Programowanie Grafiki Komputerowej

OpenGL - wprowadzenie do programowania grafiki komputerowej, transformacje 3D, oświetlenie i materiały, teksturowanie, dodatkowe efekty graficzne

mgr inż. Maciej Lasota

Katedra Informatyki, Politechnika Świętokrzyska w Kielcach

część 1



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



Politechnika Świętokrzyska

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

" Program Rozwojowy Potencjału Dydaktycznego Politechniki Świętokrzyskiej w Kielcach: *kształcenie w nowoczesnych obszarach techniki* "

Program Operacyjny Kapitał Ludzki, Priorytet IV Szkolnictwo wyższe i nauka, Działanie 4.1. Wzmocnienie i rozwój potencjału dydaktycznego uczelni oraz zwiększenie liczby absolwentów kierunków o kluczowym znaczeniu dla gospodarki opartej na wiedzy, Poddziałanie 4.1.1. Wzmocnienie potencjału dydaktycznego uczelni.

Umowa UDA–POKL.04.01.01-00395/09-00

Spis treści

1. Grafika komputerowa

2. Programowanie grafiki komputerowej
   - Ogólny proces tworzenia aplikacji graficznej
   - Biblioteki graficzne
   - Karta graficzna

3. Wprowadzenie do biblioteki OpenGL
   - Ogólna koncepcja działania
   - Architektura OpenGL
   - Potok renderowania

4. Dodatkowe biblioteki
   - Biblioteka GLU
   - Biblioteka GLUT

5. Bibliografia

Uwagi

- Animacje są uruchamiane, gdy plik PDF jest wyświetlany na pełnym ekranie.
- Animacja jest uruchamiana dopiero po naciśnięciu spacji (program Adobe Reader).
- Klatki animacji wyświetlane są co 3 sekundy.
- Symulacje są uruchamiane po kliknięciu w ikonę z odpowiednim systemem operacyjnym. W systemie Linux należy dodatkowo kliknąć dwukrotnie w nazwę programu.
- Do uruchamiania symulacji z pliku PDF, wymagany jest program Adobe Reader w wersji 5.0 lub nowszej.

Grafika komputerowa

Grafika komputerowa *(ang. computer graphics)* to jedna z wielu szeroko pojętych dziedzin informatyki, której głównym założeniem i celem jest wykorzystywanie różnorakich **technik komputerowych** do wirtualizacji oraz wizualizacji otaczającej nas rzeczywistości.

Grafika komputerowa

Grafika komputerowa *(ang. computer graphics)* to jedna z wielu szeroko pojętych dziedzin informatyki, której głównym założeniem i celem jest wykorzystywanie różnorakich **technik komputerowych** do wirtualizacji oraz wizualizacji otaczającej nas rzeczywistości. Grafika komputerowa stanowi obecnie podstawę wielu dziedzin życia. Stosowana jest w nauce, technice, przemyśle oraz rozrywce *(gry komputerowej, animacje, filmy, reklamy itp.)*.

Programowanie grafiki komputerowej

Programowanie grafiki komputerowej *(ang. programming graphics)* to złożony wieloetapowy proces tworzenia **aplikacji (programów komputerowych)**, których głównym zadaniem jest realistyczne odzwierciedlanie oraz wizualizowanie rzeczywistość.

Programowanie grafiki komputerowej

Programowanie grafiki komputerowej *(ang. programming graphics)* to złożony wieloetapowy proces tworzenia **aplikacji (programów komputerowych)**, których głównym zadaniem jest realistyczne odzwierciedlanie oraz wizualizowanie rzeczywistość. Do programowania grafiki komputerowej wykorzystuje się języki programowania wysokiego poziomu *(np. C, C++, C#, Java itp.)* i/lub języki niskopoziomowe (*asemblery)* w połączeniu z odpowiednimi **bibliotekami graficznymi** ułatwiającymi i usprawniającymi proces tworzenia aplikacji graficznych.

Programowanie grafiki komputerowej

Programowanie grafiki komputerowej *(ang. programming graphics)* to złożony wieloetapowy proces tworzenia **aplikacji (programów komputerowych)**, których głównym zadaniem jest realistyczne odzwierciedlanie oraz wizualizowanie rzeczywistość. Do programowania grafiki komputerowej wykorzystuje się języki programowania wysokiego poziomu *(np. C, C++, C#, Java itp.)* i/lub języki niskopoziomowe (*asemblery)* w połączeniu z odpowiednimi **bibliotekami graficznymi** ułatwiającymi i usprawniającymi proces tworzenia aplikacji graficznych.

