

Janusz Ganczarski

# OpenGL

Podstawy programowania grafiki 3D



Helion 

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Opieka redakcyjna: Ewelina Burska

Projekt okładki: Studio Gravite/Olsztyn

Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/opglwp>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-0193-1

Copyright © Helion 2015

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# Spis treści

<b>Rozdział 1. Od autora .....</b>	<b>7</b>
<b>Rozdział 2. Wprowadzenie do OpenGL .....</b>	<b>11</b>
Co to jest OpenGL? .....	11
Historia OpenGL .....	13
Potok renderingu .....	15
Kontekst renderingu .....	16
Typy danych .....	18
Zmienne stanu .....	18
Obsługa błędów .....	20
OpenGL ES i WebGL .....	21
OpenGL NG .....	21
Kwestie techniczne .....	21
<b>Rozdział 3. Pierwszy program .....</b>	<b>23</b>
Obiekty bufora .....	23
Wierzchołki i ich atrybuty .....	26
Prymitywy geometryczne .....	30
Scena 3D i jej przekształcenia .....	32
Podstawy GLSL .....	37
Shadery wierzchołków .....	38
Shadery fragmentów .....	38
Obsługa shaderów .....	40
Pierwszy program .....	45
<b>Rozdział 4. Rzutowanie i transformacje .....</b>	<b>69</b>
Zmienne jednorodnie w shaderach .....	69
Przekształcenia modelu-widoku i rzutowanie .....	73
Dodatkowe płaszczyzny obcinania i płaszczyzny usuwania .....	91
Selekcja obiektu .....	97

<b>Rozdział 5. Cieniowanie i oświetlenie .....</b>	<b>105</b>
RGB .....	105
Cieniowanie wielokątów .....	106
Nazwane bloki jednorodne .....	113
Oświetlenie .....	120
Model oświetlenia Lamberta .....	130
Model oświetlenia Blinna-Phonga .....	135
Model oświetlenia Phonga .....	152
<b>Rozdział 6. Tekstury .....</b>	<b>163</b>
Tablice pikseli .....	163
Podstawy teksturowania .....	168
Tekstury dwuwymiarowe .....	184
Tekstury trójwymiarowe .....	203
Tablice tekstur 2D .....	208
Tekstury sześciennie .....	212
Tekstury skompresowane .....	216
Sprajty punktowe .....	223
Tekstury buforowe .....	228
<b>Rozdział 7. Operacje na fragmentach i przetwarzanie końcowe .....</b>	<b>231</b>
Bufory ramki .....	231
Rendering pozaekranowy .....	237
Rendering do tekstury .....	241
Wiele buforów renderingu .....	246
Rendering wielowarstwowy .....	258
Operacje na fragmentach .....	266
Test szablonu .....	267
Test głębokości .....	272
Mieszanie kolorów .....	276
Test zasłaniania .....	285
<b>Rozdział 8. Antyaliasing .....</b>	<b>295</b>
Wielopróbkowanie .....	296
Wielopróbkowanie w obiektach bufora ramki .....	307
Nadpróbkowanie .....	312
<b>Rozdział 9. Zaawansowane przekształcanie geometrii .....</b>	<b>313</b>
Prymitywy rozszerzone i shadery geometrii .....	313
Teselacja .....	318
Krzywe i powierzchnie parametryczne .....	341
Transformacje sprzężone zwrotnie .....	360

---

<b>Rozdział 10. Cienie .....</b>	<b>369</b>
Rzutowanie cieni .....	369
Bryły cieni .....	376
Mapy cieni .....	390
<b>Rozdział 11. Odwzorowanie środowiska i nierówności powierzchni .....</b>	<b>415</b>
Techniki odwzorowania nierówności powierzchni .....	415
Odwzorowanie środowiska .....	431
<b>Rozdział 12. Rozszerzenia .....</b>	<b>449</b>
Podstawowe informacje o rozszerzeniach .....	449
Obsługa rozszerzeń API OpenGL .....	451
Obsługa rozszerzeń GLSL .....	456
<b>Dodatek A Macierze i wektory .....</b>	<b>463</b>
Macierze .....	463
Wektory .....	468
<b>Dodatek B Język GLSL .....</b>	<b>473</b>
Podstawy składni .....	473
Preprocesor .....	474
Zmienne, typy i konwersje .....	478
Operatory i wyrażenia .....	493
Kwalifikatory .....	496
Instrukcje i struktura programu .....	515
Wbudowane zmienne .....	520
Wbudowane funkcje .....	523
<b>Skorowidz .....</b>	<b>539</b>



# Rozdział 3.

## Pierwszy program

Po wstępnych informacjach dotyczących biblioteki OpenGL przyszedł czas na napisanie pierwszego programu, który z niej korzysta. Nie jest to zadanie banalne, ponieważ wymaga zapoznania się z kilkoma fundamentalnymi elementami OpenGL, bez których praktycznie nie ma programu w OpenGL. Po lekturze tego rozdziału Czytelnik powinien mieć już podstawową wiedzę, jak wygląda program korzystający z tej biblioteki.

### Obiekty bufora

Biblioteka OpenGL grupuje większość funkcjonalności w obiektach. Obiekty w bibliotece OpenGL są oznaczane unikatowymi identyfikatorami reprezentowanymi przez liczby całkowite bez znaku. Każdy obiekt w bibliotece OpenGL posiada swój własny zestaw zmiennych stanu, który jest inicjowany podczas tworzenia obiektu.

**Obiekty bufora** (ang. *buffer object*) służą do przechowywania różnego rodzaju danych w pamięci (a dokładniej w przestrzeni adresowej) serwera OpenGL. Tworząc nowy obiekt, w pierwszej kolejności musimy wygenerować jego identyfikator, a następnie dokonać tzw. **powiązania** (ang. *bind*) wybranego typu obiektu z jego identyfikatorem. Czynność ta tworzy wybrany rodzaj obiektu. OpenGL przechowuje dane w obiekcie bufora jako dane binarne o nieokreślonej strukturze. Dopiero podczas użycia obiektu bufora odpowiednie polecenia OpenGL określają, jak należy interpretować dane w nich zawarte.

### Tworzenie i usuwanie obiektów bufora

Do generowania identyfikatorów obiektów bufora służy funkcja:

```
void glGenBuffers( GLsizei n, GLuint *buffers );
```

która umożliwia jednoczesne utworzenie  $n$  identyfikatorów umieszczanych w tablicy wskazywanej w parametrze `buffers`. W analogiczny sposób można grupowo usuwać obiekty bufora (nie tylko same identyfikatory), do czego służy funkcja:

```
void glDeleteBuffers( GLsizei n, const GLuint *buffers );
```

Utworzenie samego obiektu bufora wymaga wywołania funkcji:

```
void glBindBuffer( GLenum target, GLuint buffer );
```

która w pierwszym parametrze określa rodzaj obiektu bufora, a w drugim parametrze zawiera jego wcześniej utworzony identyfikator. W programach będziemy wykorzystywać **obiekty bufora wierzchołków VBO** (ang. *vertex buffer object*), których zadaniem jest przechowywanie atrybutów wierzchołków. Aby utworzyć obiekty typu VBO, parametr `target` musi przyjąć wartość równą `GL_ARRAY_BUFFER`. Następne wywołania funkcji `glBindBuffer` z danym identyfikatorem pozwalają na przełączanie się pomiędzy różnymi obiektami bufora, a ostatnio tak wybrany obiekt bufora jest nazywany bieżącym obiektem bufora.

Obiekty bufora można także tworzyć jednocześnie z generowaniem ich identyfikatorów, do czego służy funkcja:

```
void glGenBuffers( GLsizei n, GLuint *buffers );
```

Tak utworzone obiekty bufora nie mają określonego rodzaju, który jest ustalany przy pierwszym wywołaniu funkcji `glBindBuffer`.

## Ładowanie danych do obiektu bufora

Nowo utworzony obiekt bufora nie zawiera żadnych danych. Ładowanie danych do obiektów bufora można realizować różnie, ale najczęściej wykonuje się jedną z poniższych operacji:

- ◆ kopiowanie całości lub części danych z przestrzeni adresowej klienta, np. z tablicy lub innej struktury danych;
- ◆ kopiowanie danych z innego obiektu bufora;
- ◆ zmapowanie danych bufora do przestrzeni adresowej klienta OpenGL poprzez pobranie wskaźnika.

Dane obiektu bufora mogą również być ładowane bezpośrednio z potoku renderingu na etapie transformacji sprzężonych zwrotnie, a także zapisywane z poziomu shadera.



Najprostszym i w praktyce najczęściej wykorzystywanym sposobem załadowania danych do bieżącego obiektu bufora jest ich skopiowanie bezpośrednio z pamięci aplikacji OpenGL. Służy do tego funkcja:

```
void glBufferData( GLenum target, GLsizei size, const GLvoid *data,
↳GLenum usage );
```

Parametr `target` określa rodzaj obiektu bufora i powinien przyjąć taką samą wartość jak w funkcji `glBindBuffer`. Drugi parametr — `size` — ustala rozmiar danych obiektu bufora w bajtach, a trzeci parametr — `data` — jest wskaźnikiem na źródło danych bufora znajdujące się w pamięci klienta. Jeśli parametr `data` nie jest pusty, to funkcja `glBufferData` kopiuje wskazane dane do obiektu bufora. W przeciwnym wypadku zawartość danych obiektu bufora jest niezdefiniowana. W każdym przypadku usuwane są wszelkie istniejące dane znajdujące się w obiekcie bufora.



Przy określaniu rozmiarów pamięci, odstępów pomiędzy elementami pamięci i podobnych wielkościach operujących na pamięci specyfikacja OpenGL używa pojęcia **podstawowych jednostek maszyny BMU** (ang. *basic machine unit*), które w pewnym uproszczeniu będziemy w niniejszym tekście utożsamiać z pojęciem 8-bitowego bajta.

