# Laboratorium Grafiki Komputerowej i Animacji

## Ćwiczenie VI

## Biblioteka OpenGL - teksturowanie

Sławomir Samolej

Rzeszów, 2013

## 1. Wstęp

Podczas tworzenia skomplikowanych obiektów graficznych przydatnym mechanizmem staje się teksturowanie - możliwość "pokrywania" wielokątów zapisanymi cyfrowo obrazami. Teksturowanie pozwala na odwzorowywanie własności pewnego typu materiałów pokrywających powierzchnię obiektu bez konieczności czasochłonnego budowania tych powierzchni z prymitywów graficznych. W systemach komputerowych z wbudowaną akceleracją sprzętową obliczania tekstur teksturowanie umożliwia znaczne przyspieszenie pracy animacji, przy bardziej realistycznych efektach wizualnych. Zastosowanie odpowiednich tekstur wzmacnia złudzenie rzeczywistości wyświetlanej sceny.

Opracowanie dotyczy sposobu wykorzystania plików typu BMP (bitmapa) jako obrazów cyfrowych przeznaczonych do teksturowania obiektów tworzonych przy pomocy biblioteki OpenGL. Dodatkowo omówiony jest sposób wykorzystywania zestawu obiektów geometrycznych wchodzących w skład biblioteki glu. Do opracowana dołączona jest dyskietka zawierająca omawiane w tekście przykłady.

## 2. Teksturowanie

W standardzie OpenGL teksturowanie obiektów powinno się składać z następujących etapów:
- Przygotowania tekstury;
- Ustalenia, w jaki sposób tekstura odpowiadać będzie poszczególnym pikselom obrazu;
- Uaktywnienia teksturowania w systemie OpenGL;
- Utworzenia sceny zawierającej prymitywy i "rozciągnięte" na nich tekstury.

Przygotowanie tekstury polega na włączeniu do pamięci programu tablicy pikseli reprezentującej jedno lub dwuwymiarowy obraz. Poniżej zaprezentowany zostanie sposób włączania plików DIB, jako tekstur OpenGL. Budowa pliku typu BMP omówiona została w opracowaniu dotyczącym ćwiczenia II z przedmiotu Grafika Komputerowa i Animacja.

Rozmiar tablicy pikseli musi spełniać zależność: $(2^m + 2b) \times (2^n + 2d)$, gdzie m, n, b, d są liczbami całkowitymi. Liczby b i d określają szerokość dodatkowej ramki obrazu. Minimalny rozmiar obrazu musi wynosić 64×64 piksele. Należy pamiętać, że w plikach typu bitmapa dane mówiące o kolorze piksela zorganizowane są w kolejności BGR (składowe: niebieska-zielona-czerwona). Standard OpenGL oczekuje tablicy pikseli zorganizowanej w kolejności RGB (składowe: czerwona-zielona-niebieska). W aplikacji dostarczonej jako przykład teksturowania zastosowano specjalną funkcję zaczerpniętą z [7]:

```
unsigned char *LoadBitmapFile(
      char *filename,
      BITMAPINFOHEADER *bitmapInfoHeader
);
```

Funkcja otwiera plik bitmapy wskazany jako *filename*, przekazuje część nagłówka pliku do struktury *bitmapInfoHeader*, zamienia dane w tablicy pikseli z opisu barw BGR na RBG i zwraca tablicę pikseli, która może być bezpośrednio przekazana do funkcji generującej teksturę OpenGL.

Definiowanie tekstury w standardzie OpenGL rozpoczyna się od wywołania funkcji:

```
void WINAPI glGenTextures(
    GLsizei n,
    GLuint *textures);
```

W funkcji określa się ilość tekstur, które będą generowane (*n*) oraz wskazanie do pierwszego elementu tablicy, która będzie zawierała identyfikatory tekstur (*textures*). Funkcja wypełnia tablicę unikalnymi identyfikatorami tekstur. Podczas generacji tekstur OpenGL następuje powiązanie aktualnie tworzonej tekstury ze wskazanym identyfikatorem (wskazanym elementem tablicy identyfikatorów).