Uwaga

Tak stworzone aplikacje mogą być uruchamiane na docelowym sprzęcie komputerowym tj. **określonej karcie graficznej**.

Ogólny proces tworzenia aplikacji graficznej

Programowanie grafiki komputerowej stanowi połączenie dziedzin **grafiki komputerowej** oraz **inżynierii programowania**. Każdy proces powstawania dowolnej aplikacji *(programu komputerowego)* można podzielić na pewne fazy.

Ogólny proces tworzenia aplikacji graficznej

1. **Specyfikacja** *(określanie i ustalanie wymagań jakie musi spełniać aplikacja)*.

Ogólny proces tworzenia aplikacji graficznej

1. **Specyfikacja** *(określanie i ustalanie wymagań jakie musi spełniać aplikacja)*.
2. **Projektowanie** *(ustalanie architektury aplikacji oraz poszczególnych jej składowych)*.

Ogólny proces tworzenia aplikacji graficznej

1. **Specyfikacja** *(określanie i ustalanie wymagań jakie musi spełniać aplikacja)*.
2. **Projektowanie** *(ustalanie architektury aplikacji oraz poszczególnych jej składowych)*.
3. **Implementacja** *(tworzenie określonej architektury aplikacji, poszczególnych składowych oraz połączeń między nimi)*.

Ogólny proces tworzenia aplikacji graficznej

1. **Specyfikacja** *(określanie i ustalanie wymagań jakie musi spełniać aplikacja)*.
2. **Projektowanie** *(ustalanie architektury aplikacji oraz poszczególnych jej składowych)*.
3. **Implementacja** *(tworzenie określonej architektury aplikacji, poszczególnych składowych oraz połączeń między nimi)*.
4. **Integracja** *(zintegrowanie wszystkich składowych aplikacji)*.

Ogólny proces tworzenia aplikacji graficznej

1. **Specyfikacja** *(określanie i ustalanie wymagań jakie musi spełniać aplikacja)*.
2. **Projektowanie** *(ustalanie architektury aplikacji oraz poszczególnych jej składowych)*.
3. **Implementacja** *(tworzenie określonej architektury aplikacji, poszczególnych składowych oraz połączeń między nimi)*.
4. **Integracja** *(zintegrowanie wszystkich składowych aplikacji)*.
5. **Ewolucja** *(uruchamianie oraz testowanie stworzonej aplikacji)*.

Ogólny proces tworzenia aplikacji graficznej

1. **Specyfikacja** *(określanie i ustalanie wymagań jakie musi spełniać aplikacja)*.
2. **Projektowanie** *(ustalanie architektury aplikacji oraz poszczególnych jej składowych)*.
3. **Implementacja** *(tworzenie określonej architektury aplikacji, poszczególnych składowych oraz połączeń między nimi)*.
4. **Integracja** *(zintegrowanie wszystkich składowych aplikacji)*.
5. **Ewolucja** *(uruchamianie oraz testowanie stworzonej aplikacji)*.

Uwaga

Ponieważ aplikacje graficzne różnią się od siebie zarówno przeznaczeniem jak i rodzajem oraz typem, przedstawiony proces oraz fazy tworzenia aplikacji mogą się różnić i składać z innych elementów.

Biblioteki graficzne

Biblioteki graficzne to zestawy **dodatkowego oprogramowania**, które ułatwiają proces tworzenia oraz wspomagają i kontrolują działanie aplikacji graficznych.

Biblioteki graficzne

Biblioteki graficzne to zestawy **dodatkowego oprogramowania**, które ułatwiają proces tworzenia oraz wspomagają i kontrolują działanie aplikacji graficznych. Stanowią one *programowy interfejs sprzętu* i są warstwą pośredniczącą między aplikacjami graficznymi, a sterownikiem karty graficznej.

Biblioteki graficzne

Biblioteki graficzne biorą udział w renderowaniu *(ang. rendering)*, czyli procesie generowania obrazu *(tzw. obrazowaniu lub inaczej prezentacji)*. Posiadaj szereg funkcji i procedur **(określone API)** które obsługują podstawowe zadania renderowania.

Biblioteki graficzne

Biblioteki graficzne biorą udział w renderowaniu *(ang. rendering)*, czyli procesie generowania obrazu *(tzw. obrazowaniu lub inaczej prezentacji)*. Posiadaj szereg funkcji i procedur **(określone API)** które obsługują podstawowe zadania renderowania. Część funkcjonalności bibliotek wykonywana jest *programowo* z wykorzystaniem **CPU** natomiast cześć zoptymalizowana jest *sprzętowo*. Oznacza to że funkcje i procedury biblioteki są bezpośrednio zaimplementowane w **GPU** karty graficznej przez producenta danej karty, co przyspiesza proces generowania obrazu.