Ostatni parametr `usage` jest wskazówką wydajności, informującą OpenGL o spodziewanym sposobie wykorzystania danych obiektu bufora (faktyczny sposób wykorzystania bufora może być inny). Dostępne są następujące wartości tego parametru:

- ♦ `GL_STREAM_DRAW` — zawartość danych obiektu bufora zostanie określona raz przez aplikację i będzie używana co najwyżej kilka razy jako źródło dla poleceń OpenGL do rysowania i specyfikacji obrazu.
- ♦ `GL_STREAM_READ` — zawartość danych obiektu bufora zostanie określona raz przez pobranie danych z OpenGL i będzie pobierana co najwyżej kilka razy przez aplikację.
- ♦ `GL_STREAM_COPY` — zawartość danych obiektu bufora zostanie określona raz przez pobranie danych z OpenGL i będzie używana co najwyżej kilka razy jako źródło dla poleceń OpenGL do rysowania i specyfikacji obrazu.
- ♦ `GL_STATIC_DRAW` — zawartość danych obiektu bufora zostanie określona raz przez aplikację i będzie używana wiele razy jako źródło dla poleceń OpenGL do rysowania i specyfikacji obrazu.
- ♦ `GL_STATIC_READ` — zawartość danych obiektu bufora zostanie określona raz przez pobranie danych z OpenGL i będzie pobierana wiele razy przez aplikację.

- ◆ `GL_STATIC_COPY` — zawartość danych obiektu bufora zostanie określona raz przez pobranie danych z OpenGL i będzie używana wiele razy jako źródło dla poleceń OpenGL do rysowania i specyfikacji obrazu.
- ◆ `GL_DYNAMIC_DRAW` — zawartość danych obiektu bufora będzie określana wielokrotnie przez aplikację i używana wiele razy jako źródło dla poleceń OpenGL do rysowania i specyfikacji obrazu.
- ◆ `GL_DYNAMIC_READ` — zawartość danych obiektu bufora będzie określana wielokrotnie poprzez pobranie danych z OpenGL i pobierana wiele razy przez aplikację.
- ◆ `GL_DYNAMIC_COPY` — zawartość danych obiektu bufora będzie określana wielokrotnie poprzez pobranie danych z OpenGL i używana wiele razy jako źródło dla poleceń OpenGL do rysowania i specyfikacji obrazu.

Wybrany obszar danych bieżącego obiektu bufora można także modyfikować przy użyciu funkcji:

```
void glBufferSubData( GLenum target, GLintptr offset,
↳GLsizeiptr size, const void *data );
```

która określa początek (`offset`) i rozmiar (`size`) modyfikowanych danych obiektu bufora typu określonego w parametrze `target`. Modyfikowane dane przechowywane są w tablicy w pamięci klienta wskazywanej w ostatnim parametrze `data`.



Transfer danych pomiędzy programem korzystającym z OpenGL a pamięcią GPU jest procesem stosunkowo wolnym. Częsty transfer dużych ilości danych może być jednym z tzw. wąskich gardeł aplikacji graficznych, zatem należy takie operacje w programie dobrze zoptymalizować.

## Wierzchołki i ich atrybuty

Jak wspomnieliśmy w poprzednim rozdziale, OpenGL operuje na prymitywach geometrycznych, które są opisane przez wierzchołki i ich atrybuty. Każdy wierzchołek prymitywu jest określony przez jeden lub większą liczbę atrybutów, z kolei każdy atrybut jest określony przez wartość skalarną bądź wektor dwu-, trzy- lub czteroelementowy. Skalary oraz składowe wektora mogą być: liczbami zmiennieprzecinkowymi (pojedynczej lub podwójnej precyzji), liczbami stałoprzecinkowymi (typ `GLfixed`) lub liczbami całkowitymi o różnej liczbie bitów. Możliwe jest także definiowanie atrybutów zawierających macierze, ale w tym wypadku OpenGL traktuje taki atrybut jak zbiór atrybutów wektorowych liczący tyle atrybutów, ile kolumn ma macierz. Typowymi atrybutami wierzchołków są: współrzędne położenia, składowe kolorów, wektory normalne i współrzędne tekstury.

Atrybuty wierzchołków przechowywane są w obiektach bufora wierzchołków VBO, skąd pobierane są i przetwarzane przez shadery wierzchołków w celu wykonywania w późniejszych etapach potoku renderingu.

## Tablice wierzchołków

**Tablice wierzchołków VA** (ang. *vertex arrays*) opisują, w jaki sposób są zorganizowane dane atrybutów wierzchołków zawarte w obiektach VBO. Każda tablica wierzchołków zawiera m.in. unikatowy numer indeksu atrybutu wierzchołka (początek numeracji indeksów zaczyna się od 0), przy czym maksymalnie dostępnych jest `GL_MAX_VERTEX_ATTRIBS` indeksów atrybutów wierzchołków. Stała ta jest zależna od implementacji biblioteki OpenGL, ale nie może być mniejsza niż 16.

## Definiowanie tablic wierzchołków

Typowo do zdefiniowania tablicy wierzchołków z atrybutami zmiennoprzecinkowymi pojedynczej precyzji używana jest funkcja:

```
void glVertexAttribPointer( GLuint index, GLint size, GLenum type,
    ↪ GLboolean normalized, GLsizei stride, const GLvoid *pointer );
```

Parametr `index` zawiera numer indeksu definiowanego atrybutu wierzchołka. Numery indeksów wierzchołków generowane są automatycznie przez OpenGL lub mogą być wskazane wprost w shaderze wierzchołków albo w samym programie. Drugi parametr `size` określa, ile składowych ma pojedynczy atrybut wierzchołka przechowywany w obiekcie VBO. Możliwe są wartości 1, 2, 3 i 4, które oznaczają odpowiednio atrybut w postaci skalara oraz dwu-, trzy- lub czteroelementowego wektora. Poszczególne atrybuty wierzchołków pobierane są kolejno z pamięci obiektu VBO, w przypadku atrybutów wektorowych składowe wektorów są także pobierane kolejno z pamięci obiektu.

Następny parametr `type` określa typ danych atrybutów wierzchołka przechowywanych w obiekcie VBO. Najważniejsze wartości parametru `type` i ich powiązanie z typami danych dostępnymi w bibliotece OpenGL zawiera tabela 3.1. Poza typami prezentowanymi w tabeli atrybuty wierzchołków mogą także występować w tzw. formatach upakowanych, których jednak nie będziemy używać w przykładowych programach.

Kolejny parametr `normalized` wskazuje, czy dane zawarte w obiekcie VBO będące liczbami całkowitymi (`GL_BYTE`, `GL_UNSIGNED_BYTE`, `GL_SHORT`, `GL_UNSIGNED_SHORT`, `GL_INT` i `GL_UNSIGNED_INT`) mają być konwertowane do liczb zmiennoprzecinkowych przez normalizowanie do przedziału  $\langle 0;1 \rangle$  dla typów bez znaku

**Tabela 3.1.** Wybrane typy danych atrybutów w tablicach wierzchołków i odpowiadające im typy danych OpenGL

Parametr type	Odpowiadający typ OpenGL
GL_BYTE	GLbyte
GL_UNSIGNED_BYTE	GLubyte
GL_SHORT	GLshort
GL_UNSIGNED_SHORT	GLushort
GL_INT	GLint
GL_UNSIGNED_INT	GLuint
GL_FIXED	GLfixed
GL_FLOAT	GLfloat
GL_HALF_FLOAT	GLhalf
GL_DOUBLE	GLdouble

lub przedziału  $\langle -1;1 \rangle$  dla typów ze znakiem (wartość parametru równa `GL_TRUE`). Możliwe jest także bezpośrednie przeliczenie wartości takiego atrybutu do liczby zmiennoprzecinkowej (wartość parametru równa `GL_FALSE`). Normalizacja jest typową operacją wykonywaną w grafice komputerowej na liczbach całkowitych przechowujących np. składowe RGB(A) kolorów w plikach graficznych. Na przykład dla składowej koloru opisaną 8-bitową liczbą całkowitą bez znaku 0 przyjmuje po normalizacji wartość 0,0, a 255 przyjmuje po normalizacji wartość 1,0.

W pojedynczym obiekcie VBO można przechowywać dane wielu atrybutów wierzchołków. Dane te możemy np. całkowicie od siebie oddzielić, np. umieszczając kolejno: współrzędne wszystkich wierzchołków, współrzędne wektorów normalnych i na końcu składowe kolorów. Można także umieścić w obiekcie VBO dane atrybutów z przeplotem: współrzędne położenia pierwszego wierzchołka, współrzędne wektora normalnego pierwszego wierzchołka, składowe koloru pierwszego wierzchołka, współrzędne położenia drugiego wierzchołka itd. Sposób ułożenia danych atrybutów w obiekcie VBO określa kolejny parametr `stride`, który definiuje określone w bajtach odstęp między danymi poszczególnych atrybutów wierzchołków. Jeżeli `stride` jest równy zero, to dane definiowanej tablicy atrybutu są kolejno zapisane w obiekcie VBO, w przeciwnym wypadku dane kolejnych atrybutów są odpowiednio od siebie oddzielone.

Ostatni parametr `pointer` określa w bajtach położenie w pamięci obiektu bufora VBO (wskaźnik) pierwszego atrybutu definiowanej tablicy wierzchołków. Wskaźnik ten odnosi się do danych bieżącego obiektu VBO, czyli obiektu wybranego przy ostatnim wywołaniu funkcji `glBindBuffer`. Odpowiednie wykorzystanie tego parametru umożliwia wspomniane wyżej przechowywanie danych różnych atrybutów wierzchołków w odrębnych częściach jednego obiektu VBO.