Kolejnym etapem definiowania tekstury jest wywołanie funkcji:

```
void WINAPI glBindTexture(
    GLenum target,
    GLuint texture
);
```

Pierwszy parametr wskazuje rodzaj generowanej tekstury (GL_TEXTURE_1D – tekstura jednowymiarowa lub GL_TEXTURE_2D tekstura dwuwymiarowa). Drugi (target) wskazuje identyfikator tekstury, który będzie powiązany z daną teksturą. Funkcja **glBindTexture**() stosowana jest w 2 przypadkach. W pierwszym, podczas generowania tekstury wiąże ona numer generowanej tekstury z tą teksturą. W drugim stosuje się ją do „przywołania" z puli dostępnych tekstur, tej, którą chcemy w danej chwili pokryć jakiś obiekt na scenie.

Włączenie teksturowania do sceny poprzedzone musi być modyfikacją ustawień systemowych OpenGL. Podstawową funkcją ustalającą parametry teksturowania jest **glTexEnv*()**. Wywołanie tej funkcji w sposób:

```
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);
```

ustala sposób określania koloru tekstur na scenie. Ostatni parametr wywołania funkcji może przyjmować wartości: **GL_MODULATE**, **GL_DECAL** oraz **GL_BLEND**. Jeśli wynosi on GL_DECAL to kolor pikseli tekstury pokrywa w całości teksturowany obiekt i nie jest możliwe uzyskanie efektów związanych z oświetleniem. Parametr GL_MODULATE powoduje, że kolor pokrytego teksturą obiektu filtrowany jest przez kolor pokrywającej go tekstury. Parametr GL_BLENT ustala, że podstawowe znaczenie w wyświetlaniu koloru teksturowanego obiektu ma zdefiniowany kolor prymitywu. Jeśli nie zostanie zdefiniowany stopień przeźroczystości prymitywu to tekstura nie będzie wcale widoczna. Podanie parametrów GL_BLEND i GL_MODULATE pozwala na uzyskiwanie efektów związanych z oświetleniem. System OpenGL musi zostać poinformowany o wprowadzeniu teksturowania.

Do zdefiniowania w OpenGL tekstury dwuwymiarowej służy funkcja o prototypie:

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

Funkcja pobiera dziewięć argumentów. Argument *target* podaje, jakiego typu tekstura ma zostać zdefiniowana i musi on przyjmować wartość GL_TEXTURE_2D. Argument *level* wskazuje poziom oddawania szczegółów zawartych w teksturze i zwykle ustawiany jest na 0. Inne wartości tego argumentu służą do zdefiniowania mniej szczegółowych postaci tekstur. Argument *components* wskazuje ile wartości liczbowych w tablicy pikseli opisuje jeden piksel ekranowy. Dla trybu opisu koloru RGB wartość powinna wynosić 3 dla trybu RGBA - 4. Wartość 1 oznacza, że kolory tekstury opisywane będą w trybie indeksowym (przy pomocy palety kolorów). Parametry *width, height* i *border* opisują rozmiary obrazu. Wartość *border* kontroluje ilość pikseli tworzących ramkę wokół obrazu i może przyjmować wartości 0, 1, 2. Parametr *width* określa długość tekstury i musi być to liczba będąca potęgą liczby 2. Parametr *height*

określa długość tekstury i musi być to liczba będąca potęgą liczby 2. Argument *format* oznacza typ opisywania kolorów pikseli i może przyjmować wartości GL_COLOR_INDEX (gdy kolory opisywane są w trybie indeksowym), GL_LUMINANCE, GL_RGB (gdy kolor opisują trzy liczby), GL_RGBA (gdy kolor opisują cztery liczby). Parametr *type* określa typ danych przechowujących informacje o kolorach pikseli (dla plików typu bitmapa parametr ten powinien wynosić GL_UNSIGNED_BYTE). Ostatni parametr wywołania funkcji jest wskaźnikiem do tablicy pikseli.