Biblioteki graficzne

Biblioteki graficzne biorą udział w renderowaniu *(ang. rendering)*, czyli procesie generowania obrazu *(tzw. obrazowaniu lub inaczej prezentacji)*. Posiadaj szereg funkcji i procedur **(określone API)** które obsługują podstawowe zadania renderowania. Część funkcjonalności bibliotek wykonywana jest *programowo* z wykorzystaniem **CPU** natomiast cześć zoptymalizowana jest *sprzętowo*. Oznacza to że funkcje i procedury biblioteki są bezpośrednio zaimplementowane w **GPU** karty graficznej przez producenta danej karty, co przyspiesza proces generowania obrazu.

Uwaga

Wykorzystywanie bibliotek graficznych zwalnia programistę od zadania tworzenia własnych funkcji i procedur bezpośrednio odwołujących się do **GPU**, pozwala to skupić się na samym procesie powstawania aplikacji.

Biblioteki graficzne

Przykładowe biblioteki graficzne

- OpenGL
- DirectX *(DirectDraw, Direct2D, Direct3D)*
- SDL
- Cairo
- Allegro
- Java 2D/3D

Biblioteki graficzne

Podział biblioteki graficznych ze względu na ich przeznaczenie

- Biblioteki do grafiki 2D *(np. Allegro)*
- Biblioteki do grafiki 3D *(np. OpenGL, DirectX)*
- Biblioteki do obsługi grafiki rastrowej *(np. libjpeg)*
- Biblioteki do obsługi grafiki wektorowej *(np. Cairo)*
- Biblioteki do tworzenia interfejsu użytkownika GUI *(np. GTK+, Qt)*

Karta graficzna

Karta graficzna *(ang. video card, graphics accelerator card)* to urządzenie odpowiadające za wyświetlanie obrazu na **monitorze komputera** oraz **proces renderowania**. Współczesne karty graficzne oprócz wyświetlania obrazu, pełnią również rolę *akceleratora grafiki trójwymiarowej*, tzn. przejmują część zadań obliczeniowych związanych z generowanie obrazu co odciąża CPU.

Karta graficzna

Karta graficzna *(ang. video card, graphics accelerator card)* to urządzenie odpowiadające za wyświetlanie obrazu na **monitorze komputera** oraz **proces renderowania**. Współczesne karty graficzne oprócz wyświetlania obrazu, pełnią również rolę *akceleratora grafiki trójwymiarowej*, tzn. przejmują część zadań obliczeniowych związanych z generowanie obrazu co odciąża CPU. Karty graficzne wyposażone są zazwyczaj w szybką pamięć RAM, w której przechowują aktualnie wyświetlany obraz, a także tekstury (bitmapy wykorzystywane do pokrywania powierzchni w grafice trójwymiarowej).

Karta graficzna

Karta graficzna *(ang. video card, graphics accelerator card)* to urządzenie odpowiadające za wyświetlanie obrazu na **monitorze komputera** oraz **proces renderowania**. Współczesne karty graficzne oprócz wyświetlania obrazu, pełnią również rolę *akceleratora grafiki trójwymiarowej*, tzn. przejmują część zadań obliczeniowych związanych z generowanie obrazu co odciąża CPU. Karty graficzne wyposażone są zazwyczaj w szybką pamięć RAM, w której przechowują aktualnie wyświetlany obraz, a także tekstury (bitmapy wykorzystywane do pokrywania powierzchni w grafice trójwymiarowej).

Uwaga

Zasada działania oraz architektura współczesnych kart graficznych jest bardzo skomplikowana i wychodzi poza ramy tematyczne tej prezentacji.

Wprowadzenie do biblioteki OpenGL

OpenGL *(ang. Open Graphics Library)* to biblioteka graficzna o otwartym kodzie źródłowym, szybka w pełni uniwersalna i łatwo przenośna na różne platformy sprzętowe (w tym również systemy mobilne).

Wprowadzenie do biblioteki OpenGL

OpenGL *(ang. Open Graphics Library)* to biblioteka graficzna o otwartym kodzie źródłowym, szybka w pełni uniwersalna i łatwo przenośna na różne platformy sprzętowe (w tym również systemy mobilne). Przeznaczona jest do tworzenia **dwuwymiarowej** oraz **trójwymiarowej** grafiki komputerowej. Większość funkcji biblioteki realizowana jest programowo lub sprzętowo, współpracuje z takimi językami jak C, C++, VisualBasic, Delphi, Java, C# i innymi.