Jeżeli posiadamy atrybuty zawierające liczby całkowite, to typowo przy definiowaniu tablicy wierzchołków używamy funkcji:

```
void glVertexAttribIPointer( GLuint index, GLint size, GLenum type,  
↳ GLsizei stride, const GLvoid *pointer );
```

która w stosunku do funkcji `glVertexAttribPointer` ma listę dopuszczalnych wartości parametru `type` ograniczoną do liczb całkowitych: `GL_BYTE`, `GL_UNSIGNED_BYTE`, `GL_SHORT`, `GL_UNSIGNED_SHORT`, `GL_INT` i `GL_UNSIGNED_INT`. W przypadku użycia tej funkcji dane atrybutów wierzchołków zawsze przekazywane są bezpośrednio w wartościach całkowitych.

Biblioteka OpenGL zawiera jeszcze funkcję:

```
void glVertexAttribLPointer( GLuint index, GLint size, GLenum type,  
↳ GLsizei stride, const GLvoid *pointer );
```

która służy wyłącznie do definiowania tablic wierzchołków atrybutów zawierających liczby zmiennoprzecinkowe podwójnej precyzji (`GLdouble`). Jedyną dopuszczalną wartość parametru `type` powyższej funkcji to `GL_DOUBLE`. Zauważmy, że analogiczny typ danych obsługuje także funkcja `glVertexAttribPointer`, ale przy jej użyciu dane zmiennoprzecinkowe podwójnej precyzji będą konwertowane do liczb zmiennoprzecinkowych pojedynczej precyzji (`GLfloat`).

## Włączanie i wyłączanie tablicy wierzchołków

Pojedyncza tablica atrybutów wierzchołków jest włączana lub wyłączana przez wywołanie jednego z poniższych poleceń:

```
void glEnableVertexAttribArray( GLuint index );  
void glDisableVertexAttribArray( GLuint index );
```

których parametr `index` określa numer indeksu atrybutu w tablicy wierzchołków.

## Domyślne bieżące atrybuty wierzchołków

Gdy tablica wierzchołków zawierająca wartości atrybutów wierzchołka nie jest aktywna, OpenGL stosuje domyślne bieżące wartości ogólnych atrybutów wierzchołków. Początkowe wartości domyślne dla wszystkich atrybutów ogólnych wierzchołka wynoszą odpowiednio: 0 dla atrybutów skalarnych, (0,0) dla wektorów dwuelementowych, (0,0,0) dla wektorów trójelementowych i (0,0,0,1) dla wektorów czteroelementowych.

OpenGL umożliwia zmianę wartości domyślnych atrybutów wierzchołków. Służy do tego bardzo obszerna grupa funkcji `glVertexAttrib`, które wskazują wartości

domyślnych atrybutów dla wybranego indeksu atrybutu. Przykładowo do określenia wartości domyślnych dla atrybutu będącego wektorem czteroelementowym z liczbami zmiennoprzecinkowymi pojedynczej precyzji najlepiej użyć jednej z następujących funkcji:

```
void glVertexAttrib4f( GLuint index, GLfloat x, GLfloat y, GLfloat z,
↳ GLfloat w );
void glVertexAttrib4fv( GLuint index, const GLfloat *v );
```

OpenGL zapewnia odpowiednią wersję funkcji `glVertexAttrib` dla każdej możliwej kombinacji rodzaju atrybutu wierzchołka i typu danych przechowujących wartości atrybutów.

## Obiekty tablic wierzchołków

Tablice wierzchołków grupowane są w zbiorczych **obiekтах tablic wierzchołków VAO** (ang. *Vertex Array Objects*). Obiekty tablic wierzchołków przechowują informacje o danych tablic wierzchołków i wszystkich związanych z nimi zmiennych stanu używanych w procesie renderingu.

Obsługa obiektów VAO jest analogiczna do obsługi innych obiektów biblioteki OpenGL. Każdy obiekt VAO zawiera unikatowy identyfikator, który należy wygenerować, a następnie utworzyć sam obiekt. Niepotrzebny obiekt VAO należy usunąć. Całość realizują trzy funkcje:

```
void glGenVertexArrays( GLsizei n, GLuint *arrays );
void glDeleteVertexArrays( GLsizei n, const GLuint *arrays );
void glBindVertexArray( GLuint array );
```

Po wywołaniu funkcji `glBindVertexArray` określony w funkcji obiekt VAO jest bieżącym obiektem tablic wierzchołków i wszystkie operacje związane z tablicami wierzchołków (np. `glVertexAttribPointer` i `glEnableVertexAttribArray`) oraz polecenia renderingu, które korzystają z tablic wierzchołków (np. `glDrawArrays` i `glDrawElements`), odnoszą się do wybranego obiektu VAO. Z poleceniami renderingu zapoznamy się bliżej w dalszej części niniejszego rozdziału.

## Prymitywy geometryczne

Biblioteka OpenGL obsługuje siedem podstawowych typów prymitywów geometrycznych: punkty, łamane, łamane zamknięte, odcinki, paski trójkątów, wachlarze trójkątów i trójkąty. Każdy rodzaj prymitywu jest identyfikowany przez stałą używaną w poleceniach renderingu. Są to odpowiednio: `GL_POINTS`, `GL_LINE_STRIP`, `GL_LINE_LOOP`, `GL_LINES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN` i `GL_TRIANGLES`.

# Skorowidz

## A

accumulation buffer, *Patrz:*  
bufor akumulacyjny  
adjacency vertice, *Patrz:*  
wierzchołek przyległy  
algorytm  
Carmack's Reverse,  
*Patrz:* algorytm  
depth-fail  
depth-fail, 380, 388, 389,  
390  
depth-pass, 379, 380, 384,  
386  
dynamicznego  
generowania cieni, 369  
mapy cieni, 390, 391  
najbliższego sąsiada, 177  
odwzorowania środowiska,  
212  
PN Triangles, 331, 333,  
335, 337  
rzutowania cieni, *Patrz:* cień  
rzutowanie  
teselacji Phonga, 333, 334,  
335, 339  
VSM, 410  
z-fail, *Patrz:* algorytm  
depth-fail  
z-pass, *Patrz:* algorytm  
depth-pass  
aliasing, 295  
ambient, 121, 124  
ambient light, *Patrz:* światło  
otoczenia  
ambientne occlusion,  
*Patrz:* okluzja otoczenia  
AMD, 12, 21  
Android, 21  
anisotropic filter, *Patrz:* filtr  
anizotropowy  
antialiasing, 295  
linia, 295  
punkt, 295  
trójkąt, 295  
aplikacja podsystem, *Patrz:*  
podsystem aplikacji  
Apple, 12  
ARB, 11  
Assimp, 22, 419, 422  
asynchronous query, *Patrz:*  
zapytanie asynchroniczne  
atlas tekstury, 209  
atraktor IFS, 363  
attenuation, *Patrz:* światło  
punktowe współczynnik  
tłumienia  
auxiliary storage qualifier,  
*Patrz:* kwalifikator  
przechowywania  
pomocniczy

## B

Béziera  
krzywa, *Patrz:* krzywa  
Béziera  
powierzchnia, *Patrz:*  
powierzchnia Béziera  
biblioteka  
Assimp, *Patrz:* Assimp  
Direct3D, 14  
DirectX, 36  
GLM, *Patrz:* GLM  
Mantle, *Patrz:* Mantle  
Mesa 3D, 12  
OpenGL, *Patrz:* OpenGL  
pomocnicza, 22, 45, 450  
Vulkan, *Patrz:* Vulkan  
WGL, *Patrz:* WGL  
bind, *Patrz:* obiekt  
identyfikator wiązanie  
binormal vector, *Patrz:* wektor  
binormalny  
blend color, *Patrz:* kolor  
mieszania  
blend equation, *Patrz:*  
równanie mieszania  
blend function, *Patrz:* funkcja  
mieszania  
block compression, *Patrz:*  
kompresja blokowa  
blok  
interfejsu, 499  
jednorodny  
domyślny, 69  
nazwany, 69, 105, 113,  
119, 120, 499  
pamięci shadera, 499

- błąd
- GL\_CONTEXT\_LOST, 20
  - GL\_INVALID\_ENUM, 20
  - GL\_INVALID\_FRAME
    - ↳ BUFFER\_OPERATION, 20
  - GL\_INVALID\_OPERATION, 20
  - GL\_INVALID\_VALUE, 20
  - GL\_NO\_ERROR, 20
  - GL\_OUT\_OF\_MEMORY, 20
  - GL\_STACK\_OVERFLOW, 20
  - GL\_STACK\_UNDERFLOW, 20
  - kod, 20
  - obsługa, *Patrz:* obsługa błędów
- BMU, 25
- bounding volume, *Patrz:* bryła ograniczająca
- bryła
- cienia, 369, 376, 377, 378, 384, 386, 388
  - nieskończona, 378
  - rendering, 379, 380, 384, 386, 388, 389, 390, 395, 410
  - skończona, 378
- domknięcie
- przednie, 378, 380, 388, 390
  - tylne, 378, 388
- obcinania, 32
- domyślna, 32, 35
  - ograniczająca, 288
  - widoku kanoniczna, 35
- buffer object, *Patrz:* obiekt bufora
- bufor
- akumulacyjny, 260
  - czyszczenie, 250, 251
  - głębokości, 108, 232, 266, 272, 273, 286, 379, 390
  - czyszczenie, 108
  - dane podniesione do kwadratu, 410
  - maskowanie zapisu składowych, 249
  - koloru, 232, 247
  - czyszczenie, 108
  - do odczytu, 253, 254
  - obiekt, *Patrz:* obiekt bufora ramki, 52, 108, 231, 296
  - do odczytu, 232, 236
  - do zapisu, 232, 236
  - domyślny, 231, 232, 237
  - pobieranie pikseli, 251
  - renderingu, 232
  - wielokrotny, *Patrz:* MRT szablonu, 232, 267, 269
  - maskowanie zapisu składowych, 249
  - wielopróbkowania, 296
- bump mapping, *Patrz:* odwzorowanie nierówności powierzchni

