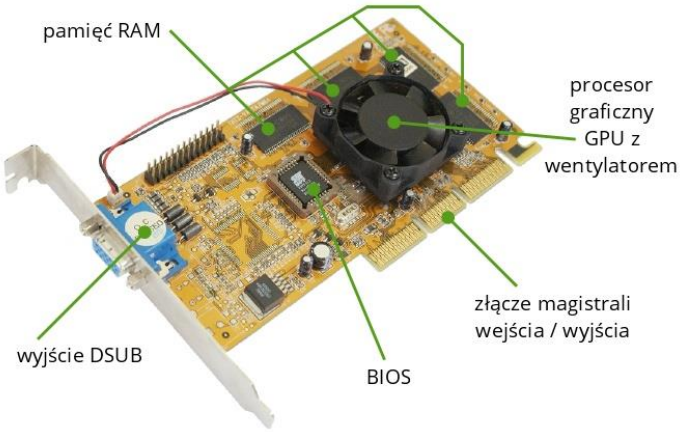
Budowa i metody programowania potoku graficznego współczesnych akceleratorów graficznych

Pytanie specjalnościowe nr 3

- Podzespół komputera, którego zadaniem jest wizualizacja danych cyfrowych.
- Akcelerator 3D = Karta graficzna obsługująca grafikę trójwymiarową
- Zintegrowana z chipsetem płyty głównej lub Karta rozszerzeń do magistrali we/wy na płycie głównej.

Architektura procesorów graficznych



1. **BIOS karty** (ang. Basic Input/Output System)
    - umożliwia działanie karty graficznej zanim zostanie wczytany system operacyjny oraz pozwala na wykonywanie instrukcji karty przez oprogramowanie systemowe.
2. **Złącze magistrali wejścia/wyjścia**
    - złącze na płycie głównej umożliwiający komunikację GPU z pozostałymi komponentami komputera
    - PCI Express
3. **Pamięć RAM** (ang. Random Access Memory)
    - pamięć operacyjna stanowiąca przestrzeń roboczą procesora GPU, przechowuje aktualnie przetwarzane dane graficzne.
    - Od wielkości pamięci RAM karty zależy liczba wyświetlanych kolorów w grafice 2D oraz jakość i rozmiary nanoszonych tekstur w grafice 3D.
4. **Procesor graficzny GPU** (ang. Graphics Processing Unit)
    - Główna jednostka obliczeniowa karty graficznej
    - Zbudowana z tysięcy rdzeni
5. **Konwerter cyfrowo-analogowy RAMDAC** (ang. Random Access Memory Digital-to-Analog Converter)
    - zamienia sygnał cyfrowy generowany przez kartę graficzną na sygnał analogowy (w którym napięcie elektryczne jest proporcjonalne do wartości reprezentowanej liczby cyfrowej), możliwy do wyświetlenia na analogowym ekranie.

6. **Zestaw wyjść (złącz)**
    - cyfrowe
        - HDMI
        - DVI
        - DisplayPort
    - Analogowe
        - D-SUB, VGA

CPU vs GPU

GPU jest zbudowane z tysięcy rdzeni, o niskim taktowaniu, w porównaniu do CPU. Dzięki takiej budowie są w stanie przetworzyć proste instrukcje na dużych zbiorach danych w wielokrotnie krótszym czasie (w szczególności operacje na macierzach, obrazach). CPU natomiast może obsługiwać bardziej złożone instrukcje, szybciej otrzymuje dane.

Powstawanie grafiki 3D

Za pośrednictwem **magistrali** nieobrobione dane są ładowane do **pamięci graficznej**, w której przechowywane są wszelkie obiekty graficzne i tekstury. Informacje są następnie odczytywane przez **procesor graficzny**, który przygotowuje do wyświetlenia na ekranie pozycje, ruchy i fakturę wszelkich obiektów widocznych w trójwymiarowej scenie. Po obróbce gotowy obraz przesyłany jest na zewnątrz poprzez **cyfrowe wyjścia** DVI, HDMI lub DisplayPort, lub wędruje do **RAMDAC** – układu, który przetwarza cyfrowe obrazy, nadając im formę odpowiednią dla **analogowego wyjścia** VGA.

Najważniejszym komponentem w procesie tworzenia grafiki 3D jest procesor graficzny GPU. Proces tworzenia kolejnych ramek obrazu jest sekwencyjny. Cała sekwencja powtarzana jest w wypadku każdej pojedynczej klatki. Aby wyświetlany ruch był płynny, ludzkie oko musi wychwycić co najmniej 25 klatek na sekundę. Największy realizm uzyskuje się dopiero przy 60 klatkach na sekundę.

