Grafika Komputerowa

- dr inż. Piotr M. Szczypiński
- e-mail: pms@ck-sg.p.lodz.pl
- http://www.eletel.p.lodz.pl/~pms/

Literatura

- Piotr Makowski, *Grafika Komputerowa 3D*, http://server.eletel.p.lodz.pl/~makowski/graph.html
- Hartfiel Hobbs, *Elementary linear algebra*
- John Levine, *Programowanie plików graficznych w C/C++*, Wiley
- Władysław Skarbek, *Metody reprezentacji obrazów cyfrowych*, PLJ
- Charles Petzold, *Programowanie Windows,* RM
- V. Toth, *Programowanie Windows 98/NT - księga eksperta*, Helion
- K. Ruta, *Programowanie w Windows*, Helion
- R.S.Wright, M.Sweet, *OpenGL - księga eksperta*, Helion
- J. Sanchez, M. Canton, *Direct 3D - biblia*, Helion
- Nate Robins, *Tools for teaching about OpenGL*, http://www.xmission.com/~nate/tutors.html

Układ treści wykładu

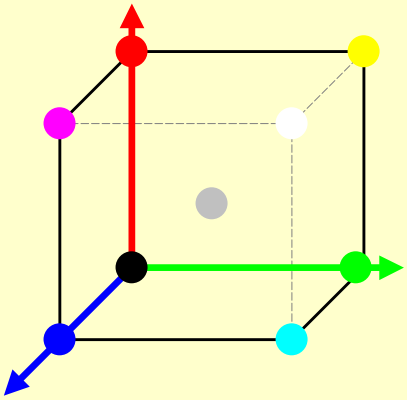
- Modele koloru
- Obrazy rastrowe
  - metody uzyskiwania, formaty zapisu, wizualizacja, zastosowania, kompresji obrazów rastrowych
- Obrazy wektorowe
  - metody uzyskiwania, formaty zapisu, wizualizacja, zastosowania
- Metapliki
- Systemy graficzne, graficzny interfejs użytkownika (GUI)
- Programowanie grafiki 2D
  - języki programowania, biblioteki obsługi formatów graficznych
- Obrazy trójwymiarowe - 3D
  - rastrowe: uzyskiwanie, formaty zapisu, wizualizacja, zastosowania
  - wektorowe: modelowanie trójwymiarowe, metody transformowania i rzutowania
- Programowanie grafiki 3D
  - biblioteki OpenGL i Direct3D

Modele składowych koloru

- RGB - Red Green Blue
- CMY - Cyan Magenta Yellow
- CMYK - Cyan Magenta Yellow Black
- HSB - Hue Saturation Brightnes
  (HLS - Hue Luminance Saturation)
- YIQ, YC_BC_R, YD_BD_R

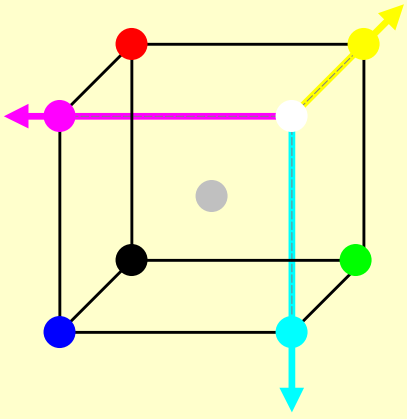
Model RGB

- Model **RGB** jest modelem addytywnym, w którym barwę otrzymuje się przez sumowanie składowych światła o różnej długości fali.
- Składowe to barwy:
  - czerwona (R)
  - zielona (G)
  - niebieska (B)
- Zastosowanie modelu:
  - lampa kineskopowa w telewizji



Model CMY

- Model **CMY** jest modelem subtraktywnym, w którym barwę wynikową otrzymuje się przez odejmowanie od światła białego (wytłumianie) składowych o różnej długości fali.
- Barwy są tłumione:
  - czerwona (R) przez (C)
  - zielona (G) przez (M)
  - niebieska (B) przez (Y)
- Zastosowanie modelu:
  - fotografia barwna

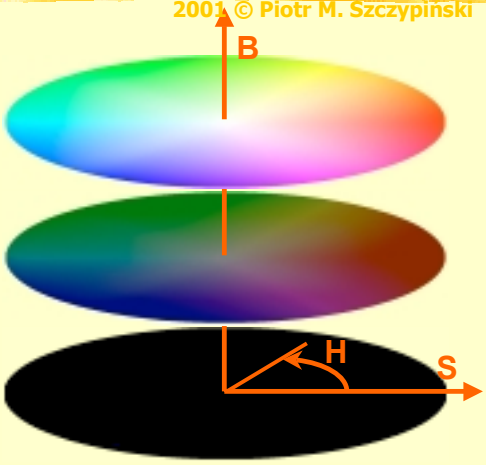


Model CMYK

- Model **CMYK** jest modyfikacją modelu CMY związaną z ograniczeniami farb drukarskich. Złożenie wszystkich farb nie zapewnia prawidłowego oddania na papierze czerni, konieczne jest dodanie dodatkowej składowej określającej ile czarnej farby ma być dodane do wydruku.
- Zastosowanie modelu:
  - druk barwne z zastosowaniem farb drukarskich i tuszy

Model HSB (HLS)

- Model **HSB** jest wynikiem prac prowadzonych nad systemem telewizji kolorowej NTSC i PAL. W modelu rozdzielono składową luminancji (B) od składowych chrominancji (HS), gdzie S jest składową nasycenia koloru natomiast H jest składową kątową barwy.
- Zastosowanie modelu:
  - modulacja kwadraturowa chrominancji w systemach telewizji NTSC i PAL



Inne modele koloru

YIQ (NTSC)
Y = 0,299R+0,587G+0,114B
I = 0,596R-0,274G-0,322B
Q = 0,211R-0,528G+0,312B

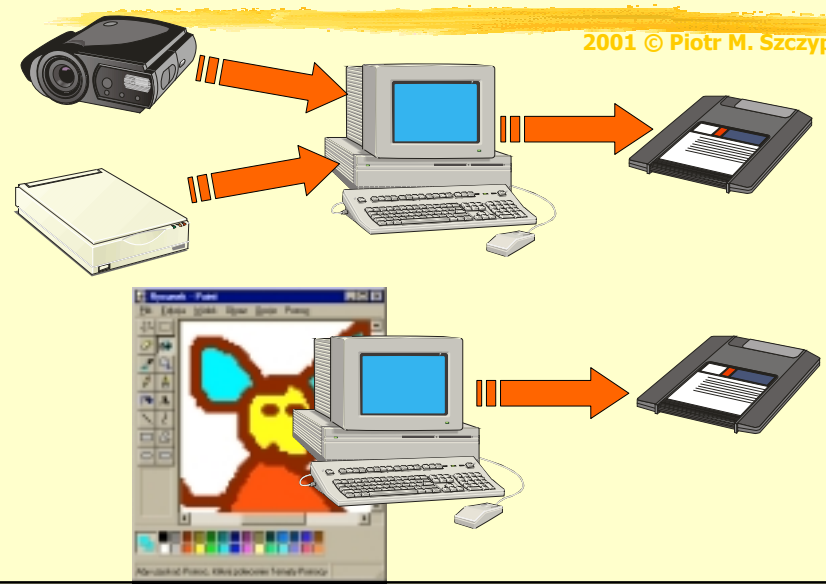
YD_BD_R (SECAM)
Y = 0,30R+0,59G+0,11B
D_B = 1,5(R-Y)
D_R = -1,9(B-Y)

YC_BC_R (CCITT)
Y = 0,299R+0,587G+0,114B
C_B = 1,772(R-Y)
C_R = 1,402(B-Y)

Definicja obrazu rastrowego

- Obrazy rastrowe powstają w wyniku akwizycji obrazów rzeczywistych lub syntezy programowej, w wyniku fizycznych lub symulowanych procesów dyskretyzacji i kwantyzacji.
  - Proces dyskretyzacji wprowadza podział przestrzeni analogowego obrazu na elementarne części zwane pikselami (ang. *pixel – picture element*). Dla każdego piksela jest wyznaczana przez próbkowanie lub przez fizyczne całkowanie jego jasność w pewnym zakresie widma elektromagnetycznego.
  - Proces kwantyzacji obrazu zamienia analogową wartość jasności na jeden z poziomów dyskretnych, dając w wyniku wektor składowych wartości danego piksela (punkt obrazu cyfrowego) określających jego jasność, składowe RGB, itp.