## C

- callback function, *Patrz:* funkcja wywoływana zwrotnie
- canonical view volume, *Patrz:* bryła widoku kanoniczna
- centroid, 535
- CGL, 17
- cieniowanie, 105
- gładkie, *Patrz:* cieniowanie Gourauda
  - Gourauda, 106, 107, 109, 140, 142
  - Phonga, 113, 140, 143
  - płaskie, 106, 107, 110
- cień
- bryła, *Patrz:* bryła cieni dynamiczny, 369
  - mapa, *Patrz:* mapa cieni

- objętościowy, 369, *Patrz:* bryła cieni płaski, 371
  - przezroczystość, 376
  - rzutowanie, 369
  - statyczny, 369
- clip coordinates, *Patrz:* współrzędne obcinania
- clip plane, *Patrz:* płaszczyzna obcinania
- clip volume, *Patrz:* bryła obcinania
- color blending, *Patrz:* mieszanie kolorów
- color picking, *Patrz:* wybieranie kolorów
- compatibility profile, *Patrz:* profil kompatybilny
- conditional rendering, *Patrz:* rendering warunkowy
- cone light, *Patrz:* światło stożek
- convolve filter, *Patrz:* filtr splotowy
- Core OpenGL, *Patrz:* CGL
- core profile, *Patrz:* profil podstawowy
- cube map texture, *Patrz:* tekstura sześcienna
- culling plane, *Patrz:* płaszczyzna usuwania
- curved point-normal triangles, *Patrz:* algorytm PN Triangles
- czajnik z Utah, 137, 349
- czworokąt, 321, 324

## D

- dane typ, *Patrz:* typ
- de Casteljau Paul, 343
- default uniform block, *Patrz:* blok jednorodny domyślny
- depth buffer, *Patrz:* bufor głębokości



depth of field, *Patrz:* głębia ostrości  
 depth offset, *Patrz:*  
   przesunięcie wartości  
   głębokości  
 diffuse, 121, 124  
 diffuse light, *Patrz:* światło rozproszone  
 directional light, *Patrz:* światło kierunkowe  
 DirectX, 84, 217  
 displacement mapping, *Patrz:*  
   mapowanie przemieszczeń,  
   *Patrz:* mapowanie  
   przemieszczeń  
 dyrektywa  
   #, 475  
   #define, 475, 478  
   #elif, 475  
   #else, 475  
   #endif, 475  
   #error, 475  
   #extension, 456, 475, 477  
   #if, 475  
   #ifdef, 475  
   #ifndef, 475  
   #include, 457  
   #line, 475, 476  
   #pragma, 475, 476  
   #undef, 475, 478  
   #version, 57, 459, 475, 476  
 dyspersja, 444, 447  
 dzielenie perspektywiczne, 34  
 dziennik informacyjny, 43, 59, 475

## E

efekt Fresnela, 444  
 EGL, 17  
 elipsoida, 356  
   teselacja, 358  
 environment mapping, *Patrz:*  
   środowisko odwzorowanie  
 eye coordinates, *Patrz:*  
   współrzędne oka

## F

FBO, 231, 232, 238, 241, 242, 307  
   przyłączanie RBO, 237  
   tworzenie, 233  
 filtr  
   anizotropowy, 196, 198  
   bezszwowy, 213  
   dolnoprzepustowy  
     splotowy, 312  
   Gaussa, 312, 411  
   izotropowy, 198  
   Laplace'a, 312  
   liniowy, 177  
   PCF, 401, 402, 404  
     3×3, 404  
     5×5, 405, 409  
   pomniejszający, 177, 189, 196  
   powiększający, 178  
   splotowy, 199, 200, 201  
   trójliniowy, 196, 198  
 flat shading, *Patrz:*  
   cieniowanie płaskie  
 format, 165  
   BC1, 217  
   BC2, 217  
   BC3, 217  
   BC4, 216  
   BC5, 216  
   BC7, 217  
   BPTC, 216, 217, 220  
   DDS, 188  
   ETC, 216  
 format  
   KTX, 188  
   LATC/RGTC, 220  
   RGTC, 216  
   S3TC/DXTC, 217, 220  
   tekstur skompresowanych,  
     216, 217, 220  
 FPS, 291  
 fragment, 13, 15, 231, 266

fragment shader, *Patrz:* shader fragmentów  
 fraktal, 363  
 frames per second, *Patrz:* FPS  
 FreeGLUT, 22, 46, 51, 52, 54, 55  
 FreeImage, 22, 188  
 Fresnela efekt, *Patrz:* efekt Fresnela  
 front cap, *Patrz:* bryła domknięcie przednie  
 funkcja, 525  
   argumentów, 516  
   barrier, 536  
   bitfieldExtract, 299  
   bitfieldInsert, 299  
   BlinnPhongDirectional  
     ↪Light, 136  
   całkowita, 529  
   definiowanie, 516  
   deklarowanie, 516  
   DeleteDrawText, 112  
   DeleteScene, 45, 54, 64  
   DepthFailShadow, 389  
   DepthPassShadow, 386  
   DisplayFloor, 270  
   DisplayScene, 45, 54, 290  
   DisplayWord, 272  
   DisplayWorld, 270  
   DrawBoundingBox, 289  
   DrawObject, 289  
   DrawRoom, 386  
   DrawShadowVolume, 386  
   DrawTeapot, 386  
   DrawText8x16, 112  
   EmitBackCap, 388  
   EmitFrontCap, 388  
   EmitQuad, 384  
   EmitStreamVertex, 536  
   EmitVertex, 536  
   EndPrimitive, 536  
   EndStreamPrimitive, 536  
   EyePosition, 128  
   GenerateShadowMap, 395  
   geometryczna, 527

- funkcja
- glActiveShaderProgram, 71
  - glAttachShader, 42
  - glBeginConditionalRender, 292, 293
  - glBeginQuery, 287
  - glBeginTransformFeedback, 361
  - glBindAttribLocation, 62
  - glBindBuffer, 24
  - glBindBufferRange, 119
  - glBindFragDataLocation
    - ↳ Indexed, 280
  - glBindFramebuffer, 233
  - glBindRenderbuffer, 234
  - glBindSamplers, 181
  - glBindTexture, 175, 176
  - glBindTransformFeedback, 361
  - glBindVertexArray, 30
  - glBlendEquation, 277
  - glBlendEquationSeparate, 277
  - glBlendFunci, 278
  - glBlendFuncSeparate, 277
  - glBlitFramebuffer, 236
  - glBlitNamedFramebuffer, 237
  - glBufferData, 25
  - glBufferSubData, 26, 147
  - glCheckFramebufferStatus, 239
  - glClear, 108, 250
  - glClearBuffer, 250, 251
  - glClearColor, 62
  - glClearNamedFramebuffer, 251
  - glClipControl, 36
  - glColorMaski, 249
  - glCompileShader, 41
  - glCompressedTexImage2D, 219
  - glCompressedTexSub
    - ↳ Image2D, 219
  - glCopyTexImage2D, 185
  - glCopyTexSubImage2D, 186
  - glCreateBuffers, 24
  - glCreateFramebuffers, 233
  - glCreateProgram, 42
  - glCreateRenderbuffers, 234
  - glCreateShader, 40, 41
  - glCreateTextures, 175
  - glCullFace, 283
  - glDeleteBuffers, 24
  - glDeleteFramebuffers, 233
  - glDeleteNamedStringARB, 458
  - glDeleteProgram, 42
  - glDeleteRenderbuffers, 234
  - glDeleteSamplers, 181
  - glDeleteShader, 41
  - glDeleteTextures, 175
  - glDeleteTransformFeed
    - ↳ backs, 361
  - glDeleteVertexArrays, 30
  - glDepthFunc, 273
  - glDetachShader, 43
  - glDisable, 213
  - glDisablei, 276
  - glDisableVertexAttribArray, 29
  - glDrawArrays, 30, 63, 67
  - glDrawBuffer, 247, 248
  - glDrawBuffers, 247
  - glDrawElements, 30, 67, 355
  - glDrawElementsBaseVertex, 355
  - glDrawElementsInstanced, 351
  - glEnable, 213, 270
  - glEnablei, 276
  - glEnableVertexAttribArray, 29, 30, 65
  - glEndConditionalRender, 292
  - glEndTransformFeedback, 361
  - GLenum, 20
  - glwInit, 53
  - glFramebufferTexture, 242
  - glFramebufferTextureLayer, 259, 260
  - glFrontFace, 31, 271
  - glGenBuffers, 23
  - glGenerateMipmap, 174
  - glGenFramebuffers, 233
  - glGenRenderbuffers, 234
  - glGenSamplers, 181
  - glGenTextures, 175
  - glGenTransformFeedbacks, 361
  - glGenVertexArrays, 30
  - glGet, 18
  - glGetActiveUniformBlockiv, 120
  - glGetActiveUniformsiv, 116, 117
  - glGetAttribLocation, 61, 65
  - glGetBooleanv, 18
  - glGetCompressedTexImage, 252
  - glGetDoublev, 18
  - glGetError, 20
  - glGetFloatv, 18
  - glGetFragDataLocation, 246
  - glGetInteger64v, 18
  - glGetIntegerv, 18
  - glGetMultisamplefv, 311
  - glGetnUniform, 73
  - glGetProgramInfoLog, 43, 44
  - glGetProgramiv, 43, 44
  - glGetProgramResource
    - ↳ Index, 116, 119, 156
  - glGetProgramResourceiv, 117, 120
  - glGetProgramResource
    - ↳ Location, 62, 70, 157
  - glGetShaderInfoLog, 44
  - glGetShaderiv, 44

