współrzędne tekstury

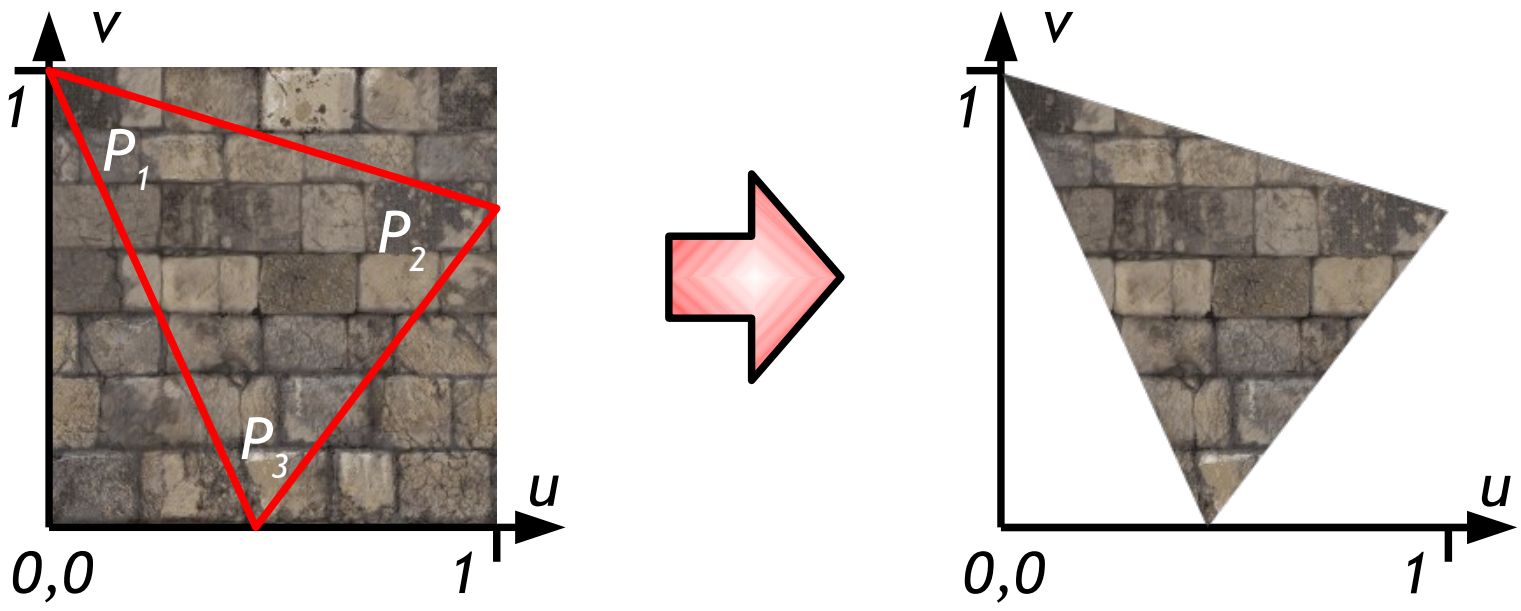
Prostokątna mapa tekstury określana jest we własnym układzie współrzędnych **u** i **v** (lub **s** i **t**). Przy renderingu poszczególnych pikseli odpowiednie teksele są używane do zastąpienia lub skalowania jednej lub więcej właściwości materiału powierzchni.



Współrzędne tekstury

Dla każdego wierzchołka wielokąta należy określić współrzędne **u** i **v** dla tekstury.

P_1:	u=0,	v=1
P_2:	u=1,	v=0.7
P_3:	u=0.45,	v=0



Współrzędne i rozmiar tekstury

Rozmiar tekstury:

Współrzędne **u** i **v** nie mają nic wspólnego z rozmiarem samej tekstury wyrażonej w pikselach. Rozmiary tekstur są zawyczaj wielokrotnościami liczby 2. Tekstury nie muszą być kwadratowe.

Zakres współrzędnych:

Współrzędna tekstury dla wierzchołka może mieć wartość większą niż 1, wtedy tekstura jest (zazwyczaj) powtarzana.

Mipmapping (łac. mip – wiele) do teksturowania obiektów wykorzystuje nie jedną, a cały zestaw tekstur różnych rozmiarów. Zazwyczaj jest to tekstura w oryginalnych rozmiarach i jej pomniejszone kopie.

