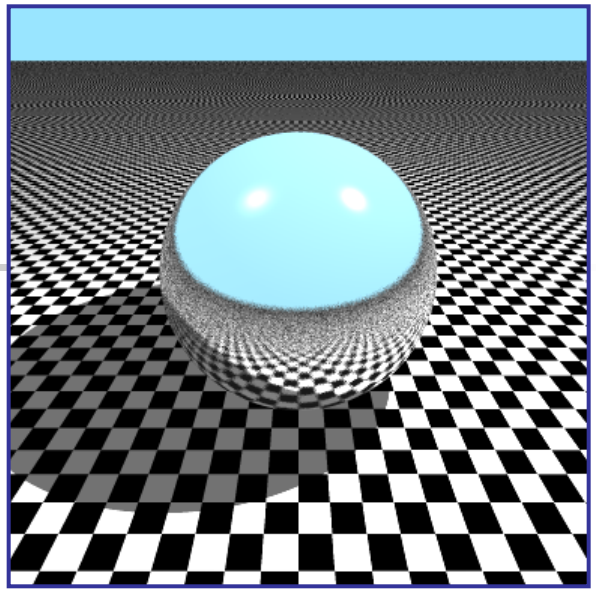
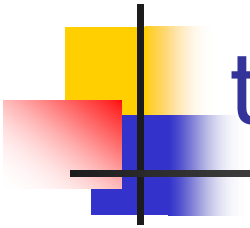
Wykład 6:



Tekstury
Potok graficzny w OpenGL

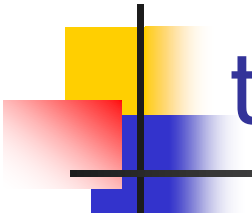


teksturowanie:

Teksturowanie to postępowanie polegające na nakładaniu na obiekty (wielokąty) obrazów (tzw. tekstur) (obrazów bitmapowych lub obrazów generowanych przy pomocy funkcji matematycznych), co pozwala zwiększyć stopień realizmu generowanych scen, dodając pewne cechy obiektom, które bez tego wyglądałyby nienaturalnie.

Mówiąc o teksturach będziemy posługiwali się pojęciem teksela, zamiast piksela (piksel odnosić się będzie do punktu w oknie monitora), natomiast teksel możemy rozumieć jako „piksel” tekstury.

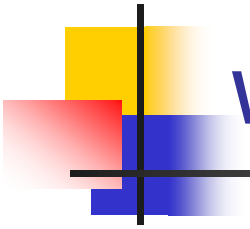
Mapa tekstury jest określona we własnym układzie współrzędnych.



tekstury w OpenGL:

Tekstury w OpenGL mogą być:
- jednowymiarowe – reprezentowane przez obrazy wymiaru n × 1 lub 1 × m tekseli
- **dwuwymiarowe** – reprezentowane przez obrazy wymiaru m × n tekseli
- trójwymiarowe – reprezentowane przez obrazy przestrzenne wymiaru m × n × k tekseli.

Przykładowo, dla tekstur dwuwymiarowych obraz m × n tekseli jest przekształcany do układu współrzędnych ciągłych s, t na obszar kwadratu [0, 1] × [0, 1]. Tekstury dwuwymiarowe są nakładane na powierzchnie wielokątów przez jednoznaczne przypisanie każdemu wierzchołkowi wielokąta pary współrzędnych (s, t). Ponieważ każda taka para oznacza punkt należący do płaszczyzny obrazu (leżący we wnętrzu któregoś teksela), uniezależniamy się od używanej rozdzielczości obrazu i podajemy miejsce na obrazie, które chcemy przypisać do danego wierzchołka.

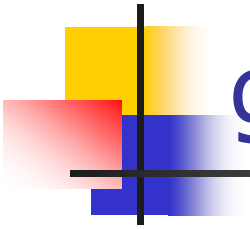


włączenie teksturowania:

Włączenie teksturowania wymaga wywołania funkcji
glEnable(GLenum target)
gdzie target – określa rodzaj tworzonej tekstury:
- GL_TEXTURE_1D - tekstura jednowymiarowa
- GL_TEXTURE_2D - tekstura dwuwymiarowa
- GL_TEXTURE_3D - tekstura trójwymiarowa

Wyłączenie teksturowania wymaga wywołania funkcji **glDisable** z tym samym parametrem.