- glGetSubroutineIndex, 156, 157
- glGetSubroutineUniform
  - ↳ Location, 156
- glGetTexImage, 252
- glGetUniform, 73
- glGetUniformIndices, 116
- glGetUniformLocation, 70, 157
- glHint, 220
- glLinkProgram, 43
- glMinSampleShading, 298
- glNamedFramebufferDraw
  - ↳ Buffer, 248
- glNamedFramebufferDraw
  - ↳ Buffers, 248
- glNamedFramebuffer
  - ↳ Texture, 242
- glNamedFramebuffer
  - ↳ TextureLayer, 259, 260
- glNamedRenderbufferStorage, 235
- glNamedStringARB, 457
- glPatchParameterfv, 322, 325
- glPatchParameteri, 322
- glPauseTransformFeedback, 362
- glPixelStore, 164, 252
- glPointParameteri, 223
- glPointSize, 112
- glPolygonMode, 275
- glProgramUniform, 72
- glProgramUniformMatrix, 72
- glProvokingVertex, 106
- glReadPixels, 251, 252
- glRenderbufferStorage, 235
- glRenderbufferStorage
  - ↳ Multisample, 307
- glResumeTransform
  - ↳ Feedback, 362
- glSampleCoverage, 302
- glSampleMaski, 302
- glSamplerParameter, 182
- glShaderSource, 41
- glStencilFunc, 268
- glStencilFuncSeparate, 268, 269
- glStencilMask, 249
- glStencilMaskSeparate, 249
- glStencilOp, 268
- glStencilOpSeparate, 268, 269
- glTexBuffer, 228
- glTexImage2D, 184, 185, 219
- glTexImage2DMultisample, 308
- glTexImage3D, 203, 204
- glTexImage3DMultisample, 308
- glTexParameter, 175, 176, 182
- glTexStorage2D, 187
- glTexSubImage2D, 186
- glTransformFeedback
  - ↳ Varyings, 362
- glUniform, 71, 72
- glUniformMatrix, 72
- glUniformSubroutinesuiv, 156, 157
- glUseProgram, 44
- glutCreateWindow, 53
- glutInit, 52
- glutInitContextProfile, 53
- glutInitContextVersion, 53
- glutInitDisplayMode, 52
- glutInitWindowSize, 53
- glValidateProgram, 43
- glVertexAttrib4f, 30
- glVertexAttrib4fv, 30
- glVertexAttribIPointer, 29
- glVertexAttribLPointer, 29
- glVertexAttribPointer, 27, 30, 65
- glViewport, 35, 36
- groupMemoryBarrier, 537
- InitDrawText, 112
- InitScene, 45, 54, 108, 459
- interpolacyjna, 535
- kontroli pamięci shadera, 537
- kontroli wywołania shadera, 536
- LambertDirectionalLight, 131, 133
- LambertPointLight, 133
- licznika atomowego, 532
- LinkValidateProgram, 59, 60
- LoadShader, 58, 59, 60
- LoadTextures, 205
- LocalAmbientLight, 133
- logarytmiczna, 525
- lookAt, 90
- macierzowa, 528
- main, 51, 517
- memoryBarrier, 537
- memoryBarrierAtomic
  - ↳ Counter, 537
- memoryBarrierBuffer, 537
- memoryBarrierImage, 537
- memoryBarrierShared, 537
- MeshRendering, 424
- MeshRenderingOpacity, 425
- mieszania, 276
- MouseButton, 103
- MouseMotion, 103
- nazwa przyrostek v, 19
- obrazu tekstury, 533
- odpytująca teksturę, 529
- ortho, 79
- pakowania liczb zmiennoprzecinkowych, 526
- pamięci, 532
- PhongDirectionalLight, 153
- PhongPointLight, 153
- PickObject, 101
- pierwiastkowa, 525
- porównywania wektorów, 528
- potęgowa, 525
- próbująca teksturę, 530

## funkcja

przeładowanie nazwy, 516  
 przestarzała, 13  
 reflect, 433  
 Reshape, 45, 54, 80  
 rozbłyску, 121, 135  
 rozkładu odbicia, 121  
 rozpakowywania liczb  
 zmiennoprzecinkowych,  
 526  
 różniczkowa, 534  
 shadera geometrii, 536  
 teksturowa, 529  
 TranslateObject, 102  
 trygonometryczna, 524  
 unProject, 100  
 wbudowana, 299, 523  
 wielopróbkująca teksturę,  
 531  
 wykładnicza, 525  
 zdefiniowana przez  
 rozszerzenie, 450

**G**

generator prymitywów,  
*Patrz:* teselator  
 geometria instancyjna, 205,  
 344, 349  
 geometry instancing,  
*Patrz:* geometria instancyjna  
 geometry shader,  
*Patrz:* shader geometrii  
 GLEW, 22, 46  
 glFramebufferRenderbuffer, 237  
 GLI, 22, 188  
 GLM, 22, 45, 74  
 glNamedFramebufferRenderb  
 uffer, 238  
 GLSL, 13, 14, 155, 299, 473  
 kompilator, 476  
 preprocesor, 474  
 program, 515  
 rozszerzenie, 456, 477

GL\_ARB\_shading\_  
 language\_include, 457,  
 458, 459, 461  
 składnia, 37, 473  
 słowo zarezerwowane, 474  
 wersja, 37, 57, 476  
 zmienna, *Patrz:* zmienna  
 GLUT, 22, 296, 297  
 GLX, 17, 232, 450  
 głębia ostrości, 264  
 Gouraud shading, *Patrz:*  
 cieniowanie Gourauda

**H**

half vector, *Patrz:* wektor  
 połówkowy  
 height map, *Patrz:* mapa  
 wysokości  
 hiperboloida, 356  
 homogeneous coordinates,  
*Patrz:* współrzędne  
 jednorodne

**I**

immutable format, *Patrz:*  
 tekstura o stałym formacie  
 implementacja, 12  
 indexed buffer object, *Patrz:*  
 obiekt bufora indeksowany  
 Intel, 12  
 iOS, 21  
 IRIS, 11  
 iterated function system,  
*Patrz:* IFS

**J**

jednostka  
 podstawowa maszyny,  
*Patrz:* BMU  
 teksturująca, 174  
 język  
 cieniowania GLSL,  
*Patrz:* GLSL  
 IRIS GL, 11

**K**

kafelkowanie, 14  
 kamera, *Patrz:* obserwator  
 karta graficzna, 21, 292  
 Khronos Group, 12  
 klient-serwer, 15  
 Kocha krzywa, *Patrz:* krzywa  
 Kocha  
 kolor  
 mieszania, 276, 279  
 mieszanie, *Patrz:* mieszanie  
 kolorów  
 próbki, *Patrz:* próbka kolor  
 komentarz, 473  
 kompresja  
 blokowa, 216  
 obrazów HDR, 216  
 obrazów  
 monochromatycznych,  
 216  
 konsolidator, 49  
 konstruktor, 482  
 kontekst renderingu, 13,  
*Patrz też:* rendering kontekst  
 krawędź szkieletowa, 377, 378  
 krzywa  
 Béziera, 343, 345, 348  
 punkt kontrolny, 343  
 rendering, 343  
 stopień, 343  
 teselacja, 346  
 Kocha, 363  
 kwadryka, 356  
 kwalifikator, 496, 501, 502,  
 503, 504, 505  
 formatu  
 bloku, 507, 508  
 liczników atomowych, 511  
 obrazów tekstury, 510  
 transformacji  
 sprzężonych zwrotnie,  
 511  
 interpolacji, 513  
 inwariancji, 514

pamięci, 514, 515  
 parametru funkcji, 517  
 precyzji, 514  
 przechowywania, 496, 497, 498  
 wejściowego, 505, 506  
 wyjściowego, 505

## L

Laplace'a wzór, 466  
 layout qualifier, *Patrz:*  
 kwalifikator formatu  
 level of detail, *Patrz:* LOD  
 libktx, 22, 188  
 licznik atomowy, 489, 511, 532  
 light map, *Patrz:* mapa  
 świetlna  
 light source, *Patrz:* źródło  
 światła  
 lighting equation, *Patrz:*  
 równanie oświetlenia  
 lighting model, *Patrz:* model  
 oświetlenia  
 linia  
 przecięcie z obiektem, 100, 101  
 przecięcie z płaszczyzną, 97  
 przecięcie z trójkątem, 99  
 local tangent plane, *Patrz:*  
 przestrzeń lokalna styczna  
 LOD, 179, 356

