Wykład 2

OpenGL i GLSL

Notatki do wykładu

Grafika Komputerowa (GRK)

dr Wojtek Palubicki

Adam Mickiewicz University

Lektura uzupełniająca: Real-Time Rendering, Rozdz. 3 (The Graphics Processing Unit)

1. Wprowadzenie do OpenGL

OpenGL (Open Graphics Library) to wieloplatformowe API do renderowania grafiki 2D i 3D. Działa jako interfejs między aplikacją a sterownikiem GPU. OpenGL operuje jako maszyna stanów — ustawiamy różne parametry (stan), a następnie wydajemy polecenia rysowania.

Współczesne wersje OpenGL (3.3+) korzystają z tzw. rdzeniowego profilu (Core Profile), który wymaga od programisty jawnego definiowania shaderów i zarządzania buforami danych.

> *Lektura: Real-Time Rendering, 3rd Ed., Rozdział 3.1 — GPU Pipeline Overview*

2. Architektura programu OpenGL

2.1 Kontekst i okno

Przed rozpoczęciem renderowania należy utworzyć kontekst OpenGL oraz okno do wyświetlania. Najczęściej używa się bibliotek pomocniczych takich jak GLFW (zarządzanie oknem) i GLEW lub GLAD (ładowanie rozszerzeń OpenGL).

2.2 Bufory wierzchołków (VBO)

Vertex Buffer Object (VBO) to bufor przechowujący dane wierzchołków w pamięci GPU. Dane te mogą zawierać pozycje, normalne, współrzędne tekstur i kolory. Przesłanie danych do VBO eliminuje konieczność ich wielokrotnego transferu z CPU do GPU.

2.3 Tablice wierzchołków (VAO)

Vertex Array Object (VAO) przechowuje konfigurację atrybutów wierzchołków — informację o tym, jak dane w VBO mają być interpretowane (rozmiar, typ, krok, przesunięcie). VAO pozwala szybko przełączać się między różnymi konfiguracjami geometrii.

3. Język GLSL

GLSL (OpenGL Shading Language) to język programowania shaderów. Składniowo przypomina C, ale zawiera wbudowane typy wektorowe i macierzowe oraz funkcje matematyczne przydatne w grafice.

3.1 Vertex Shader

Vertex shader jest wywoływany raz dla każdego wierzchołka. Jego głównym zadaniem jest przekształcenie pozycji wierzchołka z przestrzeni modelu do przestrzeni obcinania (clip space) poprzez pomnożenie przez macierz MVP (Model-View-Projection). Shader może również przekazywać dane (normalne, współrzędne tekstur) do następnych etapów potoku.

3.2 Fragment Shader

Fragment shader jest wywoływany raz dla każdego fragmentu (potencjalnego piksela). Otrzymuje interpolowane dane z vertex shadera i oblicza końcowy kolor piksela. Tu realizowane jest oświetlenie, teksturowanie i inne efekty wizualne.

3.3 Zmienne uniform

Zmienne uniform to dane przesyłane z CPU do shadera, które pozostają stałe dla wszystkich wierzchołków/fragmentów w danym wywołaniu rysowania. Typowe uniform to macierze transformacji, pozycje świateł i parametry materiału.

> *Lektura: Real-Time Rendering, 3rd Ed., Rozdział 3.2-3.4 — Programmable Shader Stage, API Evolution*

4. Główna pętla renderowania

Typowy program OpenGL działa w pętli: wyczyść bufor ramki, ustaw shadery i uniform, podłącz VAO, wywołaj funkcję rysowania (glDrawArrays lub glDrawElements), zamień bufory. Pętla powtarza się z częstotliwością odświeżania ekranu.

5. Podsumowanie

OpenGL i GLSL stanowią podstawowe narzędzia programowania grafiki czasu rzeczywistego. Znajomość zarządzania buforami, pisania shaderów i organizacji pętli renderowania jest niezbędna do tworzenia aplikacji graficznych.