Wykład 1

Potok Renderowania

Notatki do wykładu

Grafika Komputerowa (GRK)

dr Wojtek Palubicki

Adam Mickiewicz University

Lektura uzupełniająca: Real-Time Rendering, Rozdz. 2 (The Graphics Rendering Pipeline)

1. Wprowadzenie do potoku renderowania

Potok renderowania (ang. rendering pipeline) to sekwencja etapów, przez które przechodzą dane geometryczne, zanim zostaną wyświetlone na ekranie jako obraz 2D. Jest to podstawowy mechanizm współczesnej grafiki komputerowej czasu rzeczywistego, zaimplementowany zarówno sprzętowo (GPU) jak i programowo.

Głównym zadaniem potoku jest przekształcenie opisu trójwymiarowej sceny — składającej się z geometrii, materiałów, źródeł światła i kamery — w końcowy obraz rastrowy. Proces ten odbywa się w kilku następujących po sobie etapach.

> *Lektura: Real-Time Rendering, 3rd Ed., Rozdział 2.1 — Architecture*

2. Etapy potoku

2.1 Etap aplikacji (Application Stage)

Jest to etap realizowany całkowicie na CPU. Programista ma pełną kontrolę nad tym, co się dzieje. Typowe zadania obejmują: wykrywanie kolizji, obsługę wejścia użytkownika, obliczenia fizyki, sztuczną inteligencję i aktualizację animacji. Na końcu tego etapu generowane są prymitywy geometryczne (najczęściej trójkąty), które trafiają do następnego etapu.

- Detekcja kolizji i odpowiedź na kolizje
- Aktualizacja animacji i symulacji fizycznej
- Przygotowanie danych geometrycznych do wysłania na GPU

> *Lektura: Real-Time Rendering, 3rd Ed., Rozdział 2.2 — The Application Stage*

2.2 Etap geometrii (Geometry Stage)

Ten etap odpowiada za większość operacji na wierzchołkach i trójkątach. Składa się z kilku podetapów:

- Transformacja modelu (Model Transform) — przekształca wierzchołki z układu lokalnego obiektu do układu świata.
- Transformacja widoku (View Transform) — przenosi scenę do układu kamery, gdzie kamera znajduje się w początku układu współrzędnych.
- Rzutowanie (Projection) — stosuje rzut perspektywiczny lub ortogonalny, tworząc znormalizowaną kostkę widoku (NDC).
- Obcinanie (Clipping) — usuwa fragmenty prymitywów leżące poza bryłą widzenia.
- Mapowanie ekranowe (Screen Mapping) — przekształca współrzędne NDC na współrzędne pikseli ekranu.

> *Lektura: Real-Time Rendering, 3rd Ed., Rozdział 2.3 — The Geometry Stage*

2.3 Etap rasteryzacji (Rasterizer Stage)

Na tym etapie prymitywy (trójkąty) są zamieniane na fragmenty (potencjalne piksele). Dla każdego fragmentu oblicza się jego kolor, uwzględniając interpolowane atrybuty wierzchołków, tekstury, oświetlenie i inne efekty. Wynik zapisywany jest do bufora ramki (framebuffer).

- Przygotowanie trójkąta (Triangle Setup) — obliczanie krawędzi i gradientów.
- Przechodzenie trójkąta (Triangle Traversal) — wyznaczanie, które piksele leżą wewnątrz trójkąta.
- Cieniowanie pikseli (Pixel Shading) — obliczanie końcowego koloru każdego fragmentu.
- Operacje łączenia (Merging) — test głębi, mieszanie alfa, zapis do bufora ramki.

> *Lektura: Real-Time Rendering, 3rd Ed., Rozdział 2.4 — The Rasterizer Stage*

3. GPU a programowalny potok

Współczesne procesory graficzne (GPU) realizują potok renderowania sprzętowo. Kluczową cechą nowoczesnych GPU jest programowalność — programista może pisać własne programy cieniujące (shadery) w językach takich jak GLSL, HLSL czy Metal Shading Language.

- Vertex Shader — program wykonywany dla każdego wierzchołka, odpowiada za transformacje geometryczne.
- Fragment Shader (Pixel Shader) — program wykonywany dla każdego fragmentu, oblicza końcowy kolor piksela.
- Geometry Shader — opcjonalny shader przetwarzający całe prymitywy, może generować nową geometrię.

> *Lektura: Real-Time Rendering, 3rd Ed., Rozdział 3 — The Graphics Processing Unit*

4. Podsumowanie

Potok renderowania stanowi fundament grafiki czasu rzeczywistego. Zrozumienie kolejnych etapów — od przygotowania danych na CPU, przez transformacje geometryczne, aż po rasteryzację — jest kluczowe dla efektywnego programowania grafiki. W kolejnych wykładach omówimy szczegółowo każdy z tych etapów.