## Ł

łamana, 30

## M

macierz, 72, 463  
 diagonalna, 465  
 element, 488  
 główna przekątna, 464  
 inicjalizacja, 487  
 jednostkowa, 464

kwadratowa, 468  
 mnożenie, 464, 495, 528  
 przez skalar, 465  
 modelu, 89  
 modelu-widoku, 34, 88, 89, 90  
 przekształcenia, 73  
 nieodwracalna, *Patrz:*  
 macierz osobliwa  
 nieosobliwa, 467  
 niezdegenerowana, 467  
 odwracanie, 528  
 odwrotna, 467  
 ortogonalna, 77, 468  
 osobliwa, 467  
 przekształcenia  
 kamery, 90  
 obiektu sceny, 90  
 reprezentacja, 84  
 rzutowana, 34  
 cienia, 370, 371  
 perspektywicznego, 85, 87, 88  
 prostokątnego, 78, 79, 80  
 skalowania, 76  
 transformacji, 74  
 translacji, 75  
 transponowana, 468  
 transponowanie, 466, 528  
 układ, 84  
 widoku, 89  
 wyznacznik, 466, 528  
 równy 0, 467  
 zdegenerowana, 467  
 magnification filter,  
*Patrz:* filtr powiększający  
 makro, 478  
 \_\_FILE\_\_, 478  
 \_\_LINE\_\_, 478  
 \_\_VERSION\_\_, 478  
 Mantle, 21  
 mapa, *Patrz też:* mapowanie  
 cieni, 369, 390, 391, 392  
 filtrowanie, 401  
 wariancyjna, 410

przemieszczeń, 425, 427  
 świetlna, 369  
 wektorów normalnych,  
 415, 416, 417, 419  
 wysokości, 415, 416  
 mapowanie, *Patrz też:* mapa  
 cieni, 390, 391  
 otoczenia, 431, 433  
 dynamiczne, 436  
 przemieszczeń, 415, 425, 427  
 UV, 168  
 wektorów normalnych, 415, 417, 419, 420, 421  
 maszyna stanów, 18, 19  
 materiał, 121, 122  
 błyszczący, 135  
 Matrox, 12  
 menu kontekstowe, 54  
 mieszanie kolorów, 266, 269, 276, 277, 284, 285, 303, 376  
 dwuźródłowe, 280  
 minification filter, *Patrz:* filtr  
 pomniejszający  
 mipmapa, 168, 174, 179, 187, 192, 196, 241, 530  
 poziom, 258  
 współczynnik skalowania,  
 179  
 MLLA, 295  
 model  
 materiału, 121  
 oświetlenia, 120, 121  
 Blinna-Phonga, 135, 136, 137, 145, 285, 420, 421, 422  
 Lamberta, 130, 131, 133, 420, 422  
 Phonga, 135, 152, 153, 154, 335, 420, 422  
 przestrzeni barw RGB,  
*Patrz:* RGB  
 model matrix, *Patrz:* macierz  
 modelu

model-view matrix, *Patrz:*  
 macierz modelu-widoku  
 monitora synchronizacja  
 pionowa, 291  
 morphological anti-aliasing,  
*Patrz:* MLLAA  
 motion blur, *Patrz:* rozmycie  
 ruchu  
 MRT, 246, 250, 254  
 MSAA, 295, 296, 297, 299,  
 307, 309  
 multiple render targets,  
*Patrz:* MRT  
 multisample anti-aliasing,  
*Patrz:* MSAA  
 multisample buffer, *Patrz:*  
 bufor wielopróbkowania  
 multisample texture, *Patrz:*  
 tekstura wielopróbkowania  
 multisampling, *Patrz:* MSAA,  
 wielopróbkowanie

## N

nadpróbkowanie, 295, 312  
 named interface block, *Patrz:*  
 blok interfejsu nazwany  
 named uniform block, *Patrz:*  
 blok jednorodny nazwany,  
*Patrz:* blok jednorodny  
 nazwany  
 Native Platform Graphics  
 Interface, *Patrz:* EGL  
 Newell Martin, 137, 349  
 Next Generation,  
*Patrz:* OpenGL NG  
 nieciągłość, *Patrz:* szew  
 normal map, *Patrz:* mapa  
 wektorów normalnych  
 normal mapping, *Patrz:*  
 mapowanie wektorów  
 normalnych  
 normalized device coordinates,  
*Patrz:* współrzędne  
 znormalizowane urządzenia  
 NVIDIA, 12

## O

obiekt, 23  
 bufora, 23  
 biejący, 24  
 GL\_TEXTURE\_BUFFER,  
 228  
 indeksowany, 118  
 jednorodnego,  
*Patrz:* UBO  
 ładowanie danych, 24  
 odczytu pikseli, 163  
 ramki, *Patrz:* FBO  
 renderingu, *Patrz:* RBO  
 transformacji  
 sprzężonych zwrotnie,  
 360  
 tworzenie, 23, 24  
 usuwanie, 24  
 wierzchołków,  
*Patrz:* VBO,  
*Patrz:* obiekt VBO  
 identyfikator, 23  
 programu, 40  
 konsolidacja, 43  
 parametry, 44  
 tworzenie, 42, 59  
 usuwanie, 42  
 próbkowania tekstury, 180,  
 181, 182  
 rzutowanie, *Patrz:*  
 rzutowanie  
 selekcja, 97  
 shadera, 40  
 kod źródłowy, 41  
 parametry, 44  
 tworzenie, 40  
 usuwanie, 41  
 synchronizacji, 19  
 tablic wierzchołków, 30  
 tekstury, 175  
 transformacja,  
*Patrz:* transformacja  
 transformacji sprzężonych  
 zwrotnie, 361

VBO, 24  
 współrzędne,  
*Patrz:* współrzędne  
 zmienna stanu,  
*Patrz:* zmienna stanu  
 object coordinates,  
*Patrz:* obiekt współrzędne  
 obraz  
 dwuskładnikowy, 216  
 monochromatyczny, 216  
 o wysokim zakresie jasności  
 HDR, 216  
 obrót, 77  
 obserwator, 89, 128  
 obsługa błędów, 20  
 obszar renderingu, 35, 36  
 modyfikacja, 63  
 rozmiar, 35  
 occlusion query,  
*Patrz:* zasłaniania  
 odbicie  
 rozproszone, 135  
 zwierciadlane światła,  
*Patrz:* światło  
 zwierciadlane  
 odcinek, 30, 224  
 odwzorowanie  
 nierówności powierzchni,  
 415  
 środowiska,  
*Patrz:* środowisko  
 odwzorowanie  
 off-screen rendering, *Patrz:*  
 rendering pozaekranowy  
 okluzja otoczenia, 369  
 okna rozmiar, 55  
 OpenGL, 11, 17  
 implementacja,  
*Patrz:* implementacja  
 rozszerzenie, 12, 449  
 funkcja, 450  
 nazwa, 450  
 obsługa, 451  
 specyfikacja, 451  
 stała, 450

- specyfikacja, 18, 19
  - wersja, 12, 13, 14, 15
  - OpenGL ES, 12, 14, 15, 17, 21, 217
    - rozszerzenie, 450
  - OpenGL Extension to the X Window System, *Patrz:* GLX
  - OpenGL Mathematics, *Patrz:* GLM
  - OpenGL NG, 21
  - OpenVG, 17
  - operator, 493
  - oświetlenie, 105, *Patrz też:* światło
    - generowane
      - na fragment, 140, 143
      - na wierzchołek, 140, 141, 142
    - model, *Patrz:* model oświetlenia
  - oversampling, *Patrz:* SSAA, nadpróbkowanie
- P**
- packed pixel, *Patrz:* piksel
    - typ upakowany
  - paraboloida, 356
  - particle system, *Patrz:* system cząstek
  - pasek
    - czworokątów, 324
    - trójkątów, 30, 31
  - pasma Macha, 107, 142
  - patch vertices, *Patrz:* płat wierzchołków
  - PCF, *Patrz:* filtr PCF
  - percentage closer filtering, *Patrz:* filtr PCF
  - perspective division, *Patrz:* dzielenie perspektywiczne
  - pick ray, *Patrz:* promień przecięcia
  - piksel, 165
    - poberanie z bufora ramki, 251
  - tablica, *Patrz:* tablica pikseli
    - typ, 166
    - w buforze ramki, 232
  - pixel unpack buffer object, *Patrz:* obiekt bufora odczytu
  - pikseli
  - plik
    - blinn\_phong.glsl, 137
    - GL/gl.h, 450
    - GL/glccorearb.h, 450
    - GL/gltext.h, 450
    - GL/glx.h, 450
    - GL/glxext.h, 450
    - GL/wgl.h, 450
    - GL/wgltext.h, 450
    - graficzny, 188
    - lambert\_light.glsl, 131
    - materials.glsl, 122
    - materials\_static.glsl, 122
    - nagłówkowy, 450
    - newell\_teacup.h, 137, 349
    - newell\_teapot.h, 137, 349
    - newell\_teaspoon.h, 137, 349
    - shaders.cpp, 58, 59
  - plaszczyczna, 77
    - obcinania, 32, 77
      - definiowana przez użytkownika, 91, 93
      - dodatkowa, 91, 93
    - styczna, 417
    - usuwania, 95
  - płat wierzchołków, 319, 326, 346
    - wejściowy, 319
    - wyjściowy, 319, 320
  - podprogram, 155, 518
  - podsystem aplikacji, 51
  - point light, *Patrz:* światło punktowe
  - point sprite, *Patrz:* sprajt punktowy
  - position, 124
  - post processing, *Patrz:* przetwarzanie końcowe
  - potok programów, 40
  - potok renderingu, *Patrz:* rendering potok
  - powierzchnia
    - Béziera, 137, 331, 348
    - płat, 348, 354
    - punkt kontrolny, 348
    - rendering, 349
    - stopień, 348
    - teselacja, 355
    - współrzędne, 348
    - drugiego stopnia, 356
  - prawo
    - cosinusów, *Patrz:* prawo Lamberta
    - Lamberta, 130
    - załamania Snella, 442
  - preprocesor, 49
  - primitive clipping, *Patrz:* prymityw obcinanie
  - profil
    - kompatybilny, 14
    - podstawowy, 14
  - profilowanie kontekstu renderingu, 13
  - program, 40
    - konsolowy, 51
    - okienkowy, 51
    - potok, *Patrz:* potok programów
    - punkt startowy, 51
  - program object, *Patrz:* obiekt programu
  - program pipeline object, *Patrz:* obiekt:potoku programów
  - projected planar shadow, *Patrz:* cień rzutowanie
  - projection matrix, *Patrz:* macierz rzutowania
  - promień przecięcia, 100
  - próbka, 296
    - kolor, 296
    - maska pokrycia, 302
    - wartość pokrycia, 296, 301, 302, 303, 304