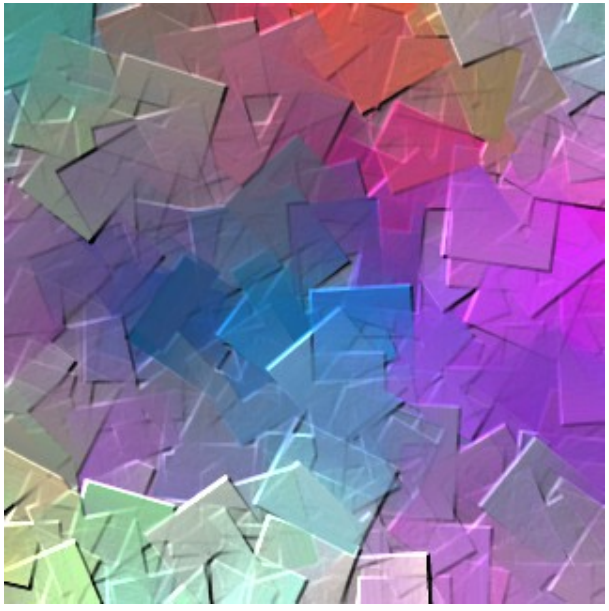
Tekstura w odpowiednim rozmiarze jest wybierana automatycznie, zależnie od wielkości danego wielokąta po rzutowaniu na ekran.

Zalety stosowania tej techniki to między innymi:

- Polepszenie jakości – zmniejszenie liczby i widoczności artefaktów.
- Przyspieszenie generowania grafiki – do mniejszych wielokątów wykorzystywane są mniejsze tekstury.

Poziomy tekstury

0



1



2 3 4 5 6 7



Poziom	Rozmiar
0 (oryginał)	128 x 128
1	64 x 64
2	32 x 32
3	16 x 16
4	8 x 8
5	4 x 4
6	2 x 2
7	1 x 1

Dane każdego wierzchołka należy uzupełnić o **współrzędne tekstury**.

Sześcian:

```
#define MOJ_FVF (D3DFVF_XYZ | D3DFVF_NORMAL | D3DFVF_TEX1)

struct MOJ_VERTEX {
    FLOAT x, y, z; // Pozycja wierzchołka
    D3DVECTOR normal; // Wektor normalny
    FLOAT u, v; // Współrzędne tekstury
};

// Sześcian opisany z pomocą 24 wierzchołków
MOJ_VERTEX cube_vert[] = {
    { -1.0f, -1.0f, 1.0f,  0.0f, 0.0f, 1.0f,  0.0f, 0.0f, },
    { 1.0f, -1.0f, 1.0f,  0.0f, 0.0f, 1.0f,  1.0f, 0.0f, },
    /* ... */
```

Wczytywanie tekstur

"Uchwyt" tekstury:

"Uchwytem" tekstury jest zmienna typu `LPDIRECT3DTEXTURE9`. Tekstury można w prosty sposób wczytywać z plików graficznych z pomocą funkcji `D3DXCreateTextureFromFile()`.

```
HRESULT D3DXCreateTextureFromFile(LPDIRECT3DDEVICE9 pDevice,
        LPCTSTR pSrcFile, LPDIRECT3DTEXTURE9* ppTexture)
```

- **pDevice** – urządzenie Direct3D.
- **pSrcFile** – nazwa pliku z obrazem tekstury, obsługiwane formaty to: bmp, dds, dib, hdr, jpg, pfm, png, ppm, tga.
- **ppTexture** – wskaźnik na zmienną w której umieszczony zostanie "uchwyt" tekstury.

Jeżeli wczytanie tekstury się powiodło, funkcja zwraca stałą `D3D_OK`.

Przykład

Wczytywanie tekstury:

```
LPDIRECT3DTEXTURE9 tex1;
HRESULT res;

res = D3DXCreateTextureFromFile(d3ddev,
                    "tekstura1.jpg", &tex1);

if(res != D3D_OK) {
    MessageBox(hWnd, "Błąd ładowania tekstury!", "Info",
                    MB_ICONEXCLAMATION | MB_OK);
    exit(0);
}
```

Filtrowanie tekstur

```
HRESULT SetSamplerState(DWORD Sampler,
                        D3DSAMPLERSTATETYPE Type, DWORD Value)
```

- **DWORD Sampler** – nr jednostki teksturującej lub 0.
- **D3DSAMPLERSTATETYPE Type** – który filtr ustawiamy? Wybrane możliwości:
  - `D3DSAMP_MINFILTER` – filtr zmniejszający,
  - `D3DSAMP_MAGFILTER` – filtr powiększający,
  - `D3DSAMP_MIPFILTER` – filtr dla mipmap.
- **DWORD Value** – jak filtrować? Wybrane możliwości:
  - `D3DTEXF_POINT` – domyślne (brzydkie) skalowanie,
  - `D3DTEXF_LINEAR` – filtr liniowy,
  - `D3DTEXF_ANISOTROPIC` – filtr anizotropowy.