Metody uzyskiwania obrazu rastrowego

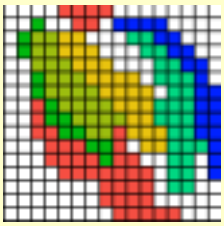


Rodzaje obrazów rastrowych

- Rodzaje obrazów:
  - binarne/czarno-białe (np. z faxu),
  - o odcieniach szarości / monochromatyczne,
  - kolorowe
  - kolorowe z paletą barw
  - i wiele innych

Zapis obrazów rastrowych

- Najczęściej spotykany sposób zapisu pliku z obrazem rastrowym:
  - część informacyjna (zazwyczaj w tzw. nagłówku pliku),
  - paleta barw - LUT (opcjonalnie),
  - macierz z informacją o kolejnych pikselach



```
42 4D 36 05 00 00 00 00 00 00 36 04 00 00 28 00
00 00 10 00 00 00 10 00 00 00 01 00 08 00 00 00
00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 80
00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80
```

```
40 00 40
40 60 00
40 40 80
00 40 40
A0 00 40
40 C0 00
40 40 E0
   ...
   ...
   ...
```

```
FF FF FF FF 4F 4F 4F FF FF FF FF FF FF FF FF
FF FF 31 FF FF 4F 4F FF FC FC FC FC FF FF FF
FF 30 35 35 37 37 FF FF FF BA BA FC FC FF FF
FF 30 35 35 37 37 37 FF FF FF BA BA FC FC FF
FF FF 35 35 35 37 37 37 FF FF FF BA BA FC FC
FF FF 31 35 35 35 37 37 37 FF BA BA BA FC FF
FF FF 30 35 35 35 35 37 37 37 FF BA BA FC FC
FF FF 4F 30 35 35 35 35 37 37 BA BA BA FC FC
FF FF 4F 30 30 35 35 35 35 37 37 BA BA BA FC
FF FF 4F 4F 30 30 35 35 4F 37 37 37 BA BA FC
FF FF FF 4F 4F 4F 30 30 4F FF 37 37 BA BA BA
FF FF FF 4F 4F 4F 4F 30 4F 4F FF FF BA BA FF
FF FF FF FF 4F 4F 4F 4F 4F 4F 4F FF BA BA FF
FF FF FF FF FF FF FF 4F 4F 4F 4F FF BA BA FF
FF FF FF FF FF FF FF 4F 4F 4F 4F FF FF FF FF
FF FF FF FF FF FF FF FF 4F 4F 4F 4F FF FF FF
```

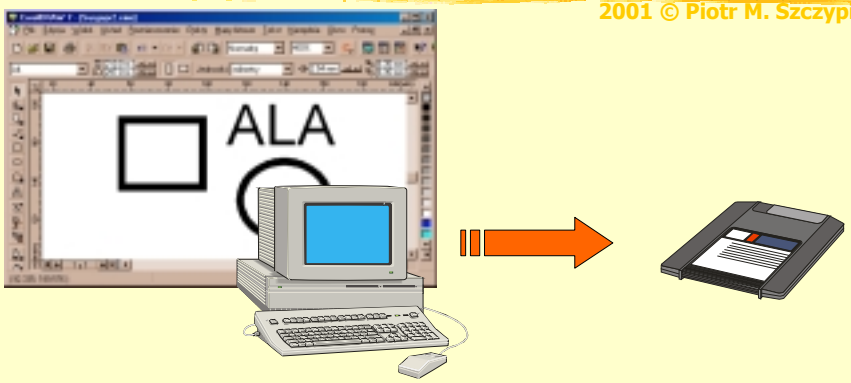
Kompresja

- Kompresja obrazów rastrowych (wybrane standardy)
  - Kompresja bezstratna
    - Transmisja faksowa (CCITT bilevel encoding)
    - Kodowanie obrazów z paletą barw LZW (CompuServe GIF)
  - Kompresja stratna
    - Kodowanie obrazów kolorowych i monochromatycznych (JPEG - Joint Photographic Expert Group)

Obrazy wektorowe - próba definicji

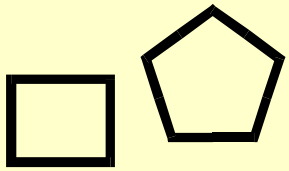
- Obrazy wektorowe są zbiorem pierwotnych składników (tzw. prymitywów) takich jak odcinki, łuki, znaki alfanumeryczne i podstawowe figury geometryczne.
- Dane o obrazie zawierają informację o rodzaju, wymiarach oraz położeniu poszczególnych składników na powierzchni rysunku lub w przestrzeni obrazu trójwymiarowego.

Metody uzyskiwania obrazów wektorowych



- Obrazy wektorowe tworzone są zazwyczaj sztucznie za pomocą komputera.

Zapis obrazów wektorowych

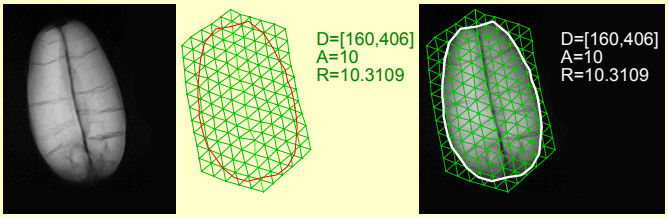


- Przykład zapisu pliku wektorowego w standardzie HPGL

```
IN;
VS32,1;
VS32,2;
VS32,3;
VS32,4;
VS32,5;
VS32,6;
VS32,7;
VS32,8;
WU0;
PW0.350,1;
PW0.350,2;
PW0.350,3;
PW0.350,4;
PW0.350,5;
PW0.350,6;
PW0.350,7;
PW0.350,8;
SP1;
PU1446 2116;
PD2011 1705;
PD2576 1295;
PD2361 630;
PD2144 -31;
PD1446 -31;
PD748 -31;
PD532 630;
PD316 1295;
PD881 1705;
PD1446 2116;
SP1;
PU-1964 947;
PD-288 947;
PD-288 -453;
PD-1964 -453;
PD-1964 947;
SP0;
```

Metapliki (same zalety)

- Zaawansowane graficzne systemy komputerowe posługują się metaplikami (ang. *metafiles*) zawierającymi listy poleceń dla systemu graficznego.
- Metapliki pozwalają między innymi na połączenie obu sposobów reprezentacji obrazów, jednoczesne wyświetlenie obrazu rastrowego i umieszczenie na jego powierzchni widocznych znaczników i opisów w formacie wektorowym.

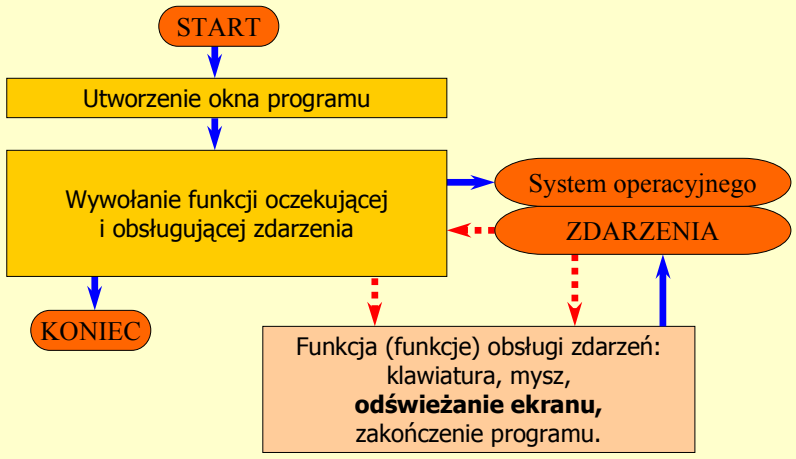


obraz rastrowy obraz wektorowy metaplik

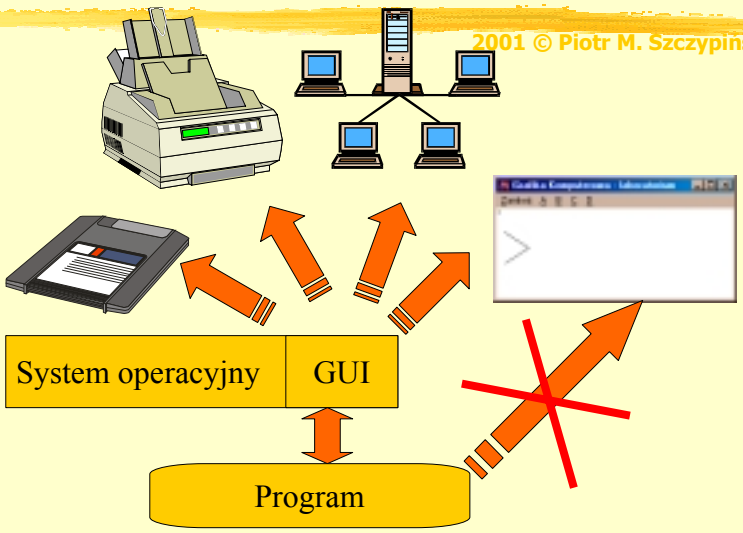
Komputerowe systemy graficzne

- Środowiska wieloprogramowe (kilka aplikacji działających równocześnie) i wielowątkowe
  - konieczność kontroli dostępu poszczególnych aplikacji i wątków do urządzeń graficznych
  - podział ekranu na fragmenty, do których dostęp mają poszczególne aplikacje
- Praca w sieci
  - ustalenie standardów przesyłania grafiki w sieci
- Różnorodność dołączonych do systemu komputerowego urządzeń graficznych
  - ujednolicenie metod tworzenia grafiki

Program w Windows
(pętla obsługi zdarzeń)



GUI - graficzny interfejs użytkownika

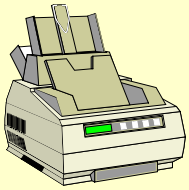


Kontekst urządzenia (w MS-Windows)


```
// Kontekst odświerzania wnętrza okna
kontekst = BeginPaint(okno, &struktrys );

// Wywołania procedur graficznych

EndPaint( okno, &struktrys );
```



```
// Kontekst drukarki
PRINTDLG pd;
PrintDlg(&pd); kontekst = pd.hDC;
StartPage(kontekst);

// Wywołania procedur graficznych

EndPage(kontekst);
DeleteDC(kontekst);
```

Przykłady funkcji graficznych (w MS-Windows)

SetPixelV(kontekst, x, y, RGB(r, g, b));
MoveToEx(kontekst, x, y, NULL);
LineTo(kontekst, x, y);
Rectangle(kontekst, x1, y1, x2, y2);
TextOut(kontekst, x, y, "Napis", z);

- Funkcje graficzne pozwalają sterować pracą różnych urządzeń graficznych poprzez utworzony kontekst tych urządzeń

Przykłady funkcji graficznych (w MS-Windows)

POINT tablica[n];
...
tablica[k].x = *współrzędna_x_punktu_k;*
tablica[k].y = *współrzędna_y_punktu_k;*
...
Polygon(kontekst, tablica, n);