## prymityw

- generator, *Patrz:* teselator
- geometryczny, 15, 26
- łamana, *Patrz:* lamana
- odcinek, *Patrz:* odcinek
- pasek trójkątów,
  - Patrz:* pasek trójkątów
- punkt, *Patrz:* punkt
- trójkąt, *Patrz:* trójkąt
- wachlarz trójkątów,
  - Patrz:* wachlarz trójkątów
- graficzny, 14
- obcinanie, 35
- rozszerzony, 31, 313
  - generowanie indeksów, 382

## prymityw

- łamana, 314
- odcinek, 313
- pasek trójkątów, 314
- trójkąt, 314
- teselacja, *Patrz:* teselacja
- wejściowy, 314, 315, 316

## przestrzeń

- lokalna styczna, 417
- nazw, 496

przesunięcie, *Patrz:* translacja

- wartości głębokości, 274

## przetwarzanie końcowe, 241

## przezroczystość, 280, 284

## ptrbits, 18, 19

## punkt, 30, 224

**R**

## rasteryzacja, 15, 105, 106

## RBO, 232, 234, 235, 236, 237, 238

- przyłączanie do FBO, 237
- tworzenie, 234

## refrakcja światła,

*Patrz:* światło refrakcja

render to texture, *Patrz:*

- rendering do tekstury
- renderbuffer object,
  - Patrz:* RBO

## rendering, 13, 30, 52, 55, 63

- do tekstury, 241, 242
- instancyjny, 282
- kontekst, 16, 51
  - monoskopowy, 232
  - utrata, 20
- liczba ramek na sekundę,
  - Patrz:* FPS
- obszar, *Patrz:* obszar
- renderingu
- potok, 15, 16, 106, 107
- pozaekranowy, 237, 238, 241, 242
- profil, *Patrz:* profil
- punktów, 223
- sceny, 62
- tekstu, 112
- tekstury, 196
- warstwowy, 241, 260
- warunkowy, 292, 294
- wieloinstancyjny, *Patrz:*
  - geometria instancyjna
- wielowarstwowy, 258, 261, 262, 263, 264
- wskazówki, 220
- wydajność, 216
- rendering hint, *Patrz:*
  - rendering wskazówki
- RGB, 105, 165, 168, 222
  - sRGB, *Patrz:* sRGB
- RGBA, 165, 171, 180, 303
- rozdzielczość, 179
- rozmycie ruchu, 260
- równanie
  - kanoniczne elipsoidy, 356
  - mieszania, 276, 277, 303
  - oświetlenia, 121, 131, 135, 153
  - parametryczne elipsoidy, 356
  - wektorowe, 348
- rzutowanie, 69, 77
  - cieni, *Patrz:* cień
  - rzutowanie
    - perspektywiczne, 85, 86
    - prostokątne, 77, 80

**S**sample, *Patrz:* próbkasample coverage, *Patrz:* próbka

wartość pokrycia

## scena

3D, 32, 35

rendering, *Patrz:* rendering sceny

seam, *Patrz:* szewseamless filter, *Patrz:* filtr

bezszwowy

## sfera, 356

## shader, 13, 40

ewaluacji teselacji, 14, 34, 319, 320, 321, 322, 325, 335, 337, 339, 355, 360, 362, 427, 429

fragmentów, 13, 38, 231, 241, 266, 534

kontrola przepływu, 519

geometrii, 14, 34, 36, 241, 243, 260, 262, 314, 315, 316, 345, 360, 362, 363, 432, 536

kontroli teselacji, 14, 34, 319, 320, 325, 335, 339

## przechowywanie, 58

wierzchołków, 13, 34, 38, 57, 65, 325, 360, 362

zmienna wejściowa,
 

- Patrz:* zmienna wejściowa

zmienna wyjściowa,
 

- Patrz:* zmienna wyjściowa

shader object, *Patrz:* obiekt

shadera

shader storage block, *Patrz:* blok

pamięci shadera

shadow, *Patrz:* cieńshadow map, *Patrz:* mapa cieni

## shadow mapping,

*Patrz:* mapowanie cieni



shadow volume,  
*Patrz:* bryła cieni  
 shininess, 121  
 silhouette edge, *Patrz:*  
 krawędź szkieletowa  
 skalowanie, 76  
 skybox, 213, 439  
 specular, 121, 124  
 specular light, *Patrz:* światło  
 zwierciadlane  
 spot light, *Patrz:* światło  
 reflektorowe  
 sprajt  
 animacja, 318  
 generowany przez shader  
 geometrii, 318  
 punktowy, 223, 224, 225,  
 318  
 sRGB, 105, 222  
 SSAA, 295  
 ST, 168  
 stacja graficzna IRIS, 11  
 stała  
 GL\_ELEMENT\_ARRAY\_  
 BUFFER, 66  
 GL\_LINE\_LOOP, 30  
 GL\_LINE\_STRIP, 30  
 GL\_LINES, 30  
 GL\_PATCHES, 319  
 GL\_POINTS, 30  
 GL\_TRIANGLE\_FAN, 30  
 GL\_TRIANGLE\_STRIP,  
 30  
 GL\_TRIANGLES, 30  
 określona przez  
 rozszerzenie, 450  
 wbudowana, 523  
 state machine, *Patrz:* maszyna  
 stanów  
 stencil buffer, *Patrz:* bufor  
 szablonu  
 stencil test, *Patrz:* test  
 szablonu  
 storage qualifier, *Patrz:*  
 kwalifikator przechowywania

stos, 20  
 STPQ, 168  
 STRQ, 168  
 struktura, 489, 490, 493  
 element, 490  
 inicjalizacja, 490  
 subroutine, *Patrz:* podprogram  
 subroutine uniform variable,  
*Patrz:* zmienna jednorodna  
 podprogramu  
 super sampling anti-aliasing,  
*Patrz:* SSAA  
 system  
 cząstek, 224, 319  
 funkcji iterowanych,  
*Patrz:* IFS  
 szew, 213, 274

## Ś

środowiska odwzorowanie, 212,  
 272, 415, 431  
 świat, 270, 271  
 światło, *Patrz też:* oświetlenie  
 dyspersja, *Patrz:* dyspersja  
 kierunkowe, 124, 154, 285  
 implementacja, 131, 136,  
 153  
 odbicie, 432, 433, 435  
 kąta padania, 432  
 otoczenia, 121, 124  
 globalne, 128  
 lokalne, 128  
 współczynnik odbicia, 121  
 punktowe, 124, 145, 150  
 implementacja, 131, 133,  
 137  
 współczynnik tłumienia,  
 125  
 reflektorowe, 124  
 refrakcja, 442, 445  
 rozproszone, 121, 124  
 współczynnik odbicia, 121  
 rozszczepienie  
 chromatyczne,  
*Patrz:* dyspersja

stożek, 124  
 współczynnik odbicia, 121  
 zwierciadlane, 121, 124, 135  
 współczynnik odbicia,  
 121  
 źródło, *Patrz:* źródło  
 światła

## T

tablica, 19, 491, 494  
 deklarowanie, 491  
 indeksów wierzchołków, 66  
 inicjalizacja, 492  
 pikseli, 163  
 format danych, 165  
 prostokątna, 163  
 tablica  
 rozmiar, 492  
 tekstur, 241  
 dwuwymiarowych, 169,  
 170, 171, 208, 209  
 jednowymiarowych, 168,  
 170  
 sześciennych, 169, 171  
 trójwymiarowych, 208  
 wierzchołków, *Patrz:* VA  
 tangent vector, *Patrz:* wektor  
 styczn  
 tekssel, 168  
 brzegowy, 183  
 filtracja, 168  
 tekstury, 183  
 wartości składowe, 180  
 tekst rendering, *Patrz:*  
 rendering tekstu  
 tekstura, 14, 163, 168, 232,  
 241, 303, 529  
 atlas, *Patrz:* atlas tekstury  
 brzeg, 177  
 buforowa, 169, 170, 228, 230  
 definiowanie, 184, 185, 187  
 dwuwymiarowa, 168, 169,  
 182, 184, 185, 195, 241  
 tworzenie, 188