Wprowadzenie do biblioteki OpenGL

Bezpośrednim przodkiem OpenGL była biblioteka **IRIS GL** firmy *Silicon Graphics*. Jest ona rezultatem prac firmy Silicon Graphics nad poprawą IRIS GL oraz przeniesieniem jej ze stacji graficznych **IRIS** na inne platformy. W lipcu 1992 powstał pierwsza wersja OpenGL 1.0.

Wprowadzenie do biblioteki OpenGL

Bezpośrednim przodkiem OpenGL była biblioteka **IRIS GL** firmy *Silicon Graphics*. Jest ona rezultatem prac firmy Silicon Graphics nad poprawą IRIS GL oraz przeniesieniem jej ze stacji graficznych **IRIS** na inne platformy. W lipcu 1992 powstał pierwsza wersja OpenGL 1.0. Obecnie OpenGL (aktualna wersja **4.1**) używana jest do różnych celów, od programów typu CAD przez aplikacje architektoniczne do generowania efektów specjalnych w filmach, reklamach oraz grach komputerowych.

Wprowadzenie do biblioteki OpenGL

Umożliwia ona rysowanie punktów, odcinków i wielokątów w trzech wymiarach. Ponadto obsługuje oświetlenie, cieniowanie, mapowanie tekstur, animacje oraz dodatkowe efekty specjalne.

Wprowadzenie do biblioteki OpenGL

Umożliwia ona rysowanie punktów, odcinków i wielokątów w trzech wymiarach. Ponadto obsługuje oświetlenie, cieniowanie, mapowanie tekstur, animacje oraz dodatkowe efekty specjalne. OpenGL nie zawiera żadnych funkcji przeznaczonych do zarządzania oknami, interakcją z użytkownikiem czy też operacjami wejścia/wyjścia.

Wprowadzenie do biblioteki OpenGL

Do połowy 2006 roku nadzór nad rozwojem biblioteki pełniła organizacja **ARB (Architectural Review Board)**.

Wprowadzenie do biblioteki OpenGL

Do połowy 2006 roku nadzór nad rozwojem biblioteki pełniła organizacja **ARB (Architectural Review Board)**. Od 31 lipca 2006 ARB stała się częścią organizacji **Khronos**, która prowadzi prace nad wieloma pokrewnymi projektami. W skład grupy Khronos wchodzą między innymi firmy: 3Dlabs, ATI (AMD), nVidia, SGI, Intel i Sun (Oracle).

Ogólna koncepcja działania

Koncepcja działania biblioteki OpenGL polega na odwoływaniu się w aplikacji graficznej do określonych funkcji **API** biblioteki w celu bezpośredniego dostępu do karty graficznej (wykorzystując sterownik karty graficznej) z pominięciem *systemu operacyjnego*.

Ogólna koncepcja działania

Koncepcja działania biblioteki OpenGL polega na odwoływaniu się w aplikacji graficznej do określonych funkcji **API** biblioteki w celu bezpośredniego dostępu do karty graficznej (wykorzystując sterownik karty graficznej) z pominięciem *systemu operacyjnego*. OpenGL wspomaga obsługę grafiki 2D oraz 3D i stanowi programowy interfejs sprzętu graficznego.

Ogólna koncepcja działania

Większość obecnie produkowanych kart graficznych wspiera OpenGL sprzętowo. Oznacza to że część funkcji biblioteki zaimplementowana jest w **GPU** karty graficznej.

Ogólna koncepcja działania

Większość obecnie produkowanych kart graficznych wspiera OpenGL sprzętowo. Oznacza to że część funkcji biblioteki zaimplementowana jest w **GPU** karty graficznej.

Uwaga

Z punktu widzenia programisty aplikacji wywoływanie tak zaimplementowanych funkcji biblioteki jest niezauważalne.

Ogólna koncepcja działania

Większość obecnie produkowanych kart graficznych wspiera OpenGL sprzętowo. Oznacza to że część funkcji biblioteki zaimplementowana jest w **GPU** karty graficznej.

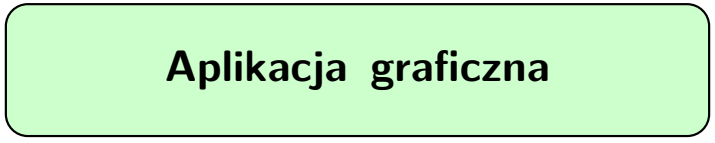
Uwaga

Z punktu widzenia programisty aplikacji wywoływanie tak zaimplementowanych funkcji biblioteki jest niezauważalne.