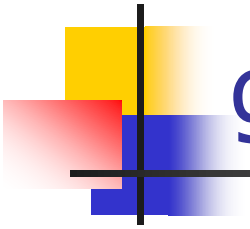
Przy jednoczesnym włączeniu więcej niż jednego rodzaju tekstury, aktywna będzie tekstura o największej ilości wymiarów,



generowanie tekstury:

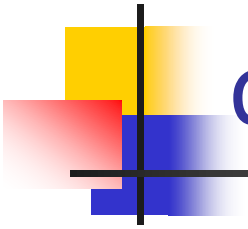
Funkcja:
void glGenTextures(GLsizei n, GLuint * textures)
gdzie n określa ilość generowanych tekstur, a textures określa poprzez wskazanie, numery unikatowych identyfikatorów tekstur, umożliwia generowanie nowych obiektów tekstur.

Funkcja:
void glDeleteTextures(GLsizei n, const GLuint * textures)
gdzie n określa ilość tekstur, a textures określa poprzez wskazanie, numery unikatowych identyfikatorów tekstur, usuwa podane obiekty tekstur celem zwolnienia wykorzystywanej przez nie pamięci.



generowanie tekstury:

Funkcja:
void glBindTexture(GLenum target, GLuint texture)
gdzie target wskazuje rodzaj tekstury (GL_TEXTURE_1D, GL_TEXTURE_2D lub GL_TEXTURE_3D), a texture zawiera identyfikator tekstury, określa podaną teksturę, jako aktualnie obowiązującą (wywołania wszelkich funkcji związanych z ładowaniem tekstury i ustawianiem jej parametrów dotyczą właśnie tego, aktualnego obiektu tekstury).



definiowanie tekstury:

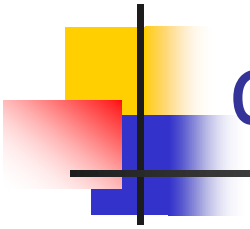
Funkcje:

void glTexImage1D(GLenum target, GLint level,
GLint internalformat, GLsizei width, GLint border,
GLenum format, GLenum type, const GLvoid * pixels)

void glTexImage2D(GLenum target, GLint level,
GLint internalformat, GLsizei width, GLsizei height, GLint border,
GLenum format, GLenum type, const GLvoid * pixels)

void glTexImage3D(GLenum target, GLint level,
GLint internalformat, GLsizei width, GLsizei height, GLsizei depth,
GLint border, GLenum format, GLenum type, const GLvoid * pixels)

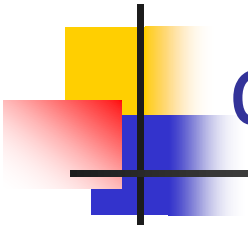
definiują dane graficzne jedno, dwu i trójwymiarowych tekstur.



definiowanie tekstury:

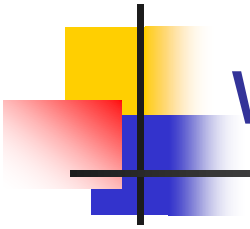
Parametry funkcji definiujących tekstury:

- **target** – określa rodzaj tworzonej tekstury: GL_TEXTURE_1D, GL_TEXTURE_2D lub GL_TEXTURE_3D
- **level** – określa poziom kolejnej tworzonej mipmapy, wartość 0 oznacza poziom podstawowy tekstury
- **internalformat** – określa wewnętrzny format danych tekstury, dla funkcji glTexImage1D i glTexImage2D dopuszczalna jest jedna z wartości: 1, 2, 3 lub 4 (odpowiada ilości składowych opisujących pojedynczy element tekstury - teksel)
- **width**, **height** i **depth** – określają szerokość, wysokość i głębokość tekstury wraz z obramowaniem (ilość dostępnych wymiarów zależy od rodzaju tekstury)



definiowanie tekstury:

- **border** – określa szerokość obramowania tekstury: wartość 0 oznacza brak obramowania, wartość 1 oznacza, że tekstura posiada obramowanie, a każdy z wymiarów tekstury jest powiększony o dwa dodatkowe teksele
- **format** – określa format danych tekstury i przyjmuje wartości: GL_COLOR_INDEX, GL_DEPTH_COMPONENT, GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_RGBA, GL_BGR, GL_BGRA, GL_LUMINANCE oraz GL_LUMINANCE_ALPHA
- **type** – określa format tekseli tekstury: GL_BYTE, GL_UNSIGNED, GL_UNSIGNED_BYTE, GL_SHORT, GL_UNSIGNED_SHORT, GL_INT, GL_UNSIGNED_INT, GL_FLOAT lub GL_BITMAP
- **pixels** – zawiera wskaźnik na dane tekstury

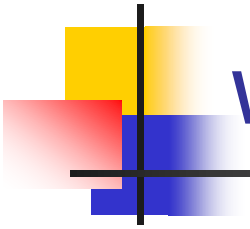


współrzędne tekstury:

Współrzędne tekstury określają, w jaki sposób tekstura nakładana jest na prymityw graficzny (wielokąt).
Ogólnie współrzędna tekstury ma cztery składowe [s, t, r, q], gdzie s określa szerokość, t wysokość, a r głębokość tekstury. Ostatnia współrzędna to współczynnik skalowania współrzędnych tekstur.

Zwykle wierzchołkom prymitywów przyporządkowywane są współrzędne z przedziału [0, 1].

Współrzędne tekstury definiują funkcje z grupy glTexCoord.
W każdej z czterech grup funkcji parametry mogą być określane liczbami całkowitymi lub zmiennoprzecinkowymi, a także mogą być przekazywane w postaci tablic.

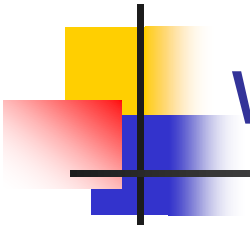


współrzędne tekstury:

Funkcje definiujące współrzędną s:

void glTexCoord1d(GLdouble s)
void glTexCoord1f(GLfloat s)
void glTexCoord1i(GLint s)
void glTexCoord1s(GLshort s)

void glTexCoord1dv(const GLdouble * v)
void glTexCoord1fv(const GLfloat * v)
void glTexCoord1iv(const GLint * v)
void glTexCoord1sv(const GLshort * v)

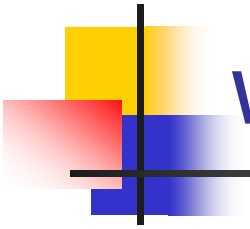


współrzędne tekstury:

Funkcje definiujące współrzędne s i t:

void glTexCoord2d(GLdouble s, GLdouble t)
void glTexCoord2f(GLfloat s, GLfloat t)
void glTexCoord2i(GLint s, GLint t)
void glTexCoord2s(GLshort s, GLshort t)

void glTexCoord2dv(const GLdouble * v)
void glTexCoord2fv(const GLfloat * v)
void glTexCoord2iv(const GLint * v)
void glTexCoord2sv(const GLshort * v)

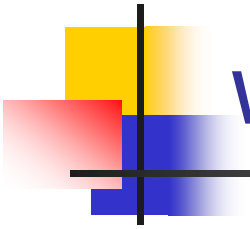


współrzędne tekstury:

Funkcje definiujące współrzędne s, t i r:

void glTexCoord3d(GLdouble s, GLdouble t, GLdouble r)
void glTexCoord3f(GLfloat s, GLfloat t,GLfloat r)
void glTexCoord3i(GLint s, GLint t, GLint r)
void glTexCoord3s(GLshort s, GLshort t, GLshort r)

void glTexCoord3dv(const GLdouble * v)
void glTexCoord3fv(const GLfloat * v)
void glTexCoord3iv(const GLint * v)
void glTexCoord3sv(const GLshort * v)

