

Spis treści

Rozdział 1. Wstęp	9
1.1. Do kogo jest skierowana ta książka?	10
1.2. Przydatne narzędzia	11
Rozdział 2. Zrozumieć GPU	13
2.1. Co to jest Shader?	13
2.2. Architektura GPU	14
2.2.1. GPU versus CPU	15
2.2.2. Jednostki wykonawcze GPU	16
2.2.3. Przelączanie kontekstu i unikanie opóźnień	19
2.2.4. Przetwarzanie rozgałęzień	20
2.2.5. Model pamięci	22
Rozdział 3. Potok renderujący OpenGL	29
3.1. Najważniejsze etapy potoku graficznego	29
3.1.1. Przetwarzanie geometrii	29
3.1.2. Rasteryzacja	30
3.1.3. Przetwarzanie fragmentów	31
3.1.4. Postprocess fragmentów	32
3.2. Wprowadzenie do programowalnego potoku	32
3.2.1. Shader wierzchołków	33
3.2.2. Teselacja	34
3.2.3. Shader geometrii	37
3.2.4. Shader fragmentów	38
3.3. Kompilacja	39
3.3.1. Proces kompilacji, wiązania i linkowania	39
3.3.2. Wielokrotne wiązanie shaderów tego samego typu	42
3.3.3. Rozłączne programy	43
3.3.4. Status kompilacji	46
Rozdział 4. Podstawy programowania	49
4.1. Język programowania shaderów GLSL	49
4.2. Profile	50
4.3. Interpretacja schematów konstrukcji programistycznych	51
4.4. Nazwy identyfikatorów obiektów	52
4.5. Preprocesor	52
4.5.1. Kontrola wersji shadera (#version)	53

4.5.2. Definiowanie symboli oraz makrodefinicji (<code>#define</code> , <code>#undef</code>)	54
4.5.3. Kontrola warunkowej kompilacji (<code>#if</code> , <code>#ifdef</code> , <code>#ifndef</code> , <code>#elif</code> , <code>#else</code> , <code>#endif</code>)	58
4.5.4. Wspomaganie warunkowej kompilacji (<code>#error</code>)	59
4.5.5. Wspomaganie diagnostyki kodu źródłowego (<code>#line</code>)	59
4.5.6. Sterowanie działaniem kompilatora (<code>#pragma</code>)	60
4.5.7. Zarządzanie zestawem rozszerzeń języka GLSL (<code>#extension</code>)	61
4.6. Typy danych	64
4.6.1. Bazowe typy numeryczne – skalary	65
4.6.2. Pochodne typy numeryczne – wektory	70
4.6.3. Pochodne typy numeryczne – macierze	76
4.6.4. Typy uchwytów	83
4.6.5. Typ subroutine	83
4.6.6. Struktury	84
4.6.7. Tablice	86
4.7. Zmienne	92
4.7.1. Zmienne wewnętrzne	93
4.7.2. Zmienne interfejsu	94
4.7.3. Blok interfejsu	96
4.7.4. Deklaracja obiektów użytkownika w modułach shadera	98
4.8. Zakres zmiennych	99
4.9. Operatory	101
4.10. Instrukcje kontroli przepływu	102
4.11. Funkcje	104
4.11.1. Deklaracja funkcji	105
4.11.2. Definicja funkcji	106
4.11.3. Przetłumaczenie funkcji	106
4.11.4. Parametry funkcji i wartości zwracane	107
Rozdział 5. Dane	112
5.1. Generyczny magazyn danych (obiekt bufora)	113
5.1.1. Tworzenie buforów	114
5.1.2. Wiązanie buforów	114
5.1.3. Zarządzanie stanem obiektów buforowych	117
5.1.4. Swobodny dostęp do danych bufora	122
5.1.5. Kopiowanie buforów	124
5.1.6. Odczytywanie zawartości buforów	124
5.1.7. Usuwanie buforów	125
5.2. Zmienne oraz bloki <i>uniform</i>	125
5.2.1. Domyślny blok <i>uniform</i>	126
5.2.2. Nazwany blok <i>uniform</i>	133
5.3. Zmienne oraz bloki <i>buffer</i>	147
5.3.1. Blok buforowy	148
5.3.2. Kontrola dostępu do pamięci	151
5.3.3. Operacje atomowe na zmiennych buforowych	155
5.3.4. Organizacja danych w bloku	157
5.3.5. Własności stanu zmiennych oraz bloków buforowych	158
5.3.6. Pozyskiwanie lokacji zmiennych buforowych oraz aktualizacja danych	159
5.3.7. Wiązanie bloku buforowego	159
5.4. Sformatowany magazyn danych (obiekt tekstury)	160
5.4.1. Reprezentacja tekstur w OpenGL	161