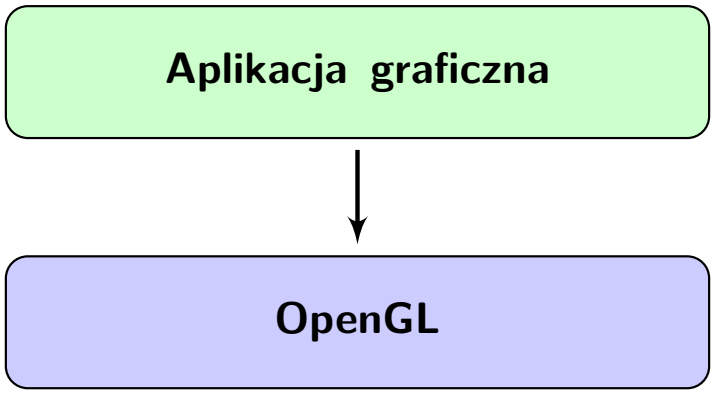
Uwaga

OpenGL zajmuje się tylko renderingiem grafiki, do obsługi *okien*, *klawiatury*, *myszy* itp. potrzebne są dodatkowe **biblioteki pomocnicze**.

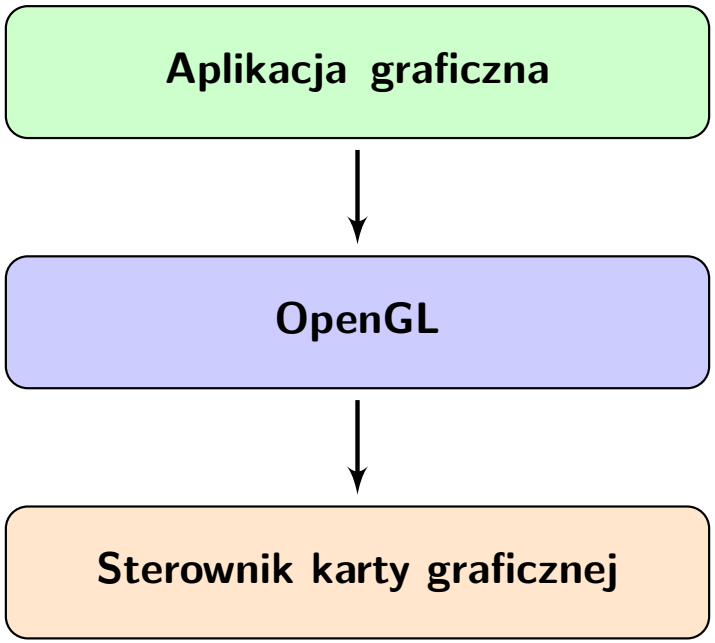
Ogólna koncepcja działania



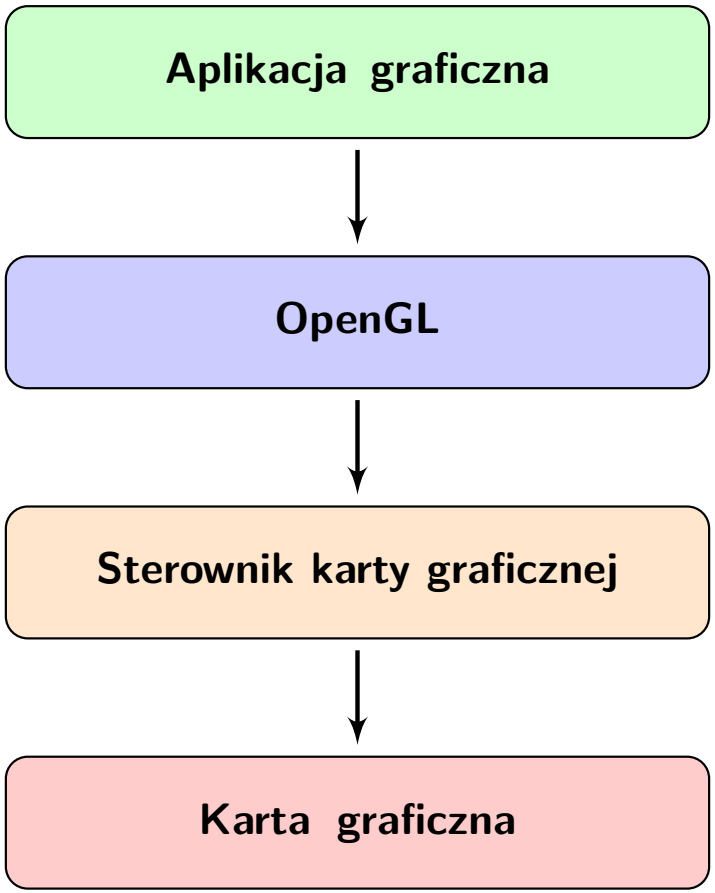
Ogólna koncepcja działania



Ogólna koncepcja działania



Ogólna koncepcja działania



Architektura OpenGL

Jedną z najważniejszych części OpenGL jest tzw. **maszyna stanu**, której głównym zadaniem jest *kontrola procesu renderowania obrazu* oraz *działań odpowiednich funkcji* biblioteki. Programista za pomocą funkcji API oraz parametrów maszyny może kontrolować jej stan.

Architektura OpenGL

Jedną z najważniejszych części OpenGL jest tzw. **maszyna stanu**, której głównym zadaniem jest *kontrola procesu renderowania obrazu* oraz *działań odpowiednich funkcji* biblioteki. Programista za pomocą funkcji API oraz parametrów maszyny może kontrolować jej stan. Na proces renderowania obrazu, którego wynik docelowo umieszczany jest w odpowiednim oknie w **buforze ramki** *(ang. framebuffer)* bezpośredni wpływ ma aktualny stan maszyny stanu.

Architektura OpenGL

Jedną z najważniejszych części OpenGL jest tzw. **maszyna stanu**, której głównym zadaniem jest *kontrola procesu renderowania obrazu* oraz *działań odpowiednich funkcji* biblioteki. Programista za pomocą funkcji API oraz parametrów maszyny może kontrolować jej stan. Na proces renderowania obrazu, którego wynik docelowo umieszczany jest w odpowiednim oknie w **buforze ramki** *(ang. framebuffer)* bezpośredni wpływ ma aktualny stan maszyny stanu.

Uwaga

Niepoprawnie ustawione parametry **maszyny stanu** mają negatywny wpływ na wynik renderowania obrazu.

Potok renderowania