- Scena 3D - Zbiór trójwymiarowych danych wejściowych służących do wygenerowania obrazu wyjściowego 2D
    - Obiekty
        - Geometria – obiekty są reprezentowane przy pomocy wielokątów (trójkątów), rozpiętych na wierzchołkach. Stosuje się transformacje afiniczne – skala, rotacja, przesunięcie w przestrzeni euklidesowej
        - Scena jest zbudowana z obiektów reprezentowanych hierarchicznie, przy pomocy podrzędnych obiektów.

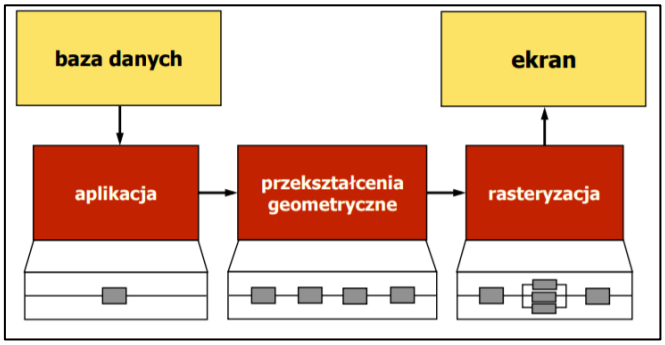
        - Materiał – kolor, tekstura, właściwości (rozproszenie, odbicie, załamanie)
    - Kamera – Definiuje jaka część świata zostanie wygenerowana
        - Określa sposób rzutowania (Prostokątne, perspektywiczne, …)
        - Kąt i kierunek patrzenia
    - Źródła światła

Potok graficzny

- **Potok graficzny** (ang. graphics pipeline) droga przepływu danych pomiędzy interfejsem karty graficznej a buforem ramki.
- **Bufor ramki** (ang. frame buffer) jest częścią pamięci RAM karty graficznej przeznaczoną do przechowywania informacji o pojedynczej ramce (klatce) obrazu. W buforze przechowywane są informacje o wartości każdego piksela tworzącego ramkę.

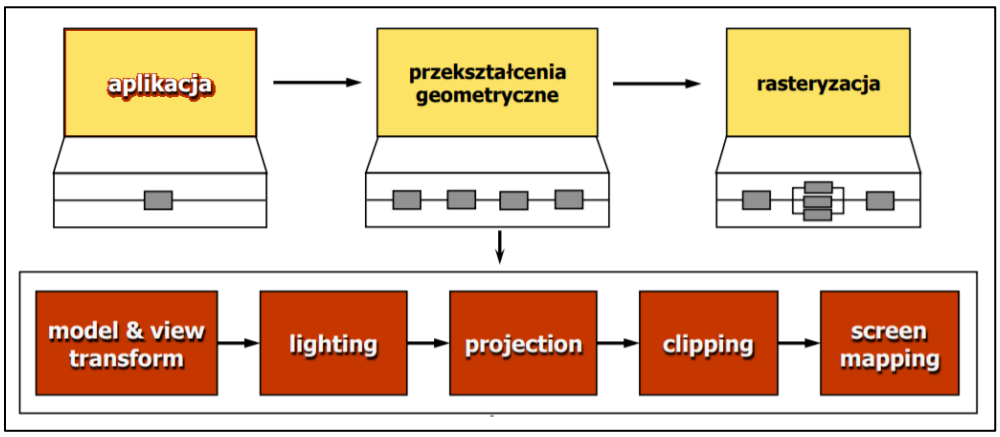
Potok graficzny dzieli się na 3 części

- **Aplikacja** – Program komputerowy, pozwalający na wyświetlanie, lub manipulację obiektami sceny 3D, ustawianie animacji, tekstur i zależności w świecie.
- **Przekształcenia geometryczne –** dane odczytane z aplikacji są przetwarzane w celu uzyskania wymaganych efektów wizualnych.
- **Rasteryzacja** – Operacje wykonywane w celu wyliczenia koloru każdego piksela w wyjściowym obrazie i umieszczenie ich w buforze ramki