Przykłady funkcji graficznych (w MS-Windows)


```
HPEN pen;
HBRUSH brush;
HGDIOBJ oldpen, oldbrush;
pen = CreatePen(PS_SOLID, 1, RGB(rp, gp, bp)); // rp, gp, bp są skład. kolorów
brush = CreateSolidBrush(RGB(rb, gb, bb)); // rb, gb, bb są skład. kolorów
oldpen = SelectObject(kontekst, pen);
oldbrush = SelectObject(kontekst, brush);
...
// Wywołanie funkcji graficznych korzystających z pióra i pędzla
// np.: Polygon(), LineTo(), Rectangle()
...
SelectObject(kontekst, oldbrush);
SelectObject(kontekst, oldpen);
DeleteObject(brush);
DeleteObject(pen);
```

Obraz rastrowy w systemie MS-Windows (bitmapa)

- BITMAPFILEHEADER
  - WORD bfType; **BM**
  - DWORD bfSize;
  - WORD bfReserved1;
  - WORD bfReserved2;
  - DWORD bfOffBits;
- BITMAPINFO
  - BITMAPINFOHEADER bmiHeader;
    - DWORD biSize;
    - LONG biWidth;
    - LONG biHeight;
    - WORD biPlanes;
    - WORD biBitCount
    - DWORD biCompression;
    - DWORD biSizeImage;
    - LONG biXPelsPerMeter;
    - LONG biYPelsPerMeter;
    - DWORD biClrUsed;
    - DWORD biClrImportant;
  - RGBQUAD bmiColors[];
- Macierz pikseli

Funkcja wyświetlania bitmapy

StretchDIBits(kontekst,
x1_okna, y1_okna,
x2_okna, y2_okna,
x1_bitmapy, y1_bitmapy,
x2_bitmapy, y2_bitmapy,
wsk_pam_bmp, wsk_pam_inf,
DIB_RGB_COLORS, SRCCOPY);

Metaplik w systemach MS-Windows

- Windows Metafile ver. 1.00
  - rozszerzenie pliku dyskowego: .wmf
- Windows Metafile ver. 3.00
  - wprowadzono w Windows 3.00
  - rozszerzenie pliku dyskowego: .wmf
- Enhanced Windows Metafile
  - wprowadzono w Windows 95
  - do wykorzystania przez aplikacje API32
  - rozszerzenie pliku dyskowego: .emf

Struktura Metapliku (wmf)

- METAFILEHEADER
  - DWORD key; 0x9AC6CDD7L
  - HANDLE hmf; 0
  - RECT bbox;
  - WORD inch;
  - DWORD reserved;
  - WORD checksum;

- METAHEADER
  - WORD mtType;
  - WORD mtHeaderSize;
  - WORD mtVersion;
  - DWORD mtSize;
  - WORD mtNoObjects;
  - DWORD mtMaxRecord;
  - WORD mtNoParameters;

- METARECORD
  - DWORD rdSize;
  - WORD rdFunction;
  - WORD rdParm[];

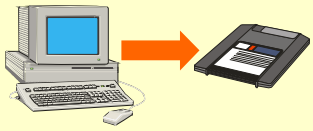
Struktura Metapliku (emf)

(WIN32 API)

- ENHMETAHEADER
  - DWORD iType;
  - DWORD nSize;
  - RECTL rclBounds;
  - RECTL rclFrame;
  - DWORD dSignature;
  - DWORD nVersion;
  - DWORD nBytes;
  - DWORD nRecords;
  - WORD nHandles;
  - WORD sReserved;
  - DWORD nDescription;
  - DWORD offDescription;
  - DWORD nPalEntries;
  - SIZEL szlDevice;
  - SIZEL szlMillimeters;
  - DWORD cbPixelFormat;
  - DWORD offPixelFormat;
  - DWORD bOpenGL;

- ENHMETARECORD
  - DWORD iType;
  - DWORD nSize;
  - DWORD dParm[];

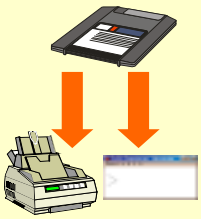
Windows Metafile



```
// ZAPIS METAPLIKU
// Kontekst pliku dyskowego wmf
kontekst=CreateMetaFile("plik.wmf");

// Wywołania procedur graficznych

CloseMetaFile(kontekst);
```



Utworzenie kontekstu urządzenia *kontekst*

```
// WYŚWIETLANIE METAPLIKU
HMETAFILE hwmf = GetMetaFile("plik.wmf");
PlayMetaFile(kontekst, hwmf);
DeleteMetaFile(hwmf);
```

Zamknięcie kontekstu urządzenia *kontekst*

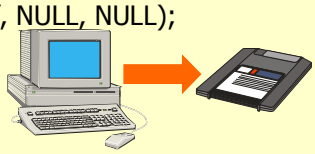
Windows Enhanced Metafile

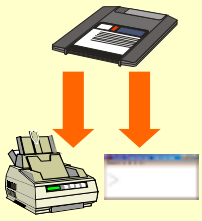
(WIN32 API)


```
// ZAPIS METAPLIKU
// Kontekst pliku dyskowego emf
kontekst=CreateEnhMetaFile(NULL, "plik.emf", NULL, NULL);

// Wywołania procedur graficznych

CloseEnhMetaFile(kontekst);
```





Utworzenie kontekstu urządzenia *kontekst*

```
// WYŚWIETLANIE METAPLIKU
HENHMETAFILE hemf = GetEnhMetaFile("plik.emf");
PlayEnhMetaFile(kontekst, hemf, &rect);
DeleteEnhMetaFile(hemf);
```

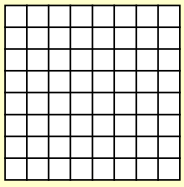
Zamknięcie kontekstu urządzenia *kontekst*

Obrazy trójwymiarowe

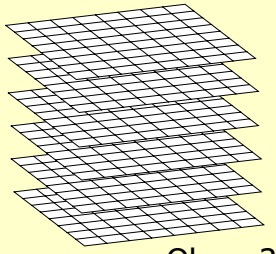
- Obrazy rastrowe
  - metody akwizycji rastrowych obrazów 3D
  - próba definicji obrazu rastrowego 3D (voxel)
  - zastosowania
- Obrazy wektorowe
  - metody otrzymywania trójwymiarowych obrazów wektorowych
  - metody obrazowania sceny 3D na płaszczyźnie
  - zastosowania

Trójwymiarowe obrazy rastrowe

- W procesie tworzenia trójwymiarowego obrazu cyfrowego wprowadza się podział pewnej objętości przestrzeni trójwymiarowej na elementarne części zwane voxelami (ang. *volume picture element*).
- Dla każdego voxela jest wyznaczana przez próbkowanie lub przez fizyczne całkowanie jego jasność, która odpowiada fizycznym cechom fragmentu przestrzeni trójwymiarowej.
- Analogowa wartość jasności voxela ulega kwantyzacji, zamianie na jeden z poziomów dyskretnych.
- Wynikowa informacja o obrazie zapisywana jest w postaci trójwymiarowej tablicy dyskretnych jasności poszczególnych voxeli obrazu.



Obraz 2D

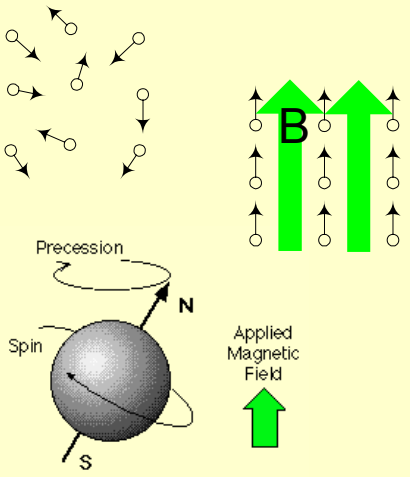


Obraz 3D

Metody akwizycji obrazów 3D

- Tomografia rezonansu magnetycznego (**MRI** - Magnetic Resonance Imaging)
- Rentgenowska tomografia komputerowa (**CT** - Computer Tomography)
- Echosonografia trójwymiarowa (**USG 3D**)

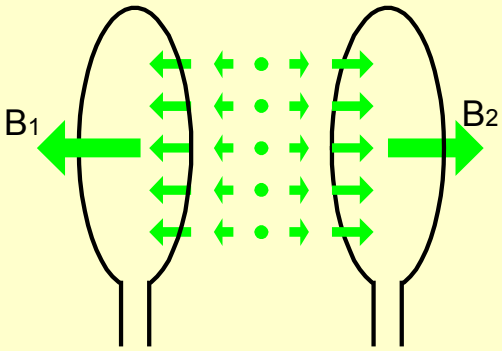
Zjawisko rezonansu magnetycznego (1)



- Spin protonów w jądrze atomowym
- Porządkowanie kierunku spinów pod wpływem pola magnetycznego
- Zjawisko precesji
- Zależność częstotliwości precesji od natężenia pola magnetycznego

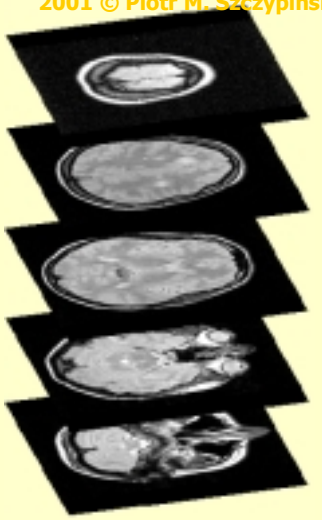
http://members.lycos.nl/mri/Nieuw/framesinhoudeng.htm
http://brainmapping.loni.ucla.edu/BMD_HTML/SharedCode/TINS/FMRI-TINS.html

Zjawisko rezonansu magnetycznego (2)

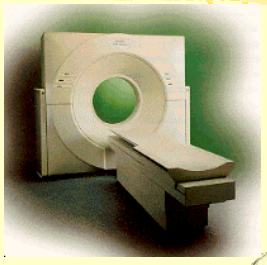


- Tworzenie gradientu pola magnetycznego za pomocą pary cewek
- Zastosowania trzech par cewek wytwarzających gradient pola magnetycznego, działających w kierunkach x, y i z, umożliwia wytworzenie pola o zadanej wielkości w jednym punkcie obrazowanej objętości