Zadania wykonywane przez bibliotekę OpenGL umieszczane są na tzw. **liście rozkazów** lub inaczej **liście wyświetlania** *(ang. display list)*.

Potok renderowania

Zadania wykonywane przez bibliotekę OpenGL umieszczane są na tzw. **liście rozkazów** lub inaczej **liście wyświetlania** *(ang. display list)*. W zależności od tego, co jest przedmiotem renderowania (**dane wierzchołków** i/lub **dane pikseli**) proces przebieg zgodnie z określoną ścieżką przedstawioną na **potoku renderowania OpenGL** *(ang. OpenGL rendering pipeline)*.

Potok renderowania

Potok renderowania OpenGL składa się z ciągu operacji w którym przetwarzane są dane wejściowe.

Potok renderowania

Potok renderowania OpenGL składa się z ciągu operacji w którym przetwarzane są dane wejściowe.

- **dane wierzchołków** *(vertex data)* – opis obiektów i prymitywów w przestrzeni 3D.

Potok renderowania

Potok renderowania OpenGL składa się z ciągu operacji w którym przetwarzane są dane wejściowe.

- **dane wierzchołków** *(vertex data)* – opis obiektów i prymitywów w przestrzeni 3D.
- **dane pikseli** *(pixel data)* – tekstury i inne obrazy bitmapowe.

Potok renderowania

Potok renderowania OpenGL składa się z ciągu operacji w którym przetwarzane są dane wejściowe.

- **dane wierzchołków** *(vertex data)* – opis obiektów i prymitywów w przestrzeni 3D.
- **dane pikseli** *(pixel data)* – tekstury i inne obrazy bitmapowe.

Uwaga

Etapem końcowym potoku renderowania zawsze jest okno w buforze ramki *(ang. framebuffer)*.

Potok renderowania

Potok renderowania OpenGL składa się z ciągu operacji w którym przetwarzane są dane wejściowe.

- **dane wierzchołków** *(vertex data)* – opis obiektów i prymitywów w przestrzeni 3D.
- **dane pikseli** *(pixel data)* – tekstury i inne obrazy bitmapowe.