Do określania właściwości tekstur w OpenGL służy funkcja **glTexParameter*()**. Pierwszy argument funkcji wskazuje, jakiego typu tekstura będzie modyfikowana (GL_TEXTURE_1D - jednowymiarowa lub GL_TEXTURE_2D - dwuwymiarowa). Argument drugi wyznacza własność teksturowania, która będzie modyfikowana. Zaś argument trzeci opisuje sposób modyfikowania własności. Jeśli drugim parametrem wywołania funkcji jest GL_TEXTURE_MAG_FILTER, to funkcja opisuje, w jaki sposób zachowywać się ma bitmapa, gdy "rozciągana" będzie na obiekcie o rozmiarach większych niż rozmiary obrazu. W przypadku, gdy drugi parametr przyjmuje wartość GL_TEXTURE_MIN_FILTER, funkcja określa sposób "rozciągania" bitmapy na obiekcie mniejszym niż rozmiar obrazu. Jeśli trzeci parametr wywołania funkcji przyjmuje wartość GL_LINEAR, to przed wyrysowaniem tekstury jest ona liniowo interpolowana. Gdy trzecim parametrem wywołania funkcji jest GL_NEAREST rozszerzanie lub zmniejszanie tekstury polega na powielaniu wybranych pikseli.

Podany poniżej fragment kodu pochodzi z programu **glTemplate_tex**, jest wywoływany w obsłudze komunikatu WM_CREATE i definiuje dwuwymiarową teksturę OpenGL na podstawie wczytanego pliku typu bitmapa:

```
// ładuje obraz tekstury:
bitmapData = LoadBitmapFile("checker.bmp", &bitmapInfoHeader);
glGenTextures(1, &texture[0]);                  // tworzy obiekt tekstury
glBindTexture(GL_TEXTURE_2D, texture[0]);       // aktywuje obiekt tekstury

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP );

// tworzy obraz tekstury:
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, bitmapInfoHeader.biWidth,
             bitmapInfoHeader.biHeight, 0, GL_RGB, GL_UNSIGNED_BYTE,
             bitmapData);
// ustalenie sposobu mieszania tekstury z tłem
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,GL_MODULATE);
// zwolnienie zaalokowanej pamięci:
if(bitmapData) free(bitmapData);
```

Jedną z typowych metod teksturowania wielokątów można zawrzeć w następujących punktach:

- Włączenie teksturowania;
- Wybranie tekstury, która ma być nakładana;
- Nałożenie tekstury;
- Wyłączenie teksturowania.

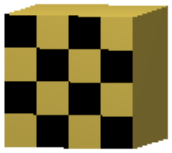
Podany poniżej fragment kodu „rozpina” pewną teksturę na czworokącie, rezultat działania programu pokazano na rysunku 2.1:

```
glBindTexture(GL_TEXTURE_2D,texture[0]);
```

```
glEnable(GL_TEXTURE_2D); // Włącz teksturowanie

glBegin(GL_QUADS);
      glNormal3d(0,0,1);
      glTexCoord2d(1.0,1.0); glVertex3d(25,25,25);
      glTexCoord2d(0.0,1.0); glVertex3d(-25,25,25);
      glTexCoord2d(0.0,0.0); glVertex3d(-25,-25,25);
      glTexCoord2d(1.0,0.0); glVertex3d(25,-25,25);
glEnd();

glDisable(GL_TEXTURE_2D); // Wyłącz teksturowanie
```



Rys. 2.1 Rezultat „rozpięcia" tekstury na jednej ze ścian sześcianu

Funkcja **glBindTexture()** wskazuje, którą z dostępnych tekstur wybieramy. Funkcja **glEnable(GL_TEXTURE_2D)** włącza teksturowanie. Do umiejscowienia tekstury na poszczególnych wielokątach wykorzystuje się funkcję **glTexCoord*()**. Parametry tej funkcji to są współrzędne tekstury. "Rozpinanie" bitmapy na prymitywie polega kolejno na podaniu współrzędnych tekstury, które mają być przywiązane do wierzchołka (funkcja glTexCoord*()), a następnie współrzędnych samego wierzchołka. Utworzona w OpenGL dwuwymiarowa tekstura ma zawsze wymiary 1.0 na 1.0. Oznacza to przykładowo, że jeśli w funkcji glCoord2f() podamy jako parametry liczby 0.0 i 1.0 to odwołujemy się do lewego górnego rogu obrazu graficznego. Jeśli liczby podawane jako współrzędne tekstur przekraczają zakres 0.0 - 1.0, to wyświetlanie bitmap zależy od ustawień podanych w wywołaniu funkcji glTexParameter*() wywołanej z pierwszym parametrem GL_TEXTURE_2D i z drugim parametrem GL_TEXTURE_WRAP_S lub GL_TEXTURE_WRAP_T (litera S odnosi się do współrzędnej poziomej tekstury, litera T odnosi się do współrzędnej pionowej tekstury). Jeśli przykładowo wywołanie funkcji wygląda:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
```

to trzeci parametr (GL_CLAMP) decyduje, że rozmiary tekstury ograniczone są do wielkości od 0.0 do 1.0 w kierunku poziomym i tekstura nie jest powielana w celu pokrycia obiektu. Wywołanie tej samej funkcji w sposób:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
```