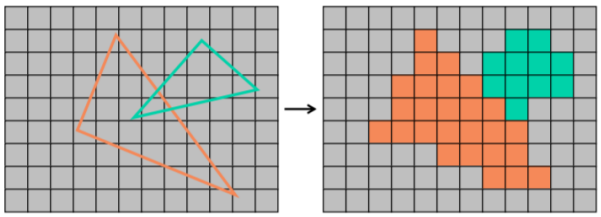


- **Przekształcenia geometryczne**
    - **Transformacja modelu** – każdy obiekt ma własny układ współrzędnych. Celem operacji jest ustalenie jego pozycji, orientacji oraz skali w układzie świata
    - **Transformacja widoku** - transformacja kamery do środka układu współrzędnych, gdzie kierunek patrzenia kamery pokrywa się z ujemną osią z. Jest to przekształcenie z układu świata do współrzędnych kamery w celu uproszczenia późniejszych obliczeń.
    - **Model oświetlenia** - obliczanie powierzchni obiektów, wynikającej z sumy komponentów
        - Ambient – aproksymacja odbić otoczenia. Odbijane jednakowo we wszystkich kierunkach . Jednolity kolor obiektu
        - Diffuse – pochodzące z jednego kierunku, odbijane we wszystkich kierunkach. Kolor z cieniem dla obiektów matowych, wynikający z prawa Lamberta
        - Specular – pochodzące z jednego kierunku, odbijane w jednym kierunku. Intensywne odbicia dla obiektów niematowych
    - **Rzutowanie**
        - Perspektywiczne – bryła widzenia jest ostrosłupem. Rzeczywisty wygląd
        - Prostokątne – bryła widzenia jest prostokątem. Pomocny podczas projektowania

- **Obcinanie**
  - Clipping – usuwanie obiektów poza polem widzenia kamery, wyznaczonych przez dwie powierzchnie przed kamerą – bliską i daleką.
  - Culling - -usuwanie obiektów zasłonionych
- **Mapowanie** na współrzędne ekranowe - Obliczenie współrzędnych rastrowych wierzchołków obiektów



- **Rasteryzacja**
  - Wypełnianie trójkątów - Określanie, które fragmenty trójkątów należą do poszczególnych pikseli obrazu. Dla każdego piksela tworzona jest pewna ilość fragmentów, ale ostatecznie jest wyświetlany ten, którego dane wskazują, że jest z przodu (głębia), lub łączone jest wiele fragmentów (przeźroczystość).
  - Cieniowanie pikseli - Obliczanie kolorów fragmentów trójkątów na podstawie światła i koloru wierzchołków



Programowanie potoku graficznego

Programowanie potoku graficznego umożliwia twórcą implementację dowolnych, ulepszonych, lub nastawionych na określone cele modyfikacje potoku, przy pomocy API, bądź poprzez programowanie jednostek zwanych **shaderami**. Programista ma wpływ geometrię oraz ostateczny obraz, przez modyfikację wierzchołków lub pikseli.

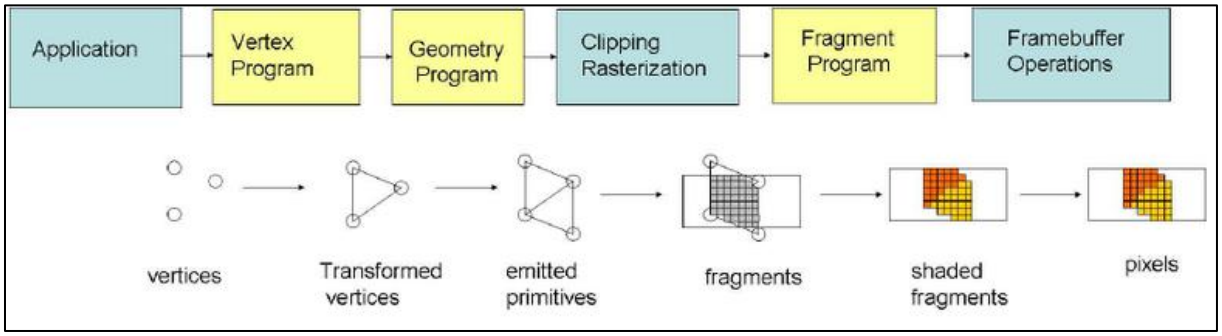
Istnieją dwa główne API graficzne, stanowiące kontakt oprogramowania ze sprzętem. Są to biblioteki definiujące sposób, w jaki zaimplementowano potok graficzny.

- **DirectX** jest produktem firmy Microsoft,
  - Dostępnym dla systemów Windows oraz konsoli XBox.
  - Jest programem, który należy zainstalować w systemie
  - Obsługa grafiki i dźwięku
  - Nakierowany na rozrywkę
- **OpenGL** to otwarta biblioteka graficzna, pod opieką Khronos Group

- Dostępny jest w wielu systemach operacyjnych (Windows, Mac OS, Unix/Linux, Android).
- Zaimplementowany przez producentów sprzętu w sterownikach
- Obsługa grafiki
- Nakierowany na inne aplikacje związane z grafiką (Blender, Google Earth, AutoCAD, Photoshop)

Potok w OpenGL:

1. Vertex Data – wczytanie danych
2. **Vertex Shader** (Edytowalny) – program wywoływany dla każdego wierzchołka. Celem jest przekształcenie wierzchołków do współrzędnych ekranu i dodatkowe przekształcenia pozycji.
3. **Geometry Shader** (Edytowalny) – Tworzenie nowych obiektów geometrycznych i dodawanie lub usuwanie wierzchołków w celu uzyskania określonej szczegółowości detali w siatce
4. Clipping – automatyczny proces usuwanie niewidocznych obiektów
5. Rasterization – Określanie, który piksele rzutni są pokryte kształtami.
6. **Fragment Shader** (Edytowalny) – Program wykonywany dla każdego fragmentu powstałego w rasteryzacji. Programista ustala zależności fragmentów, które ostatecznie tworzą piksel.