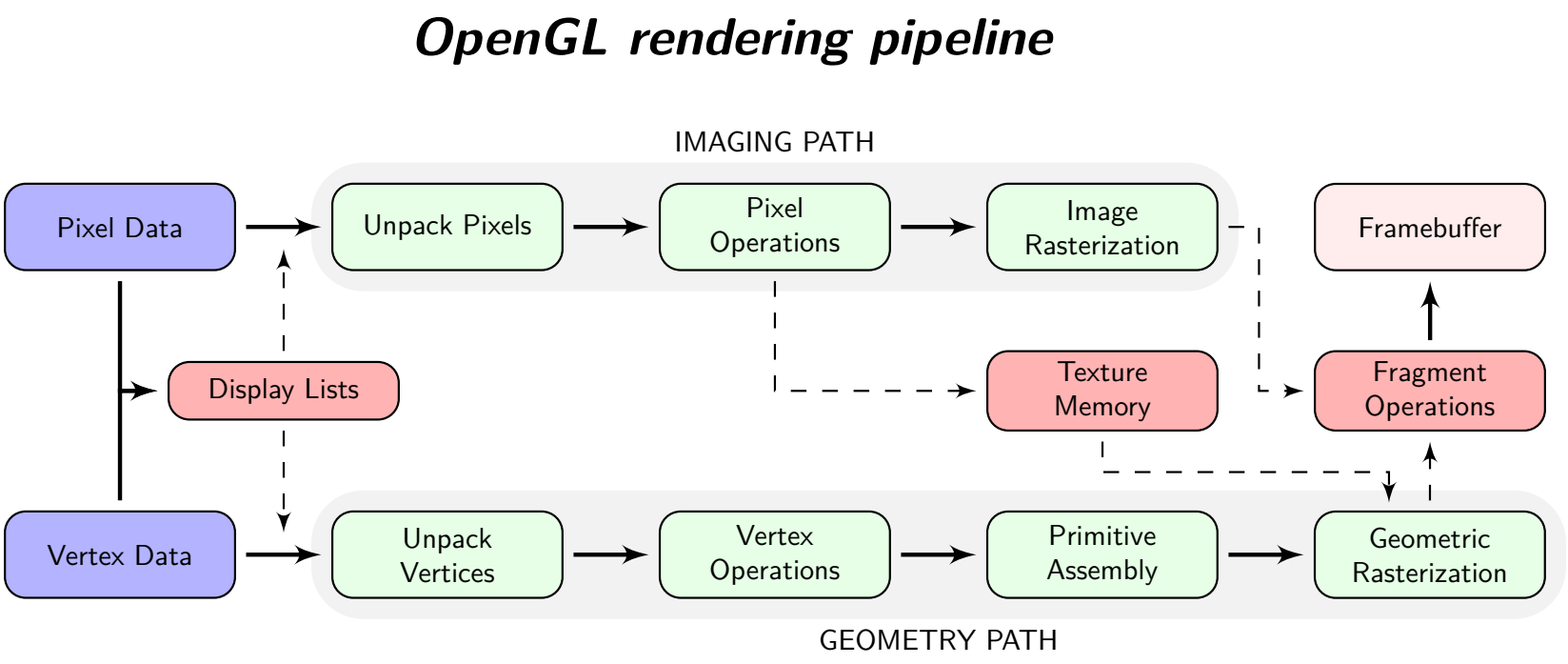
Uwaga

Etapem końcowym potoku renderowania zawsze jest okno w buforze ramki *(ang. framebuffer)*.