5.4.2. Struktura magazynu danych	161
5.4.3. Tworzenie oraz usuwanie tekstur	164
5.4.4. Wiązanie tekstur	165
5.4.5. Alokacja oraz aktualizacja magazynu danych dla tekstur	169
5.4.6. Tekstura buforowa	171
5.5. Tekstury w shaderach	173
5.5.1. Mechanizm teksturowania	174
5.5.2. Zmienne <i>sampler</i>	177
5.5.3. Podstawowa metoda dostępu do złożonych typów tekstur	181
5.5.4. Funkcje wbudowane odpytывania tekstur	189
5.5.5. Zaawansowane funkcje wbudowane dostępu do danych tekstury	190
5.6. Obrazy w shaderach	196
5.6.1. Zmienne <i>image</i>	197
5.6.2. Podstawowe operacje na obrazie	202
5.6.3. Operacje atomowe na obrazie	204
5.7. Liczniki atomowe	207
5.7.1. Tworzenie liczników	207
5.7.2. Własności stanu liczników atomowych	208
5.7.3. Wiązanie buforów z licznikami	208
5.7.4. Operacje atomowe	209
5.8. Dodatkowe metody synchronizacji w dostępie do danych	210
5.8.1. Synchronizacja dostępu w shaderach	210
5.8.2. Synchronizacja dostępu w API	212
Rozdział 6. Programowanie potoku renderującego	213
6.1. Przykładowy program zawierający wszystkie podstawowe shadery	213
6.2. Ogólny obraz komunikacji międzyetapowej	218
6.3. Przekazywanie danych w potoku	220
6.3.1. Atrybuty shadera wierzchołków	221
6.3.2. Interfejsy in/out między etapami	223
6.3.3. Lokacje przy przekazywaniu danych między shaderami	228
6.3.4. Pełne a częściowe dopasowanie	230
6.3.5. Komponenty w lokacjach	231
6.3.6. Sposoby interpolacji przy przekazywaniu danych do shadera fragmentów	232
6.3.7. Wbudowany blok <i>gl_PerVertex</i>	236
6.4. Przebieg i własności teselacji	241
6.4.1. Deklaracja płatu i jego przekształcenie na właściwy prymityw poddawany teselacji	242
6.4.2. Stopnie teselacji	243
6.4.3. Opcje rozstawu	245
6.4.4. Teselacja trójkąta	246
6.4.5. Teselacja czworokąta	250
6.4.6. Teselacja izolinii	252
6.5. Programowanie shadera wierzchołków	253
6.5.1. Optymalizacja liczby wywołań	254
6.5.2. Zmienne wbudowane	255
6.6. Programowanie shadera kontroli teselacji	256
6.6.1. Przepływ danych i deklaracja liczby wywołań	257
6.6.2. Współbieżny dostęp do danych wyjściowych	259
6.6.3. Zmienne wbudowane	261

6.7. Programowanie shadera ewaluacji teselacji.	261
6.7.1. Przepływ danych.	262
6.7.2. Konfiguracja prymitywów za pomocą wejściowego kwalifikatora layout	263
6.7.3. Zmienne wbudowane	264
6.8. Programowanie shadera geometrii.	264
6.8.1. Interfejs wejścia i deklaracja liczby wywołań shadera.	265
6.8.2. Interfejs wyjścia – deklaracja prymitywu i emisja wierzchołków.	266
6.8.3. Dedykowane prymitywy przylegające	269
6.8.4. Zmienne wbudowane	272
6.9. Programowanie shadera fragmentów	272
6.9.1. Renderowanie do bufora ramki	273
6.9.2. Odrzucanie fragmentów	274
6.9.3. Modyfikacja współrzędnych fragmentów	275
6.9.4. Wczesny test fragmentów i modyfikacja bufora głębokości.	275
6.9.5. Funkcje wbudowane i wywołania wspomagające	278
6.9.6. Zmienne wbudowane	282
Rozdział 7. Mechanizmy uzupełniające	284
7.1. Renderowanie do tekstur	284
7.1.1. Przygotowanie aplikacji	284
7.1.2. Renderowanie do wielu tekstur jako osobnych załączników koloru	286
7.1.3. Renderowanie do tekstur złożonych z wykorzystaniem shadera geometrii.	289
7.2. Mechanizm Shader Subroutine.	291
7.2.1. Funkcje wywoływane statycznie i dynamicznie	292
7.2.2. Elementy składniowe mechanizmu	294
7.2.3. Przykładowa implementacja	297
7.2.4. Konfigurowanie powiązań zmiennych z funkcjami subroutine.	299
Rozdział 8. Shader obliczeniowy	303
8.1. Wprowadzenie	303
8.1.1. Kompilacja i użycie shadera obliczeniowego.	304
8.2. Wywołania shadera obliczeniowego i grupy wykonawcze	305
8.2.1. Identyfikacja wywołań.	306
8.2.2. Ograniczenia liczby wywołań.	307
8.3. Charakterystyka przetwarzania	308
8.3.1. Przetwarzanie lokalnych grup roboczych	308
8.3.2. Pamięć współdzielona – kwalifikator shared.	309
8.3.3. Synchronizacja	310
Dodatek	313
Dodatek A.	313
Dodatek B.	314
Dodatek C	315
Dodatek D	317
Dodatek E	326
Dodatek F	335
Słownik pojęć	341
Bibliografia	343