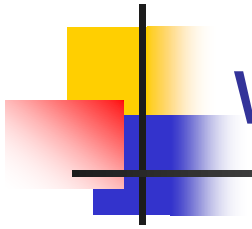


współrzędne tekstury:

Funkcje definiujące współrzędne s, t, r i q:

void glTexCoord4d(GLdouble s, GLdouble t, GLdouble r, GLdouble q)
void glTexCoord4f(GLfloat s, GLfloat t, GLfloat r, GLfloat q)
void glTexCoord4i(GLint s, GLint t, GLint r, GLint q)
void glTexCoord4s(GLshort s, GLshort t, GLshort r, GLshort q)

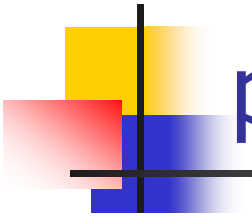
void glTexCoord4dv(const GLdouble * v)
void glTexCoord4fv(const GLfloat * v)
void glTexCoord4iv(const GLint * v)
void glTexCoord4sv(const GLshort * v)



współrzędne tekstury:

Przykład:

```
glBegin(GL_TRIANGLES);
  glTexCoord2f(0.1f, 0.0f);
  glVertex3f(0.0f, 0.0f, 0.0f);
  glTexCoord2f(1.0f, 0.0f);
  glVertex3f(2.0f, 0.0f, 0.0f);
  glTexCoord2f(0.1f, 1.0f);
  glVertex3f(0.0f, -2.0f, 0.0f);
glEnd();
```



parametry tekstury:

Funkcje:

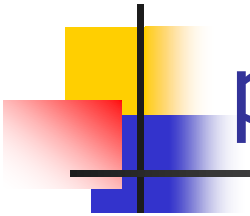
void glTexParameterf(GLenum target, GLenum pname, GLfloat param)

void glTexParameteri(GLenum target, GLenum pname, GLint param)

void glTexParameterfv(GLenum target, GLenum pname, const GLfloat * params)

void glTexParameteriv(GLenum target, GLenum pname, const GLint * params)

gdzie target to rodzaj tekstury (GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D), dla której definiowane są parametry, pname określa rodzaj parametru tekstury przekazywanego bezpośrednio jako wartość param lub jak tablica wartości params, określają parametry tekstur.



parametry tekstury:

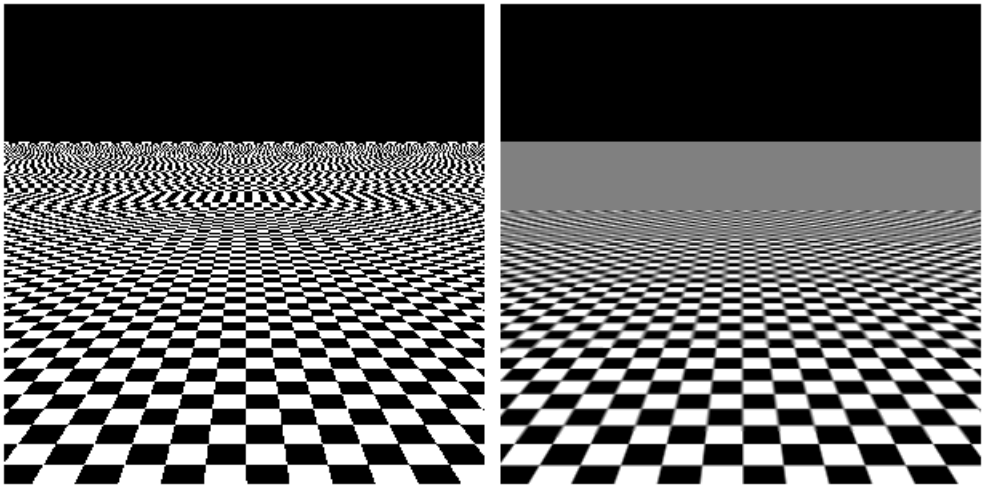
Możliwe wartości paramertu param:

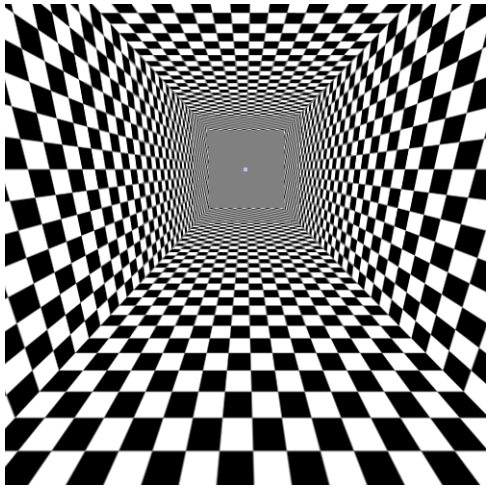
GL_TEXTURE_WRAP_S
GL_TEXTURE_WRAP_T
GL_TEXTURE_WRAP_R
GL_TEXTURE_MIN_FILTER
GL_TEXTURE_MAG_FILTER
GL_TEXTURE_BORDER_COLOR
GL_TEXTURE_PRIORITY
GL_TEXTURE_MIN_LOD
GL_TEXTURE_MAX_LOD
GL_TEXTURE_BASE_LEVEL
GL_TEXTURE_MAX_LEVEL
GL_TEXTURE_LOD_BIAS
GL_DEPTH_TEXTURE_MODE
GL_TEXTURE_COMPARE_MODE
GL_TEXTURE_COMPARE_FUNC
GL_GENERATE_MIPMAP

mipmapping:

Jeżeli tekstura jest mniejsza niż pokrywany obiekt, to jest ona skalowana lub powielana. Jeżeli jest większa, występuje konieczność przycinania, co powoduje zniekształcenia.

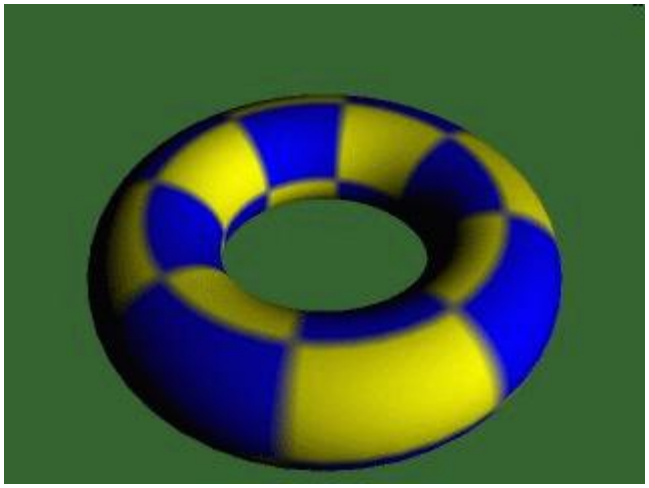
Mipmapping to technika polegająca na przygotowaniu i dobieraniu przeskalowanej wersji tekstury (dobór mipmapy jest uzależniony od liczby tekseli, które odpowiadają pikselowi). Przechowywane są tekstury (mipmapy) w kilku przeskalowanych wersjach (2 razy mniejsza, 4 razy mniejsza itd.).

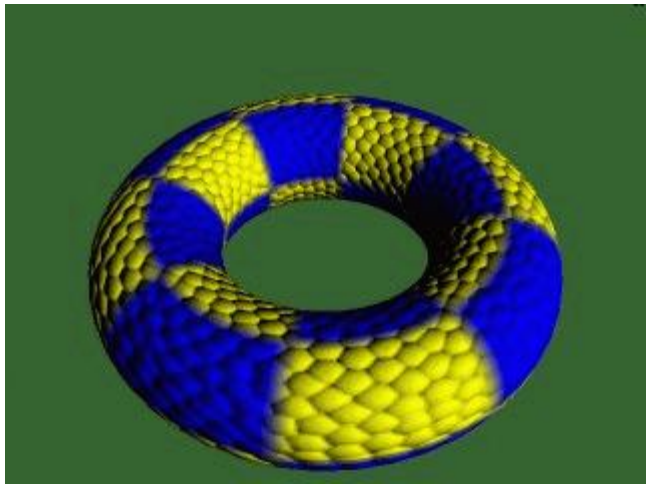


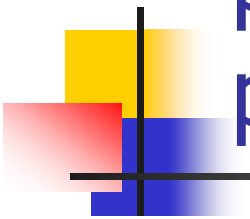


bump mapping:

Bump mapping (mapowanie nierówności) - technika symulująca niewielkie nierówności powierzchni, bez ingerencji w geometrię obiektu. Polega na użyciu tekstury, która nie jest bezpośrednio wyświetlana, ale powoduje lokalne zakłócenia wektora normalnego. Każdy model oświetlenia w jakimś stopniu używa wektora normalnego do obliczeń, czego rezultatem jest pojawienie się na obrazie złudzenia nierówności powierzchni.





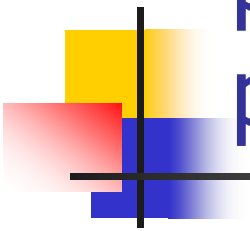


podsumowanie
potok graficzny w OpenGL:

Potok graficzny to ciąg operacji, w wyniku którego obiekty 3D zostają odwzorowane na obrazy 2D.

Typowy potok graficzny w OpenGL składa się z następujących transformacji kolejno pomiędzy układami współrzędnych następujących przestrzeni:

- przestrzeń **obiektu** (modelu) – tu definicja geometrii modelu jest zwykle prosta, np. zbiór współrzędnych wierzchołków tworzących prymityw
- przestrzeń **sceny** (świata 3D) – przestrzeń, w której rozmieszczone są już obiekty np. w wyniku przesunięć, obrotów bądź skalowania
- przestrzeń **obserwatora** – to przestrzeń, w której obserwator znajduje się w środku układu współrzędnych, a jego oś widzenia pokrywa się z osią OZ
- przestrzeń **obcinania** – to przestrzeń wyznaczona przez bryłę widzenia
- przestrzeń **współrzędnych znormalizowanych** – to przestrzeń, w której bryła widzenia staje się kostką $[-1,1]^3$ (normalizacja współrzędnych)
- przestrzeń **okna** – uzyskana w efekcie transformacji przestrzeni współrzędnych znormalizowanych do kostki [-1,1] x [-1,1] x [0,1]