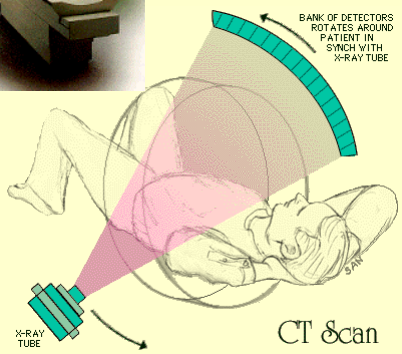
MRI - Obrazowanie rezonansu magnetycznego



Tomografia komputerowa (rentgenowska)

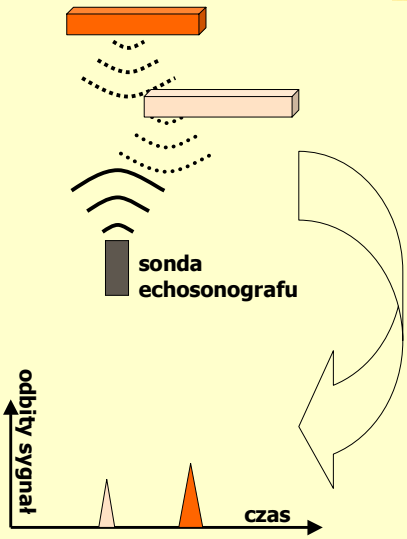


- Wykonanie prześwietleń pod wieloma kątami
- rekonstrukcja przekroju na podstawie wykonanych prześwietleń

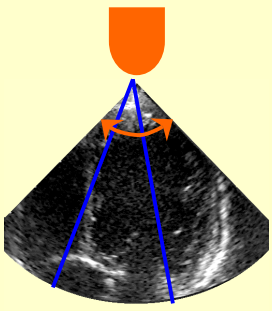


http://radiology.bidmc.harvard.edu/kinds_of_exams/CT/CT.html

Echosonografia (ultrasonografia)

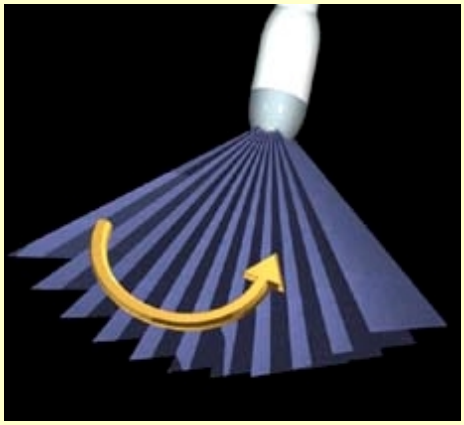


- Zjawisko odbicia fali dźwiękowej
- obrazowanie na podstawie odbitej fali dźwiękowej



Echosonografia 3D

Obrót sondy

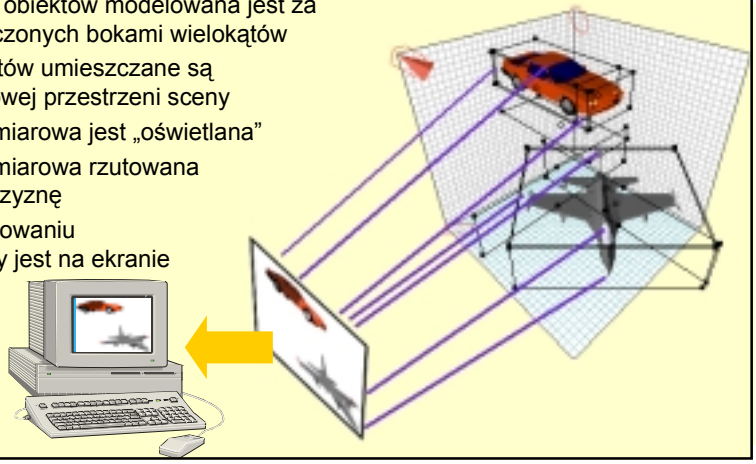


Zmiana kąta nachylenia sondy

http://www.gyneweb.fr/sources/echographie/bbenoit/Echo3D.html

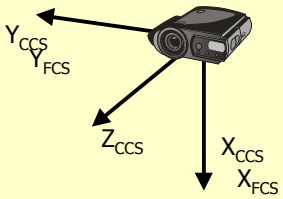
Trójwymiarowe obrazy wektorowe

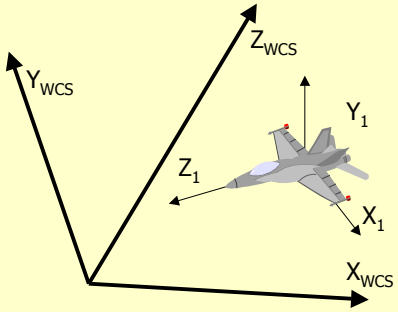
- W wirtualnej przestrzeni trójwymiarowej modelowane są obiekty (bryły)
- Obiekty definiowane są poprzez określenie kształtu ich powierzchni
- Powierzchnia obiektów modelowana jest za pomocą połączonych bokami wielokątów
- Modele obiektów umieszczane są w trójwymiarowej przestrzeni sceny
- Scena trójwymiarowa jest „oświetlana”
- Scena trójwymiarowa rzutowana jest na płaszczyznę
- Obraz po rzutowaniu przedstawiany jest na ekranie



Układy współrzędnych

- Układy współrzędnych obiektów
- Układ współrzędnych sceny (WCS)
- Układ współrzędnych kamery (CCS)
- Układ współrzędnych filmu (FCS)

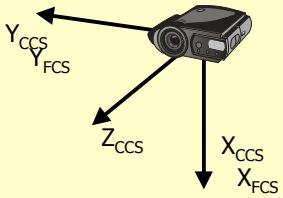




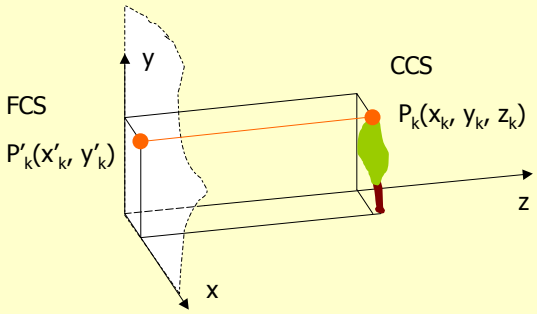
http://server.eletel.p.lodz.pl/~makowski/graph.html

Rzutowanie

- Załóżmy, że wszystkie współrzędne są określone w układzie współrzędnych kamery (zostały przeliczone).
- Zadaniem jest rzutowanie sceny trójwymiarowej na płaszczyznę obrazu.
  Rzutowanie - transformacja przestrzeni $\boldsymbol{R^n}$ *w przestrzeń* $\boldsymbol{R^{n-1}}$



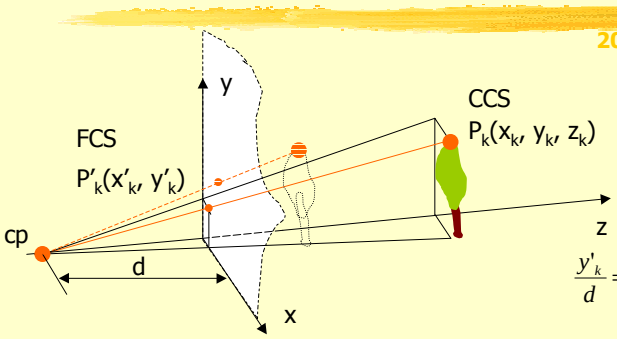
Rzutowanie równoległe



$x'_k = x_k$

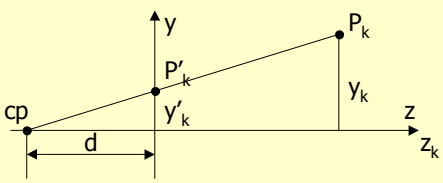
$y'_k = y_k$

Rzutowanie perspektywiczne



$$\frac{y'_k}{d} = \frac{y_k}{d + y_k} \qquad \frac{x'_k}{d} = \frac{x_k}{d + z_k}$$

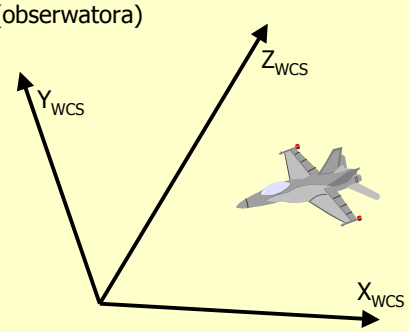
$$y'_k = \frac{y_k}{1 + \frac{z_k}{d}} \qquad x'_k = \frac{x_k}{1 + \frac{z_k}{d}}$$

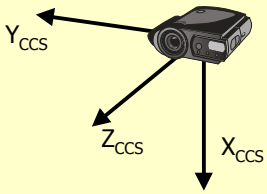


- Gdy d dąży do nieskończoności to rzutowanie perspektywiczne staje się rzutowaniem równoległym

Zmiana układu współrzędnych

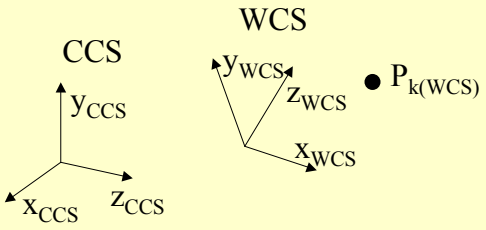
- Załóżmy, że wszystkie współrzędne są określone w układzie współrzędnych sceny.
- Zadaniem jest zmiana współrzędnych obiektu z układu współrzędnych sceny do układu współrzędnych kamery (obserwatora)



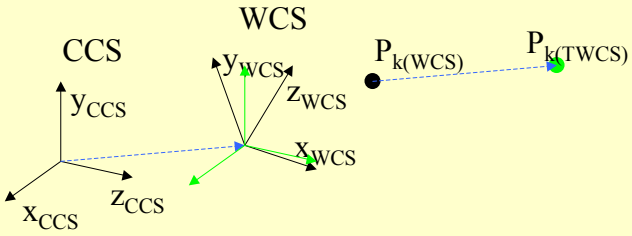


- Transformację układu współrzędnych wykonamy w dwóch etapach:
  - przesunięcia (translacji),
  - obrotu (rotacji).