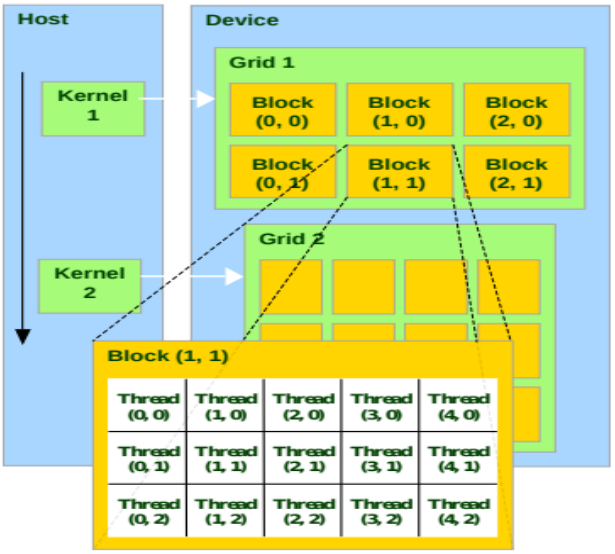
CUDA

CUDA to opracowana przez firmę NVIDIA równoległa architektura obliczeniowa, która zapewnia radykalny wzrost wydajności obliczeń, dzięki wykorzystaniu mocy układów GPU.

- Programy korzystające z CUDA pisane są w językach C1 , C++2 oraz Fortran
- W modelu CUDA procesor główny CPU stanowi tzw. procesor macierzysty (ang. host)
- Procesor graficzny GPU nazywany jest urządzeniem (and. device)
- Procesor graficzny złożony jest z tzw. wieloprocesorów (ang. multiprocessors)
- Program uruchamiany jest na CPU, oznaczone funkcje nazywane jądrami obliczeniowymi (ang. kernel) wykonywane są przez GPU

Podział pracy

- **Kernel** w modelu CUDA wykonywany jest równolegle przez wiele wątków (ang. thread)
- Wszystkie wątki kernela należą do jednej **kraty** (ang. grid)
- Wątki wewnątrz kraty podzielone są na **bloki**. Podział na bloki wynika z architektury GPU
- **Wątki** w bloku wykonywane są na tym samym wieloprocesorze strumieniowym (SM)
- Wątek ma unikalny ID wewnątrz bloku
- Blok ma unikalne ID wewnątrz kraty



OpenCL

- Platforma programistyczna do tworzenia aplikacji działających na platformach sprzętowych złożonych z procesorów (CPU), kart graficznych (GPU)
- Architektura podobna do CUDA – w większości przypadków istnieją bezpośrednie odpowiedniki dla poszczególnych metod, typów danych itp.

Problemy

- Problem z kompatybilnością kodu – np. CUDA działa na kartach Nvidia, ale nie ATI
- Mnogość modeli kart graficznych – uzyskanie najlepszej wydajności wymaga często „ręcznej” optymalizacji dla konkretnego modelu
- Programowanie dla GPU wymaga dobrej znajomości architektury i jest w pewnym sensie „niskopoziomowe”
- Trudniejsze debugowanie programów dla GPU
- Wykorzystanie mocy GPU wymaga odpowiednich algorytmów

Zastosowania

Procesory graficzne mają szerokie zastosowanie we wszelkich zadaniach, które można **zrównoleglić**. Ze względu na dużą wydajność w stosunku do zapotrzebowania na energię GPU są obecnie wykorzystywane w większości najszybszych superkomputerów (patrz top500.org).

Źródła

https://miroslawzelent.pl/informatyka/karty-graficzne-grafika-3d/ (28.04.2019)

http://ikot.edu.pl/artykuly,utk,karty_graficzne,8.html (28.04.2019)

https://shot511.github.io/2014-06-08-tutorial-04-czym-jest-programowalny-potok-renderowania/ (28.04.2019)

http://rmantiuk.strony.wi.ps.pl/data/wyklad_gk_potok_graficzny.pdf (28.04.2019)

http://web3d.toborowicz.pl/grafika-3d/ (28.04.2019)

http://www.atomaszewska.zut.edu.pl/data/wyklady/ic/2016/ic_wyklad2_2016.pdf (28.04.2019)