zapewnia, że w przypadku podania w wywołaniu funkcji glCoord2f() parametrów większych od 1.0 rozłożenie tekstury na obiekcie polegać będzie na odpowiednim powieleniu mapy bitowej wzdłuż osi poziomej (Podanie tej samej komendy z parametrem GL_TEXTURE_WRAP_T decydowało będzie o sposobie powielania tekstury wzdłuż osi poziomej tekstury). Przykładowe wywołania funkcji glTexParameter*() z omówionymi powyżej parametrami znajdują się w prezentowanym wcześniej fragmencie kodu pokazującym technikę przygotowywania tekstury. Funkcja **glDisable(GL_TEXTURE_2D)** wyłącza teksturowanie.

## 4. Kwadryki OpenGL - sfery, cylindry, dyski

Biblioteka OpenGL, w celu ułatwienia konstruowania bardziej złożonych scen zawiera zdefiniowane proste trójwymiarowe obiekty zwane kwadrykami. Definicje obiektów są składnikami biblioteki GLU (OpenGL Utility Library). Zestaw komend biblioteki GLU pozwala na tworzenie stożków, cylindrów, dysków i sfer. Każda kwadryka rysowana na ekranie posiada zestaw związanych z nią ustawień. Funkcja **gluNewQuadric()** tworzy strukturę danych identyfikującą kwadrykę w systemie oraz zawierającą zestaw zmiennych definiujących jej właściwości (sposób rysowania na ekranie, orientacja, tryb oświetlania, tryb teksturowania, funkcje odwołujące się do obiektu):

```
GLUquadricObj *obj;
obj= glNewQuadric();
```

Struktura nie zawiera informacji o kształcie kwadryki.

Po utworzeniu w pamięci kwadryki można dostosować jej kształt i właściwości do własnych potrzeb. Służą do tego następujące funkcje:

```
void gluQuadricDrawStyle( GLUquadricObj * qobj, GLenum drawStyle );
void gluQuadricNormals( GLUquadricObj *qobj, GLenum normals );
void gluQuadricOrientation( GLUquadricObj * quadObject, GLenum orientation );
void gluQuadricTexture( GLUquadricObj * quadObject, GLboolean textureCoords
);
```

Pierwszym parametrem wywołania wszystkich wymienionych funkcji jest wskaźnik do struktury identyfikującej utworzoną kwadrykę. Funkcja gluQuadricDrawStyle() ustala typ prymitywów graficznych, przy pomocy których tworzony będzie obiekt. Domyślnym ustawieniem jest tworzenie obiektów z pasów złożonych z wielokątów - prymitywy OpenGL typu STRIP. Zestawienie dostępnych parametrów funkcji przedstawia tabela poniżej:

| **Parametr drawStyle** | **Opis** |
|---|---|
| GLU_FILL | Kwadryka rysowana jest przy pomocy wypełnionych pasów wielokątów. |
| GLU_LINE | Kwadryka rysowana jest w postaci siatki składającej się z linii |
| GLU_SILHOUETTE | Kwadryka rysowana jest w postaci siatki złożonej z linii, widoczne są tylko zewnętrzne krawędzie |
| GLU_POINT | Kwadryka rysowana jest w postaci chmury wierzchołków |