Przesunięcie (translacja)

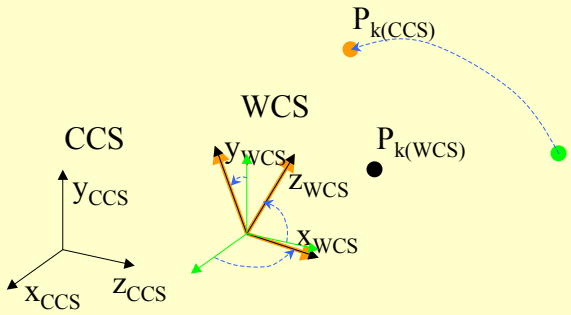


$$\begin{bmatrix} x_{k(TWCS)} \\ y_{k(TWCS)} \\ z_{k(TWCS)} \end{bmatrix} = \begin{bmatrix} x_{k(WCS)} \\ y_{k(WCS)} \\ z_{k(WCS)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$



Obrót (rotacja)

$$\begin{bmatrix} x_{k(CCS)} \\ y_{k(CCS)} \\ z_{k(CCS)} \end{bmatrix} = \begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{bmatrix} \left(\begin{bmatrix} x_{k(WCS)} \\ y_{k(WCS)} \\ z_{k(WCS)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right)$$



Właściwości macierzy C

$$\begin{bmatrix} x_{k(CCS)} \\ y_{k(CCS)} \\ z_{k(CCS)} \end{bmatrix} = \underbrace{\begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{bmatrix}}_{\mathbf{C}} \left(\begin{bmatrix} x_{k(WCS)} \\ y_{k(WCS)} \\ z_{k(WCS)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right)$$

- Macierz powinna być ortonormalna:
  - zachowanie ortogonalności układu współrzędnych (kierunki osi prostopadłe)
  - zachowanie proporcjonalnych rozmiarów we wszystkich kierunkach

$$\mathbf{C^T C = 1}$$

$$\mathbf{C^T C C^{-1} = 1 C^{-1}}$$

$$\mathbf{C^T = C^{-1}}$$

http://server.eletel.p.lodz.pl/~makowski/graph.html

Przykład 1
(założenia)

Niech:

- Kamera znajduje się w punkcie $[a_x, a_y, a_z]^T$ w układzie współrzędnych sceny,
- Kamera jest skierowana zgodnie z wektorem $[b_x, b_y, b_z]^T$ w układzie współrzędnych sceny,
- Kamera nie jest skręcona (podstawa kamery równoległa do płaszczyzny OYZ układu współrzędnych sceny)

Przykład 1
(wektor przesunięcia)

- Wektor przesunięcia jest równy zanegowanemu wektorowi położenia kamery w układzie współrzędnych sceny.

$$[t_x, t_y, t_z]^T = -[a_x, a_y, a_z]^T$$

Przykład 1
(macierz rotacji - wiersz 3)

- Wektor $[c_{zx}, c_{zy}, c_{zz}]^T$ odpowiada za przekształcenie osi Z układu współrzędnych, wzdłuż której skierowana jest kamera.
- Wektor $[c_{zx}, c_{zy}, c_{zz}]^T$ jest równy znormalizowanemu wektorowi $[b_x, b_y, b_z]^T$.

$$\begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{bmatrix}$$

$$[c_{zx}, c_{zy}, c_{zz}]^T = [b_x, b_y, b_z]^T / ||[b_x, b_y, b_z]^T||$$

Przykład 1
(macierz rotacji - wiersz 1)

- Wektor $[c_{xx}, c_{xy}, c_{xz}]^T$ odpowiada za to czy kamera jest skręcona.
- Ponieważ kamera nie jest skręcona to wektor $[c_{zx}, c_{zy}, c_{zz}]^T$ powinien być prostopadły do obliczonego wektora $[c_{zx}, c_{zy}, c_{zz}]^T$ i równoległy do osi OYZ układu współrzędnych sceny
- Wektor $[c_{xx}, c_{xy}, c_{xz}]^T$ powinien być znormalizowany

$$\begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{bmatrix}$$

$$[c_{xx}, c_{xy}, c_{xz}]^T = [c_{zz}, 0, -c_{zx}]^T / ||[c_{zz}, 0, -c_{zx}]^T||$$

Przykład 1
(macierz rotacji - wiersz 2)

- Wektor $[c_{yx}, c_{yy}, c_{yz}]^T$ powinien być prostopadły do obliczonych już wektorów $[c_{xx}, c_{xy}, c_{xz}]^T$ i $[c_{zx}, c_{zy}, c_{zz}]^T$
- Można go więc obliczyć jako iloczyn wektorowy wektorów $[c_{xx}, c_{xy}, c_{xz}]^T$ i $[c_{zx}, c_{zy}, c_{zz}]^T$

$$\begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{bmatrix}$$

$$[c_{yx}, c_{yy}, c_{yz}]^T = [c_{xx}, c_{xy}, c_{xz}]^T \times [c_{zx}, c_{zy}, c_{zz}]^T$$

http://server.eletel.p.lodz.pl/~makowski/graph.html

Przykład 2
(założenia)

Niech:

- Kamera znajduje się w punkcie $[a_x, a_y, a_z]^T$ w układzie współrzędnych sceny,
- Kamera jest skierowana względem układu współrzędnych sceny
  o kąt α (Yaw) w lewo,
  o kąt β (Pitch) do góry
  i jest skręcona o kąt γ (Roll).

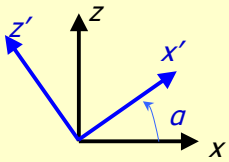
Przykład 2
(wektor przesunięcia)

- Wektor przesunięcia jest równy zanegowanemu wektorowi położenia kamery w układzie współrzędnych sceny.

$$[t_x, t_y, t_z]^T = -[a_x, a_y, a_z]^T$$

Przykład 2
(obrót o kąt *α*)

- Kamera jest obrócona o kąt *α* w osi pionowej y układu współrzędnych.
- Transformacji ze względu na kąt *α* podlegają współrzędne x i z układu współrzędnych.

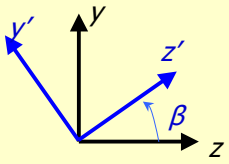


$$\begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}$$

$$x' = x\cos\alpha + z\sin\alpha$$
$$z' = z\cos\alpha - x\sin\alpha$$

Przykład 2
(obrót o kąt β)

- Kamera jest podniesiona o kąt β w osi x układu współrzędnych.
- Transformacji ze względu na kąt β podlegają współrzędne y i z.

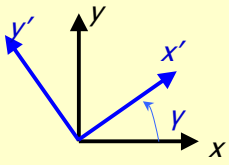


$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix}$$

$$z' = z\cos\beta + y\sin\beta$$
$$y' = y\cos\beta - z\sin\beta$$

Przykład 2
(obrót o kąt γ)

- Kamera jest skręcona o kąt γ w osi z układu współrzędnych.
- Transformacji ze względu na kąt γ podlegają współrzędne x i y.



$$\begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x' = x\cos\gamma + y\sin\gamma$$
$$y' = y\cos\gamma - x\sin\gamma$$

Przykład 2
(macierz rotacji)

$$\begin{bmatrix} x_{k(CCS)} \\ y_{k(CCS)} \\ z_{k(CCS)} \end{bmatrix} = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \left(\begin{bmatrix} x_{k(WCS)} \\ y_{k(WCS)} \\ z_{k(WCS)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right)$$

$$\begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{bmatrix} = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ \sin\alpha\sin\beta & \cos\beta & -\cos\alpha\sin\beta \\ -\sin\alpha\cos\beta & \sin\beta & \cos\alpha\cos\beta \end{bmatrix} =$$

$$= \begin{bmatrix} \cos\alpha\cos\gamma + \sin\alpha\sin\beta\sin\gamma & \cos\beta\sin\gamma & \sin\alpha\cos\gamma - \cos\alpha\sin\beta\sin\gamma \\ -\cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \cos\beta\cos\gamma & -\sin\alpha\sin\gamma 0 - \cos\alpha\sin\beta\cos\gamma \\ -\sin\alpha\cos\beta & \sin\beta & \cos\alpha\cos\beta \end{bmatrix}$$

Współrzędne jednorodne

Transformacja układu współrzędnych jest złożeniem:

- przesunięcia (dodawanie wektorów)
- rotacji (mnożenie macierzy przez wektor).