Filtr anizotropowy

Jeżeli wybrano filtr anizotropowy, należy ustawić jego siłę, domyślnie jest to 1, co jest niezauważalne. Zmiana poprawiająca jakość filtrowania:

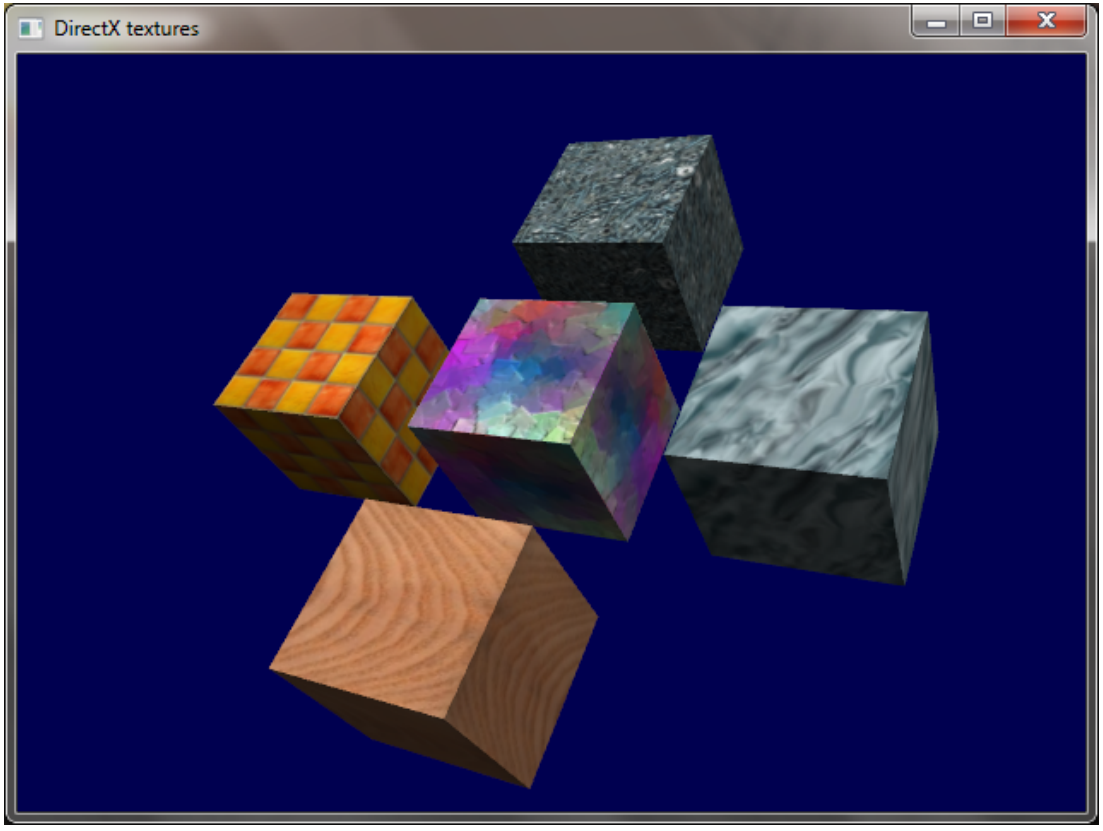
```
d3ddev->SetSamplerState(0, D3DSAMP_MAXANISOTROPY, 8);
```

Nakładanie tekstury na obiekt

Wybór tekstury która ma zostać nałożona na obiekt, następuje z pomocą metody `SetTexture()`.

Nakładanie tekstury:

```
d3ddev->SetTexture(0, tex1);
d3ddev->DrawIndexedPrimitive(D3DPT_TRIANGLELIST, 0, 0,
                             24, 0, 12);
```



Przykładowy program


Windows




KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



Politechnika Świętokrzyska

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

" Program Rozwojowy Potencjału Dydaktycznego Politechniki Świętokrzyskiej w Kielcach: *kształcenie w nowoczesnych obszarach techniki* "

Program Operacyjny Kapitał Ludzki, Priorytet IV Szkolnictwo wyższe i nauka, Działanie 4.1. Wzmocnienie i rozwój potencjału dydaktycznego uczelni oraz zwiększenie liczby absolwentów kierunków o kluczowym znaczeniu dla gospodarki opartej na wiedzy, Poddziałanie 4.1.1. Wzmocnienie potencjału dydaktycznego uczelni.

Umowa UDA–POKL.04.01.01-00395/09-00