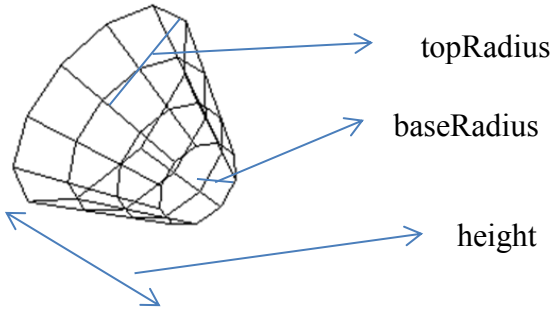
Normalne do kwadryk generowane są zwykle w sposób automatyczny. Funkcja gluQuadricNormals() steruje sposobem wyliczania normalnych. Dostępne warianty obliczania reakcji na padające światło dla kwadryk prezentuje tabela poniżej:

| Parametr normals | Opis |
|---|---|
| GLU_NONE | Normalne nie są generowane |
| GLU_FLAT | Normalne generowane są jako prostopadłe do wielokątów tworzących powierzchnię |
| GLU_SMOOTH | Normalne generowane są dla każdego wierzchołka osobno w celu nadania "gładkości" powierzchniom tworzącym obiekt |

Do ustalenia zwrotu wygenerowanych normalnych służy funkcja gluQuadricOrientation(). Jeśli normalne skierowane mają być na zewnątrz to drugim parametrem wywołania funkcji powinna być stała GLU_OUTSIDE. W przeciwnym wypadku - GLU_INSIDE.

Możliwe jest także automatyczne wygenerowanie koordynat tekstur pokrywających kwadryki. Funkcja gluQuadricTexture() uaktywnia (GL_TRUE) lub zabrania (GL_FALSE) automatyczną generację współrzędnych tekstur przeznaczonych do pokrywania kwadryg.

**Cylindry** tworzy się przy pomocy funkcji **gluCylinder()**. Utworzony obiekt jest w rzeczywistości tubą, której oś przebiega wzdłuż osi "z" układu współrzędnych. Końce tuby nie są nigdy wypełniane (rys 4.1).




Rys. 4.1 Interpretacja parametrów funkcji gluCylinder()

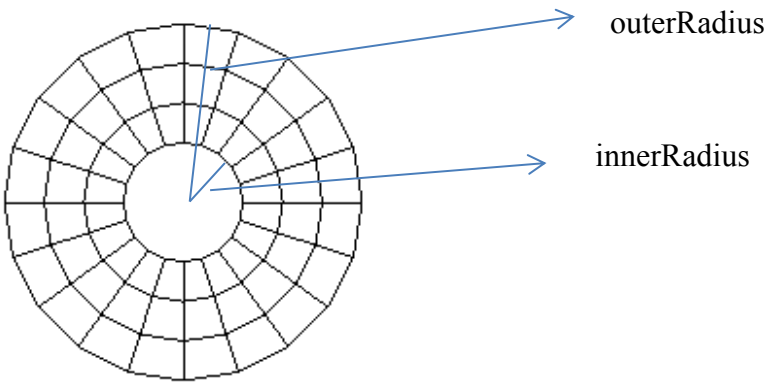
Prototyp funkcji ma postać:

```
void gluCylinder(
GLUquadricObj * qobj,
GLdouble baseRadius,
GLdouble topRadius,
GLdouble height,
GLint slices,
GLint stacks );
```

Parametry *baseRadius* i *topRadius* definiują promienie dolnej i górnej podstawy cylindra. Parametr height definiuje wysokość cylindra. Argumenty *slices* i *stacks* decydują o ilości wielokątów, z których składa się obiekt. Zwykle do utworzenia "gładkiego" walca wystarcza ustawienie parametru *slices* na 20. Dla uzyskania wysokiej jakości efektów oświetlenia obiektu parametr *stacks* powinien być zbliżony do parametru *slices*. Dla zwykłych zastosowań ustala się go na 2. Obiekty typu cylinder mogą także posłużyć do przybliżania brył o przekroju wielokąta jak na przykład ołówek. Zdefiniowany jak powyżej cylinder można w prosty sposób przekształcić w **stożek**. Wystarczy tylko nadać jednemu z promieni walca wartość 0. Nakładanie tekstury na cylinder odbywa się od punktu o współrzędnych (0, radius,0).