Możliwe jest sprowadzenie dwóch działań do jednego działania - mnożenia macierzy. Konieczne jest dodanie dodatkowej, pomocniczej współrzędnej.

$$\begin{bmatrix} x_{k(CCS)} \\ y_{k(CCS)} \\ z_{k(CCS)} \end{bmatrix} = \begin{bmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{bmatrix} \left(\begin{bmatrix} x_{k(WCS)} \\ y_{k(WCS)} \\ z_{k(WCS)} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right)$$

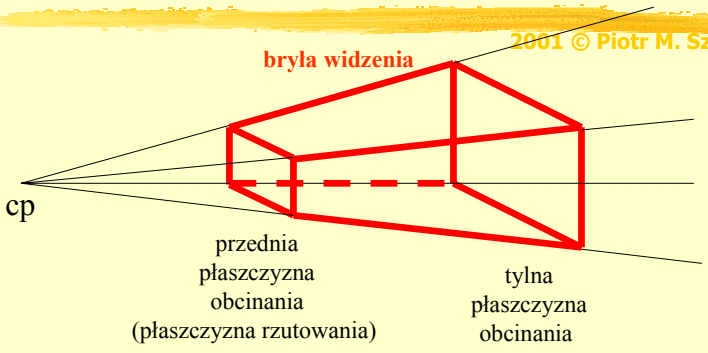
$$\begin{bmatrix} x_{k(CCS)} \\ y_{k(CCS)} \\ z_{k(CCS)} \\ 1 \end{bmatrix} = \begin{bmatrix} c_{xx} & c_{xy} & c_{xz} & 0 \\ c_{yx} & c_{yy} & c_{yz} & 0 \\ c_{zx} & c_{zy} & c_{zz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k(WCS)} \\ y_{k(WCS)} \\ z_{k(WCS)} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_{k(CCS)} \\ y_{k(CCS)} \\ z_{k(CCS)} \\ 1 \end{bmatrix} = \begin{bmatrix} c_{xx} & c_{xy} & c_{xz} & c_{xx}t_x + c_{xy}t_y + c_{xz}t_z \\ c_{yx} & c_{yy} & c_{yz} & c_{yx}t_x + c_{yy}t_y + c_{yz}t_z \\ c_{zx} & c_{zy} & c_{zz} & c_{zx}t_x + c_{zy}t_y + c_{zz}t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k(WCS)} \\ y_{k(WCS)} \\ z_{k(WCS)} \\ 1 \end{bmatrix}$$

Widoczne elementy sceny

- Bryła widzenia
- Eliminowanie niewidocznych ścianek
- Przesłanianie

Bryła widzenia

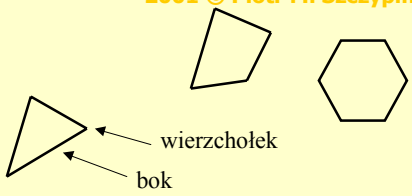


Jeśli ekran ma rozmiar W x H, a tylna płaszczyzna obcinania jest w odległości D od płaszczyzny rzutowania to na ekranie będą widoczne jedynie punkty P_k, których współrzędne w układzie CCS są:

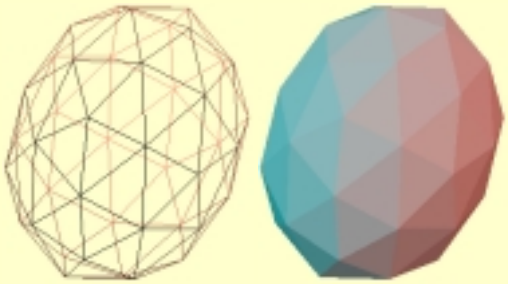
$$0 \le z_k \le D \qquad -\frac{W}{2}\left(1+\frac{z_k}{d}\right) \le x_k \le \frac{W}{2}\left(1+\frac{z_k}{d}\right) \qquad -\frac{H}{2}\left(1+\frac{z_k}{d}\right) \le y_k \le \frac{H}{2}\left(1+\frac{z_k}{d}\right)$$

Modelowanie obiektów

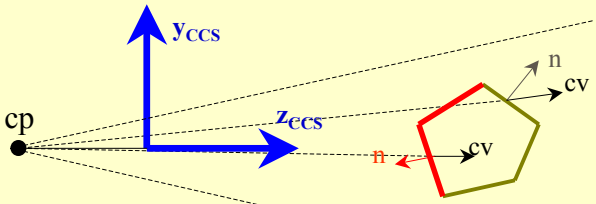
- Obiekty definiowane są poprzez określenie kształtu ich powierzchni
- Powierzchnia obiektów modelowana jest za pomocą połączonych bokami wielokątów koplanarnych



- Powierzchnie są zazwyczaj zamknięte (zamykają pewną objętość)



Eliminowanie niewidocznych ścianek obiektów



cv - wektor o kierunku i zwrocie zgodnym z wektorem łączącym punkt środka rzutowania (cp) z punktem ścianki

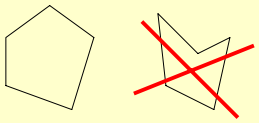
n - wektor normalny do powierzchni obiektu (ścianki)

Jeśli **cv•n > 0** to ścianka nie jest widoczna

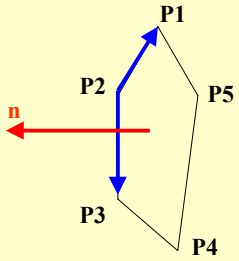
Wyznaczanie wektora normalnego

Jeśli założymy, że:

- wielobok ścianki jest koplanarny i wypukły,



- punkty wieloboku zapisane w kolejności zgodnej z ruchem wskazówek zegara (jeśli patrzeć od wewnątrz obiektu)

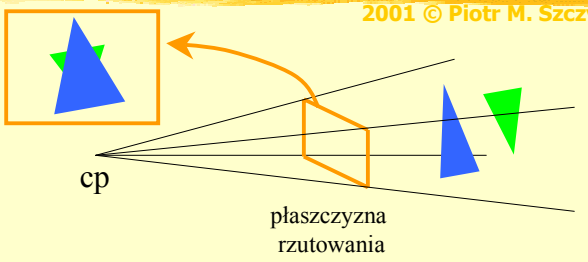


To:

$n = (\overrightarrow{p_2 p_3} \times \overrightarrow{p_2 p_1}) / || \overrightarrow{p_2 p_3} \times \overrightarrow{p_2 p_1} ||$

Przesłanianie
algorytm bufora głębokości

Obiekt bliższy obserwatorowi przesłania obiekt, który znajduje się dalej.



- **Bufor głębokości** (z-bufor) jest macierzą o rozmiarach obrazu, przechowującą odległości obiektu od obserwatora. Każdemu pikselowi tworzonego obrazu odpowiada wartość odległości w buforze głębokości.
- Podczas rysowania obiektu na powierzchni ekranu sprawdza się czy w danym pikselu już czegoś nie narysowano. Jeżeli tak, to w buforze głębokości sprawdza się czy to co narysowano wcześniej nie znajduje się bliżej niż to co ma być rysowane. Jeżeli tak, to rysowanie dla danego piksela nie jest wykonywane.

Oświetlenie

- Modele oświetlenia
- Modele odbić
- Interakcje obiektów:
  - cienie,
  - oświetlenie pośrednie.

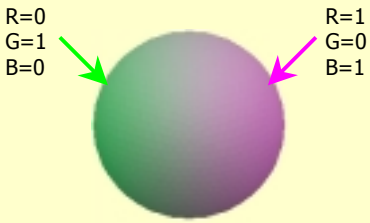
Modele oświetlenie

- Oświetlenie otaczające (ang. *ambient ilumination*)

- Oświetlenie kierunkowe

- Punktowe źródła światła

- Kolor światła (model RGB)

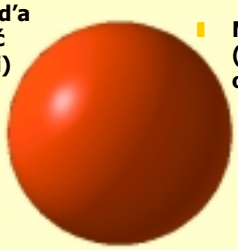


Modele odbić

- Oświetlenie światłem otaczającym

- Odbicie płaskie (można wyróżnić poszczególne ścianki)

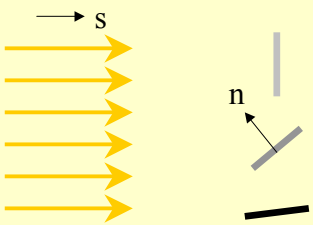
- Model odbić Gouraud'a (złagodzenie przejść pomiędzy ściankami)



- Model odbić Phong'a (możliwość obliczania odbłysków światła)

Modele odbić
odbicie płaskie (stałe)

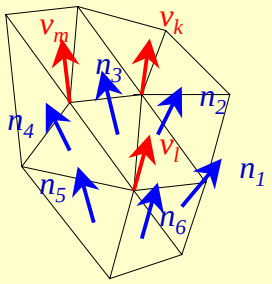
- Oświetlenie kierunkowe
- Cała powierzchnia ścianki (wielokąta) rozprasza promienie równomiernie we wszystkich kierunkach



- Jeśli s jest znormalizowanym wektorem kierunku oświetlenia, n jest wektorem normalnym do oświetlanej powierzchni, $L_R\,L_G\,L_B$ są składowymi barwy światła a $J_R\,J_G\,J_B$ są składowymi barwy powierzchni to wynikowe składowe koloru powierzchni po rzutowaniu można zapisać:

$$- \begin{bmatrix} L_R J_R \\ L_G J_G \\ L_B J_B \end{bmatrix} n \bullet s$$

Modele odbić
odbicie (cieniowanie) Gouraud'a

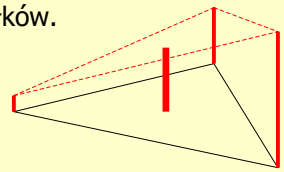


- Powierzchnia modelu zdefiniowana jako siatka trójkątów
- Oświetlenie jest kierunkowe lub punktowe

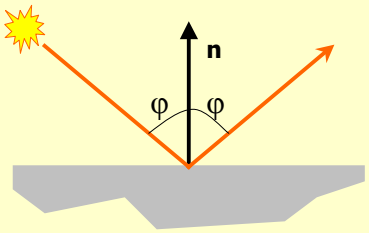
- Wektor normalny w wierzchołku jest średnią arytmetyczną wektorów normalnych ścianek, do których wierzchołek przynależy:

$$v_l = (n_1 + n_2 + \ldots + n_i)/i$$

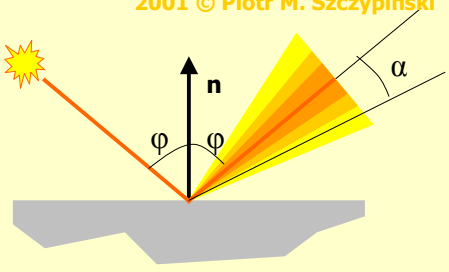
- Barwa wierzchołków po rzutowaniu jest obliczana jak w modelu odbić płaskich.
- Barwa ścianki jest interpolowana na podstawie obliczonej barwy wierzchołków.