- tekstura
- filtrowanie, 177, 178, 179, 190, 196, 198
  - głębokości, 392, 393, 395
  - jako bufor renderingu, 241
  - jednowymiarowa, 168, 169, 182, 241
  - kompresja, 14, 216, 217, 220, *Patrz też:* kompresja
  - kubiczna, *Patrz:* tekstura sześcienna
  - mapowanie, *Patrz:* mapowanie
  - o stałym formacie, 186
  - obiekt, *Patrz:* obiekt tekstury
  - obraz, 489, 533
  - prostokątna, 169, 170
  - próbkowanie, 168, 177, 178, 179, 180, 181, 182, 190, 530
  - rendering, *Patrz:* rendering tekstury
  - szczegółowość, 179
  - sześcienna, 169, 170, 212, 213, 214, 241, 258, 432, 439
  - mapowanie, 213
  - trójwymiarowa, 168, 169, 182, 203, 241, 258
  - typ danych, 171
  - uchwyty, 168, 489
  - wielopróbkowania, 308, 309, 310
  - wielowarstwowa, 259
  - współrzędne, 168, 169, 178, 182, 190
  - znormalizowane, 183
  - zawijanie, 178, 195
- teselacja, 14, 34, 319, 321, 322
- czworokątów
    - na trójkąty, 321, 323, 324, 326
    - na zbiór linii
      - konturowych, 321, 329
  - elipsoidy, *Patrz:* elipsoida
  - teselacja
  - ewaluacja, 14, 34
  - interpolacja atrybutów
    - wierzchołków, 330, 331, 333
    - algorytm PN Triangles, *Patrz:* algorytm PN Triangles
  - algorytm teselacji
    - Phonga, *Patrz:* algorytm teselacji
    - Phonga
  - kontrola, 14, 34
  - Phonga, 333, 334, 335, 339
  - teselator, 319, 321, 330, *Patrz też:* teselacja
  - tessellation, *Patrz:* teselacja
  - test
    - bufora głębokości, 266, 272, 273, 286
    - bufora szablonu, 379
    - cienia, 401
    - MSAA, 297, 299
    - nożycowy, 266
    - składowej alfa, 207, 303, 304
    - szablonu, 266, 267
    - własności piksela, 266
    - zasłaniania, 266, 285, 287, 288, 292, 294
  - texel, *Patrz:* tekssel
  - texture buffer, *Patrz:* tekstura buforowa
  - texture coordinates, *Patrz:* tekstura współrzędne
  - texture handle, *Patrz:* tekstura uchwyt
  - texture mapping, *Patrz:* mapowanie
  - texture object, *Patrz:* obiekt tekstury
  - texture sampler object, *Patrz:* obiekt próbkowania tekstury
  - texture sampling, *Patrz:* tekstura próbkowanie
  - texture unit, *Patrz:* jednostka teksturująca
  - tomografia komputerowa, 207
  - transform feedback, *Patrz:* transformacja sprzężona zwrotnie
  - transformacja, 69
    - obrót, *Patrz:* obrót obszaru renderingu, 35
    - przesunięcie, *Patrz:* translacja
    - skalowanie, *Patrz:* skalowanie
    - sprzężona zwrotnie, 15, 360, 361, 362
  - translacja, 75
  - trójkąt, 30, 31
    - orientacja, 31
    - pasek, *Patrz:* pasek trójkątów
    - wachlarz, *Patrz:* wachlarz trójkątów
  - typ, 171, 479
    - bool, 484
    - całkowity, 484
    - danych w języku, 18
    - GLbitfield, 19
    - GLboolean, 19
    - GLbyte, 19, 28
    - GLchar, 19
    - GLdouble, 19, 28
    - GLenum, 19
    - GLfixed, 19, 28
    - GLfloat, 19, 28
    - GLhalf, 19, 28
    - GLint, 18, 19, 28
    - GLint64, 19
    - GLintptr, 18, 19
    - GLshort, 19, 28
    - GLsizei, 19
    - GLsizei\_ptr, 18, 19
    - GLsync, 18, 19
    - GLubyte, 19, 28
    - GLuint, 19, 28

GLuint64, 19  
 GLushort, 19, 28  
 GLvoid, 18  
 konwersja, 482  
 logiczny, 19  
 macierzowy, 487  
 void, 483  
 wektorowy, 485  
 zmiennoprzecinkowy, 485

## U

UBO, 113, 118, 119, 146  
 format danych, 113, 114  
 układ współrzędnych, 73  
 przekształcanie, 154  
 przestrzeni stycznej, 420  
 uniform buffer object,  
*Patrz:* UBO  
 uniform variable, *Patrz:*  
 zmienna jednorodna,  
*Patrz:* zmienna jednorodna  
 user clipping plane, *Patrz:*  
 płaszczyzna obcinania  
 definiowana przez  
 użytkownika  
 UV, 168  
 UV mapping, *Patrz:*  
 mapowanie UV

## V

VA, 27  
 VAO, 30, 61, 66  
 variance shadow map, *Patrz:*  
 mapa cienia wariacyjna  
 VBO, 27, 28, 61, 129  
 Vertex Array Objects,  
*Patrz:* VAO  
 vertex arrays, *Patrz:* VA  
 vertex shader, *Patrz:* shader  
 wierzchołków

vertical synchronization, *Patrz:*  
 monitor synchronizacja  
 pionowa  
 view matrix, *Patrz:* macierz  
 widoku  
 viewport, *Patrz:* obszar  
 renderingu  
 viewport transformation,  
*Patrz:* transformacja  
 obszaru renderingu  
 VSM, *Patrz:* mapa cienia  
 wariacyjna  
 Vulkan, 21

## W

wachlarz trójkątów, 30, 31  
 warstwa, 258, 260  
 Web Graphics Library,  
*Patrz:* WebGL  
 WebGL, 21  
 wektor, 75, 468, 485, 486, 494  
 binormalny, 349  
 bistyczny, 417, 418, 419  
 długość, 468, 469  
 dodawanie, 469  
 iloczyn  
 skalarny, 470  
 wektorowy, 471  
 jednostkowy, 470  
 kierunek, 468  
 kierunku światła, 139  
 mnożenie, 495  
 normalnej, 28  
 normalny, 113, 129, 131,  
 349, 415, 419, 420, 426  
 generowanie, 137  
 transformacja, 130, 139  
 uśredniony, 129, 130  
 odbicia, 153  
 odejmowanie, 469  
 połówkowy, 135, 153, 420  
 porównywanie, 528

styczny, 349, 417, 418, 419  
 zerowy, 469  
 zwrot, 468  
 WGL, 17, 232, 450  
 wielopróbkowanie, 266, 295,  
 296, 297, 299, 303, 531  
 bufor, *Patrz:* bufor  
 wielopróbkowania  
 tekstura, *Patrz:* tekstura  
 wielopróbkowania  
 w FBO, 307  
 wierzchołek, 13, 15, 26  
 atrybut, 26, 28  
 numer indeksu, 27, 65, 66  
 główny, 106  
 kolejność, 31, *Patrz też:*  
 trójkąt orientacja  
 płąt, *Patrz:* płąt  
 wierzchołków  
 przyległy, 313  
 tablica, *Patrz:* VA  
 tablica indeksów, 66  
 współrzędne, 28, 56, 57,  
*Patrz też:* współrzędne  
 transformacja, 32  
 Wiggle, *Patrz:* WGL  
 window coordinates,  
*Patrz:* współrzędne okna  
 współrzędne  
 barycentryczne, 98, 99, 321,  
 328  
 jednorodne, 32, 33  
 obcinania, 34  
 obserwatora, 34  
 oka, 34  
 okna, 35  
 transformacja, 32, 33  
 znormalizowane  
 urządzenia, 34, 35  
 wybieranie kolorów, 254  
 wydajność, 25  
 wyrażenie, 493  
 wzór Laplace'a, 466

**Z**

## zapytanie

- asynchroniczne, 285, 286, 287, 288

- zasłaniania, *Patrz:* test zasłaniania

## zmienna, 478

- dwustanowa, 18

- gl\_InstanceID, 38

- gl\_Layer, 260, 262

- gl\_Position, 34, 38

- gl\_VertexID, 38

- inPosition, 57

- interpolowana, 106, 107

- jednorodna, 37, 69, 71, 73, 113, 116

- inicjalizacja, 69

- macierz, 72

- podprogramu, 155

- położenie, 70, 71

- klienta OpenGL, 18

- przypisanie wartości, 493

- serwera OpenGL, 18

- stanu, 18, 23

- GL\_UNIFORM\_

- BUFFER\_OFFSET\_

- ALIGNMENT, 120

- GL\_UNIFORM\_

- MATRIX\_STRIDE, 115

- GL\_UNIFORM\_OFFSET,

- 114

- odczyt, 18

- wbudowana, 38, 39, 115,

- 520, 521, 522`

- jednorodna, 523

- wejściowa, 38, 39, 497

- interpolacja, 39

- wektorowa, 486

- wyjściowa, 38, 39, 106, 497

- zakres widoczności, 496

**Ż**

- źródło światła, 120, 124

- położenie, 150

- zmiana kierunku, 139

# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄZKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

**Darmowa biblioteka OpenGL** jest świetnym narzędziem do tworzenia grafiki trójwymiarowej na różnych platformach sprzętowych i w różnych systemach operacyjnych. Jej legendarne możliwości, doskonałe od 1992 roku do dziś, znakomicie sprawdzają się na przykład w grach komputerowych. Z kolei otwarty interfejs programistyczny pozwolił na zbudowanie wokół niej mnóstwa narzędzi uzupełniających i rozszerzeń, które przydadzą się każdemu użytkownikowi pragnącemu wykreować swój własny świat w formacie 3D.

**Już niedługo zobaczysz** na ekranie to, co na razie pozostaje jedynie w świecie Twojej wyobraźni! Ta książka omawia budowę biblioteki OpenGL, zasady pisania programów i wykorzystania obiektów. Nauczysz się programować grafikę — przekształcać scenę, budować modele oświetlenia i obsługiwać tekstury. Poznasz różne rodzaje buforów i zobaczysz, na czym polega antyaliasing. Odkryjesz też, jak przetwarzać geometrię obiektów i jak tworzyć realistyczne cienie. W dodatkach do książki znajdziesz informacje, które pomogą Ci rozpocząć pracę z tą biblioteką. Poznaj OpenGL i zachwyć świat swoimi grafikami!

- Co to jest OpenGL?
- Historia i perspektywy OpenGL
- Pierwszy program
- Rzutowanie i transformacje
- Cieniowanie i oświetlenie
- Tekstury i antyaliasing
- Zaawansowane przekształcanie geometrii
- Cienie
- Odwzorowanie środowiska i nierówności powierzchni
- Rozszerzenia
- Macierze i wektory
- Język GLSL

**Wykreuj swój świat z OpenGL!**

**Helion**

31685 numer katalogowy

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Sprawdź najnowsze promocje:

- <http://helion.pl/promocje>
- Książki najchętniej czytane:
- <http://helion.pl/bestsellery>
- Zamów informacje o nowościach:
- <http://helion.pl/nowosci>

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

sięgnij po **WIĘCEJ**



**KOD KORZYŚCI**

ISBN 978-83-283-0193-1



9 788328 301931

Informatyka w najlepszym wydaniu

cena: 89,00 zł