**Dyski** w bibliotece GLU mają postać płaskich kół z możliwością zdefiniowania w nich kolistych otworów o środku pokrywającym się ze środkiem pierwotnego koła (rys. 4.2). Prototyp funkcji definiującej dysk ma postać:

```
void gluDisk(
GLUquadricObj * qobj,
GLdouble innerRadius,
GLdouble outerRadius,
GLint slices,
GLint loops );
```




Rys. 4.2 Interpretacja parametrów funkcji gluDisk()

Parametry *innerRadius* i *outerRadius* decydują o rozmiarze dysku i ewentualnym rozmiarze otworu wewnątrz niego (rys. 4.2). Jeśli parametr *innerRadius* ustalony jest na 0, to dysk rysowany jest jako wypełnione koło. Parametr *slices* decyduje z ilu wycinków koła przybliżonych wielokątami składać się będzie rysowany obiekt (Do uzyskania efektu gładkości zwykle wystarcza wartość 20). Parametr *loops* decyduje o ilości koncentrycznych pierścieni dzielących dysk (znajdujących się pomiędzy wewnętrzną a zewnętrzną średnicą dysku). Zwykle parametr ten ustala się na 1 dla kół lub na 2 dla kół z wyciętym otworem wewnątrz. Dla osiągnięcia specjalnych efektów z oświetleniem liczbę tę należy powiększyć.

Istnieje możliwość zdefiniowania **wycinka dysku**. Dodatkowymi parametrami nowej funkcji są kąt początkowy (*startAngle*) i kąt końcowy (*sweepAngle*) liczone zgodnie z kierunkiem obrotu wskazówek zegara od góry dysku:

```
void gluPartialDisk(
GLUquadricObj * qobj,
GLdouble innerRadius,
GLdouble outerRadius,
GLint slices,
GLint loops,
GLdouble startAngle,
GLdouble sweepAngle );
```

Do tworzenia sfer służy funkcja o prototypie:

```
void gluSphere(
GLUquadricObj * qobj,
GLdouble radius,
GLint slices,
GLint stacks );
```

Parametr *radius* decyduje o promieniu sfery. Parametry slices (ilość południków sfery) i stacks (ilość równoleżników sfery) decydują o ilości wielokątów, z których zbudowany będzie obiekt.

Jeśli maszyna stanu OpenGL jest w swoich standardowych ustawieniach, to kwadryki rysowane są w postaci powierzchni, do których automatycznie wystawiane są normalne. Stąd budowanie niezłożonych, oświetlonych scen z kwadryk jest relatywnie proste. Jeśli kwadryka ma być pokryta teksturą, to wystarczy „włączyć” dla niej teksturowanie i wskazać, którą teksturą ma być pokryta. Podany poniżej kod pokazuje, jak narysować kulę pokryta kwadryką. Rezultat działania programu jest pokazany na rysunku 3.1.

```
void kula(void)
{     GLUquadricObj*obj;
```

```
obj=gluNewQuadric();
gluQuadricTexture(obj,GL_TRUE);
glBindTexture(GL_TEXTURE_2D,texture[0]);
glColor3d(1.0,0.8,0.8);
glEnable(GL_TEXTURE_2D);
gluSphere(obj,20,15,7);
glDisable(GL_TEXTURE_2D);}
```

Rys. 3.1 Przykładowa kwadryka pokryta teksturą

Bibliografia:

[1] R. S. Wright Jr., M. Sweet, *OpenGL Księga Eksperta*, Helion, 1999
[2] *OpenGL Programming Guide*, Addison-Wessley, 2006
[3] E. Angel, *Interactive Computer Graphics*, Addison-Wesley, 2008
[4] R. Fosner, *OpenGL Programming for Windows and Windows NT*, Addison-Wesley, 1997
[5] R. Leniowski, *Wykłady z przedmiotu Grafika Komputerowa i Animacja*
[6] *Guide to OpenGL® on Windows® From Silicon Graphics®*, 1997
[7] K. Hawkins, D. Astle, *OpenGL programowanie gier*, Helion 2003