Modele odbić
odbłyski w modelu Phong'a



- Odbicie w idealnym zwierciadle



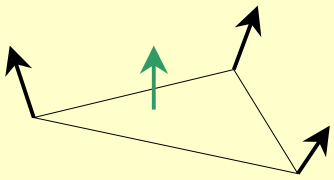
- Odbicie w modelu Phong'a (model nieidealnego zwierciadła)

Intensywność odbłysku dana jest równaniem:

$$I = \cos^n\alpha$$

barwa odbłysku zależy od źródła światła, nie od barwy powierzchni

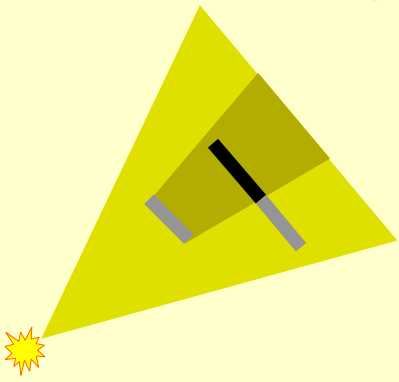
Modele odbić
model Phong'a



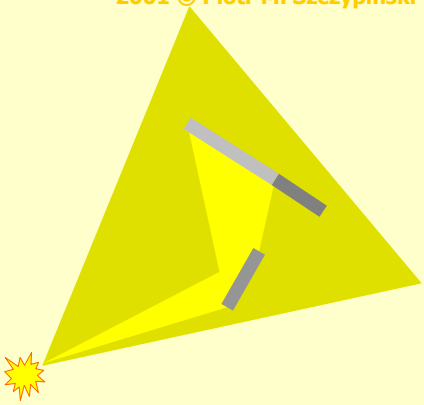
- Obliczanie wektorów normalnych w wierzchołkach
- Obliczanie wektorów normalnych w punktach ścianki jako interpolacji wektorów normalnych w wierzchołkach
- Obliczanie odbić i odbłysków światła w poszczególnych punktach ścianki

- Powierzchnia modelu zdefiniowana jako siatka trójkątów
- Oświetlenie jest kierunkowe lub punktowe

Interakcje obiektów



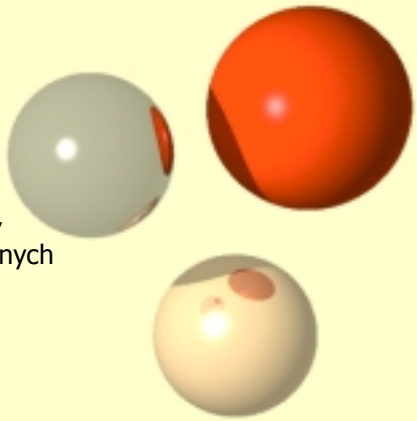
- **Cienie**



- **Oświetlenie pośrednie**

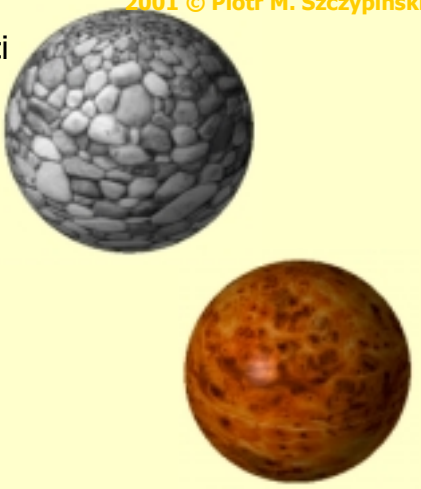
Metoda śledzenia promieni

- Wsteczna analiza promienia światła od punktu środka rzutowania do źródła (źródeł) światła
- Zalety:
  - rozwiązanie problemów cieni,
  - rozwiązanie problemu oświetlenia pośredniego,
  - usuwanie niewidocznych ścianek,
  - analiza przesłaniania się obiektów,
  - możliwość analizy odbić wewnętrznych w obiektach przeźroczystych.
- Wady:
  - duży koszt obliczeniowy.



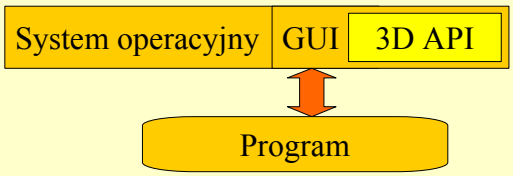
Tekstury
Powierzchnie niejednorodne

- Powierzchnie w rzeczywistości mogą odbijać światło w sposób niejednorodny,
- Do definiowania takiej powierzchni wykorzystuje się tekstury (wzory, desenie),
- Tekstury są zazwyczaj definiowane w formie bitmap „rozpinanych” na powierzchni obiektów.



Programowanie grafiki 3D

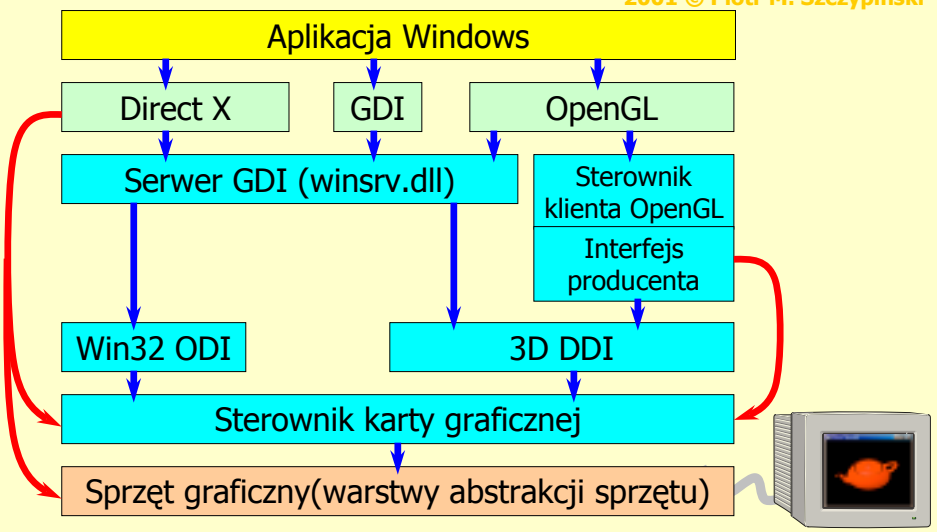
- Korzystanie z bibliotek obsługi grafiki 3D (3D API):
  - Wykorzystanie gotowych, sprawdzonych i zoptymalizowanych metod,
  - możliwość skorzystania z możliwości sprzętu graficznego (akceleracji 3D)



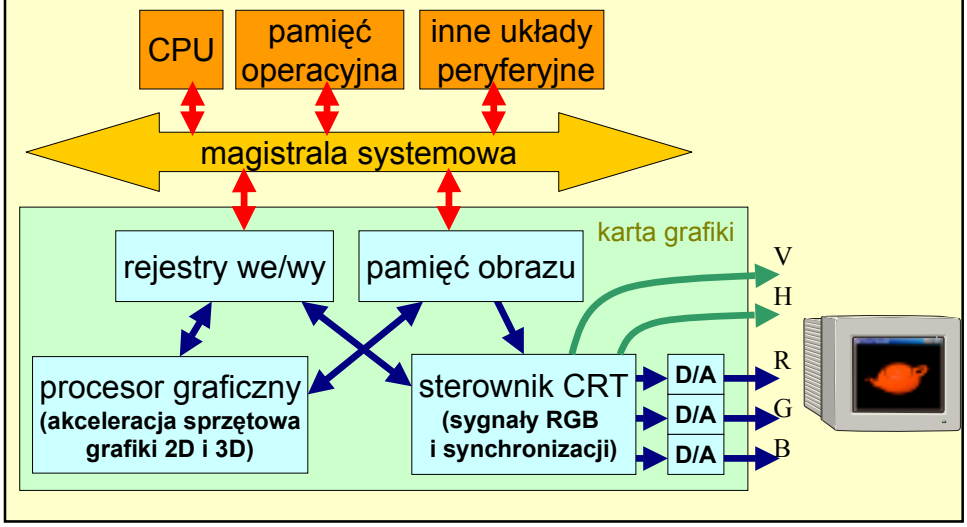
Interfejsy programowe grafiki 3D

- OpenGL (*GL - Graphics Language*)
  - standard i algorytmy opracowane przez Silicon Graphics dla stacji IRIS,
  - specyfikacja 1.0 OpenGL w 1992 roku (opracowano pakiety dla wielu platform komputerowych),
  - Microsoft dołączył OpenGL do systemów operacyjnych Windows NT 4.0 i Windows 95,
- Direct 3D (część Direct X)
  - celem pakietu Direct X jest ominięcie ograniczeń GUI systemu Windows i umożliwienie korzystania z zaawansowanych możliwości kart dźwiękowych i graficznych (np. w celu uruchamiania gier),
  - najpierw był Reality Lab firmy RenderMorphing (1994),
  - Microsoft kupuje RenderMorphing (1995) i tworzy Direct3D,
  - pakiet wciąż jest udoskonalany,
  - Direct X dostępne jedynie dla systemów operacyjnych Windows.