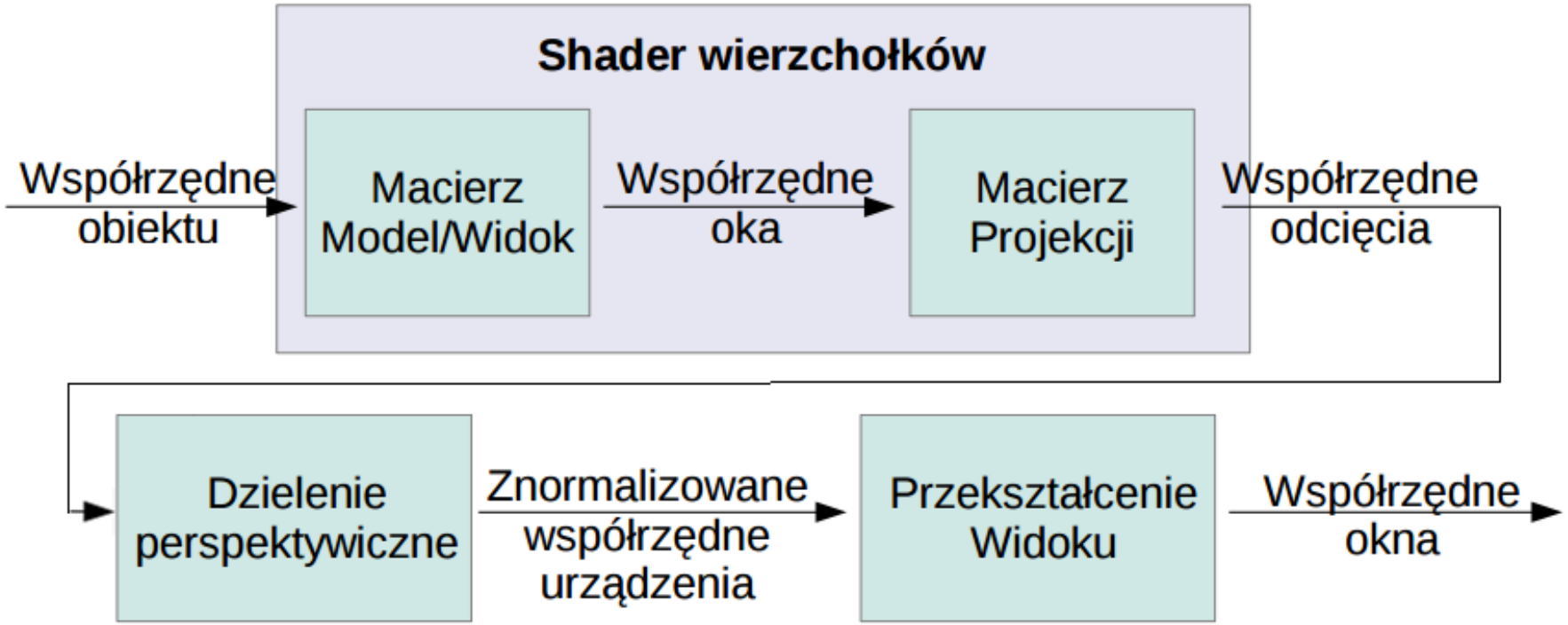
podsumowanie
potok graficzny w OpenGL:

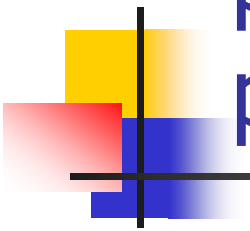
Potok graficzny polega zatem na przekształceniu punktu P leżącego w przestrzeni obiektu w punkt P’ leżący w przestrzeni okna za pomocą ciągu operacji opisanych przekształceniami:

$$\mathbf{P' = (U_z \circ S_w \circ N \circ M \circ K)\ P}$$

gdzie K, M, N, S_w i U_z opisują kolejno przekształcenia jednej przestrzeni w drugą.

podsumowanie
potok graficzny w OpenGL:

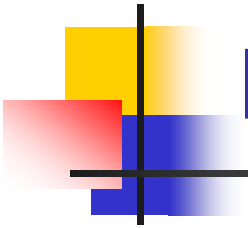




podsumowanie
potok graficzny w OpenGL:

Shader to program napisany w specjalnym języku (np. GLSL – OpenGL Shading Language), który w grafice trójwymiarowej opisuje właściwości pikseli oraz wierzchołków.

Cieniowanie pozwala na znacznie bardziej skomplikowane modelowanie oświetlenia i materiału na obiekcie niż standardowe modele oświetlenia i teksturowanie. Jest jednak dużo bardziej wymagające obliczeniowo i dlatego dopiero od kilku lat sprzętowa obsługa cieniowania jest obecna w kartach graficznych komputerów domowych.



Literatura pomocnicza:

http://wazniak.mimuw.edu.pl/index.php?title=Grafika_komputerowa_i_wizualizacja (kurs autorstwa Dariusza Sawickiego):
- Moduł 8: Modelowanie oświetlenia
- Moduł 9: Oświetlenie globalne
- Moduł 10: Dążenie do realizmu

Kurs OpenGL, C++
http://cpp0x.pl/kursy/Kurs-OpenGL-C++/101
OpenGL Programming Guide (Addison-Wesley Publishing Company)
http://neo.dmcs.pl/tgk/redbook.pdf
OpenGL Programming Guide
http://www.glprogramming.com/red/index.html
Wojciech Kowalewski: Wykłady z OpenGL – materiały dostępne na Contact.dir

KONIEC