Schemat działania akceleracji sprzętowej w Windows

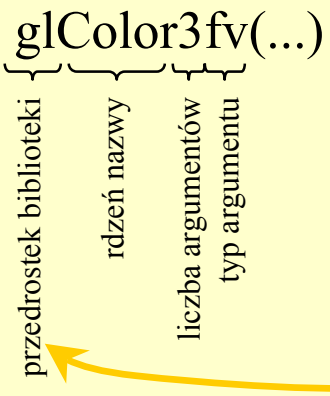


Podstawy działania kart graficznych



OpenGL

- Konwencja nazw funkcji OpenGL API:



- Składniki OpenGL API - biblioteki:

Główna **Opengl32.dll**
Narzędziowa **Glu32.dll**
Pomocnicza **glaux.lib**

- Pliki nagłówkowe C:

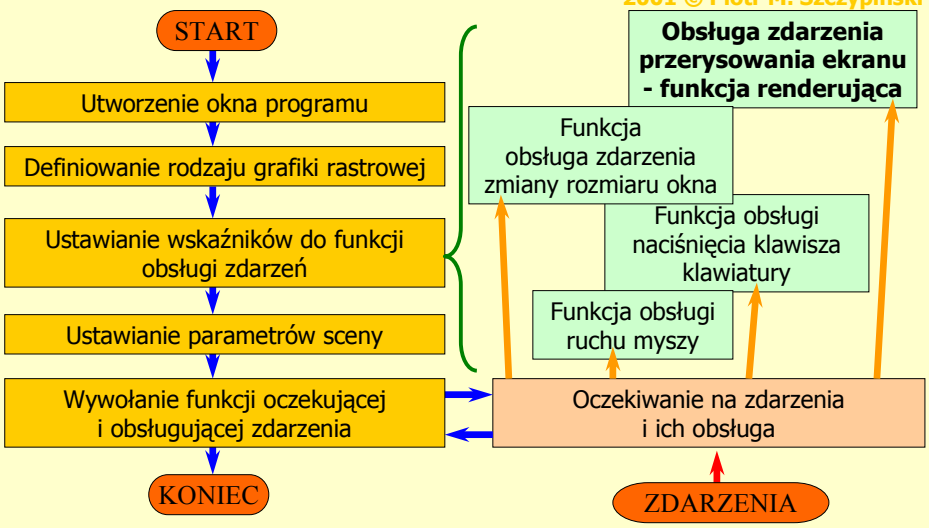
gl.h
glu.h
aux.h

Przykłady działania funkcji OpenGL

Tools for teaching about OpenGL
Autor: Nate Robins
nate@pobox.com, http://www.pobox.com/~nate
http://www.xmission.com/~nate/tutors.html

Transformacje | **Oświetlenie** | **Mgła**
Projekcja | **Współrzędne** | **Funkcje 2D**

Program OpenGL w Windows



Przykład programu OpenGL (1)


```
/*************************************************************************/
/*                                                                       */
/* MODULE: Gk_gl.c                                                       */
/*                                                                       */
/* FUNCTIONALITY:                                                        */
/*   Program przykladowy na zajecia Grafika Komputerowa.                 */
/*   Programowanie OpenGL                                                */
/*                                                                       */
/* Copyright (C) 2001 by Piotr M. Szczypinski                            */
/*                                                                       */
/*************************************************************************/

#include <windows.h>            // Window defines
#include <gl\gl.h>              // OpenGL
#include <gl\glu.h>             // GLU library
#include <gl\glaux.h>           // Biblioteka AUX
#include <math.h>               // Funkcje matematyczne

// Katy rzutowania obiektu
static GLfloat xRot = 0.0f;
static GLfloat yRot = 0.0f;
```

Przykład programu OpenGL (2)


```
// Glowna funkcja programu w Windows
int WINAPI WinMain(
    HINSTANCE hInstance,
    HINSTANCE hPrevInstance,
    LPSTR lpCmdLine,
    int nCmdShow
   )
{
// Utworzenie okna OpenGL
  auxInitDisplayMode(AUX_DOUBLE | AUX_RGBA);
  auxInitPosition(100,100,250,250);
  auxInitWindow("Okno OpenGL");

// Wskazanie funkcji wywoływanej w momencie zmiany rozmiarów okna
  auxReshapeFunc(ZmianaRozmiaruOkna);

// Wskazanie funkcji obslugi klawiszy
  auxKeyFunc(AUX_LEFT, StrzalkaLewo);
  auxKeyFunc(AUX_RIGHT, StrzalkaPrawo);
  auxKeyFunc(AUX_DOWN, StrzalkaDol);
  auxKeyFunc(AUX_UP, StrzalkaGora);

// Wywolanie funkcji ustawiajacej scene (tlo, swiatla, rodzaje odbic)
  UstawScene();

// Uruchomienie petli obslugi zdarzan (komunikatow)
  auxMainLoop(RenderujScene);
}
```

Przykład programu OpenGL (3)


```
// Funkcja ustawiajaca scene (tlo, swiatla, rodzaje odbic)
void UstawScene()
{
// Wartosci i wspolrzedne zrodel swiatla
  GLfloat  ambientLight[] = {0.4f, 0.4f, 0.4f, 1.0f };
  GLfloat  diffuseLight[] = {0.7f, 0.7f, 0.7f, 1.0f };
  GLfloat  specular[] = { 0.9f, 0.9f, 0.9f, 1.0f};
  GLfloat  lightPos[] = { -500.0f, 2000.0f, 2000.0f, 1.0f };
  GLfloat  specref[] =  { 0.6f, 0.6f, 0.6f, 1.0f };

  glEnable(GL_DEPTH_TEST); // Wlaczenie metdy bufora glebokosci
  glEnable(GL_CULL_FACE); // Wlaczenie metody eliminowania niewidocznych scianek
  glEnable(GL_LIGHTING); // Wlaczenie metody obliczania oswietlenia
  glEnable(GL_NORMALIZE); // Ustawienie automatycznego obliczania normalnych do powierzchni
  glEnable(GL_AUTO_NORMAL);

// Przygotowanie i wlaczenie swiatla GL_LIGHT0
  glLightModelfv(GL_LIGHT_MODEL_AMBIENT,ambientLight);
  glLightfv(GL_LIGHT0,GL_AMBIENT,ambientLight);
  glLightfv(GL_LIGHT0,GL_DIFFUSE,diffuseLight);
  glLightfv(GL_LIGHT0,GL_SPECULAR,specular);
  glLightfv(GL_LIGHT0,GL_POSITION,lightPos);
  glEnable(GL_LIGHT0);

// Ustawienie wlaciwosci odbijania swiatla dla materialu
  glEnable(GL_COLOR_MATERIAL);
  glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
  glMaterialfv(GL_FRONT, GL_SPECULAR,specref);
  glMateriali(GL_FRONT,GL_SHININESS,64);

  glClearColor(0.0f, 0.0f, 0.0f, 1.0f ); // Ustawienie koloru tla
}
```

Przykład programu OpenGL (4)


```
// Zmiana bryly widzenia i widoku.
// Wywolywane w momencie zmiany wymiaru okna
void CALLBACK ZmianaRozmiaruOkna(GLsizei w, GLsizei h)
{
  GLfloat nRange = 100.0f;

// Zabezpieczenie przed dzieleniem przez 0
  if(h == 0) h = 1;

// Ustawienie widoku na rozmiary okna
  glViewport(0, 0, w, h);

// Wyzerowanie stosu macierzy rzutowania
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();

// Ustanowienie bryly obcinania (lewa, prawa, dolna, górna, blizsza, dalsza)
  if (w <= h)
    glOrtho (-nRange, nRange, -nRange*h/w, nRange*h/w, -nRange*2.0f, nRange*2.0f);
  else
    glOrtho (-nRange*w/h, nRange*w/h, -nRange, nRange, -nRange*2.0f, nRange*2.0f);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
}
```

```
// Funkcje wywolywane po nacisnieciu klawiszy
void CALLBACK StrzalkaLewo(void)
{
  yRot -= 5.0f;
  RenderujScene();
}
```

Przykład programu OpenGL (5)


```
// Funkcja renderowania - rysowania sceny
void CALLBACK RenderujScene(void)
{
// Czysci okno
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

// Zapisuje macierz transformacji
  glMatrixMode(GL_MODELVIEW);
  glPushMatrix();
  glRotatef(xRot, 1.0f, 0.0f, 0.0f);
  glRotatef(yRot, 0.0f, 1.0f, 0.0f);

// Definiowanie ksztaltu obiektu
// Obiekt sklada sie z trojkatnych scianek ze zdefiniowanymi normalnymi
  glFrontFace(GL_CW);

  glBegin(GL_TRIANGLES);
    glColor3f(0.2f, 0.2f, 0.8f);
    glNormal3f(0.58f, 0.58f, 0.58f);
    glVertex3f(0.0f, 60.0f, 0.0f);
    glVertex3f(60.0f, 0.0f, 0.0f);
    glVertex3f(0.0f, 0.0f, 60.0f);
                ...
  glEnd();

// Wykorzystanie zdefiniowanego obiektu czajniczka
  glColor3f(0.9f, 0.0f, 0.0f);
  auxSolidTeapot(50.0f);

// Odczytuje macierz transformacji
  glPopMatrix();
  glFlush();

// Zamiana buforow obrazu
  auxSwapBuffers();
}
```

Bufor głębokości

Piramida

Czajnik

Podsumowanie

- Wiedza (wykład):
  - Modele koloru, ich pochodzenie i zastosowanie
  - Obrazy rastrowe
    (metody uzyskiwania, formaty zapisu, wizualizacja, zastosowania)
  - Obrazy wektorowe
    (metody uzyskiwania, formaty zapisu, wizualizacja, zastosowania)
  - Metapliki (wykorzystanie metaplików w komputerowych systemach graficznych)
  - Obrazy rastrowe w 3D (metody uzyskiwania, zastosowania)
  - Modelowanie 3D (układy współrzędnych, transformacje przestrzeni, rzutowanie, zagadnienia związane z modelowaniem oświetlenia)
- Umiejętności (laboratorium):
  - Programowanie grafiki 2D z wykorzystaniem GUI Windows
    (grafika rastrowa i wektorowa)
  - Programowanie procedur grafiki 3D
  - Podstawy korzystania z interfejsów programowych grafiki 3D