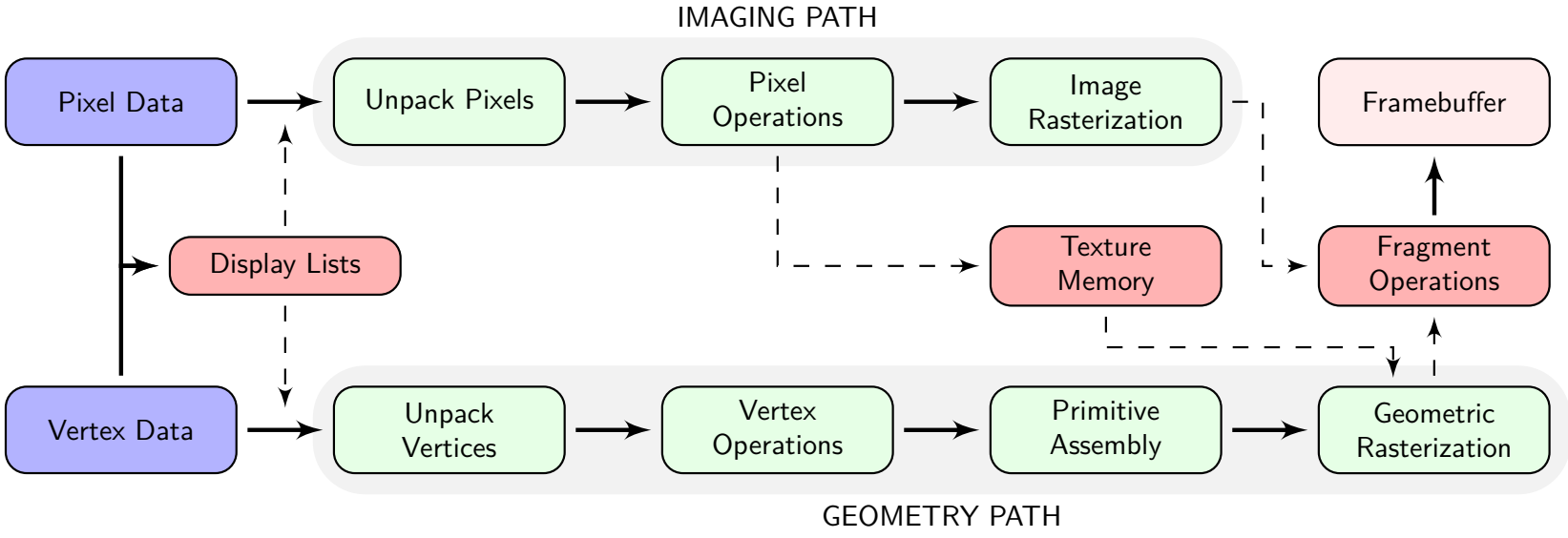
Uwaga

Możliwe są inne (wsteczne) kierunki przepływu danych – odczyt danych z buforów OpenGL: *koloru*, *głębokości*, *akumulacyjnego*, itp.

Potok renderowania



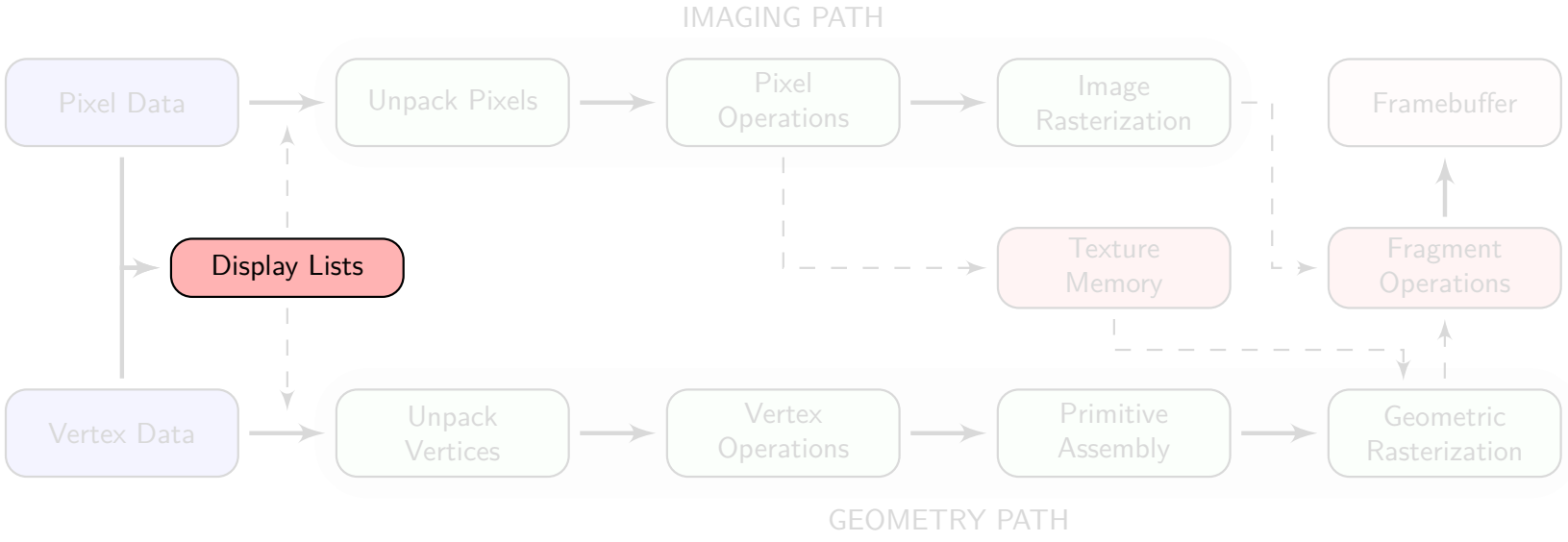
Potok renderowania



Listy wyświetlania

Listy wyświetlania *(display lists)* przechowują skompilowane instrukcje OpenGL. Pozwalają znacznie zwiększyć szybkość renderingu grafiki.

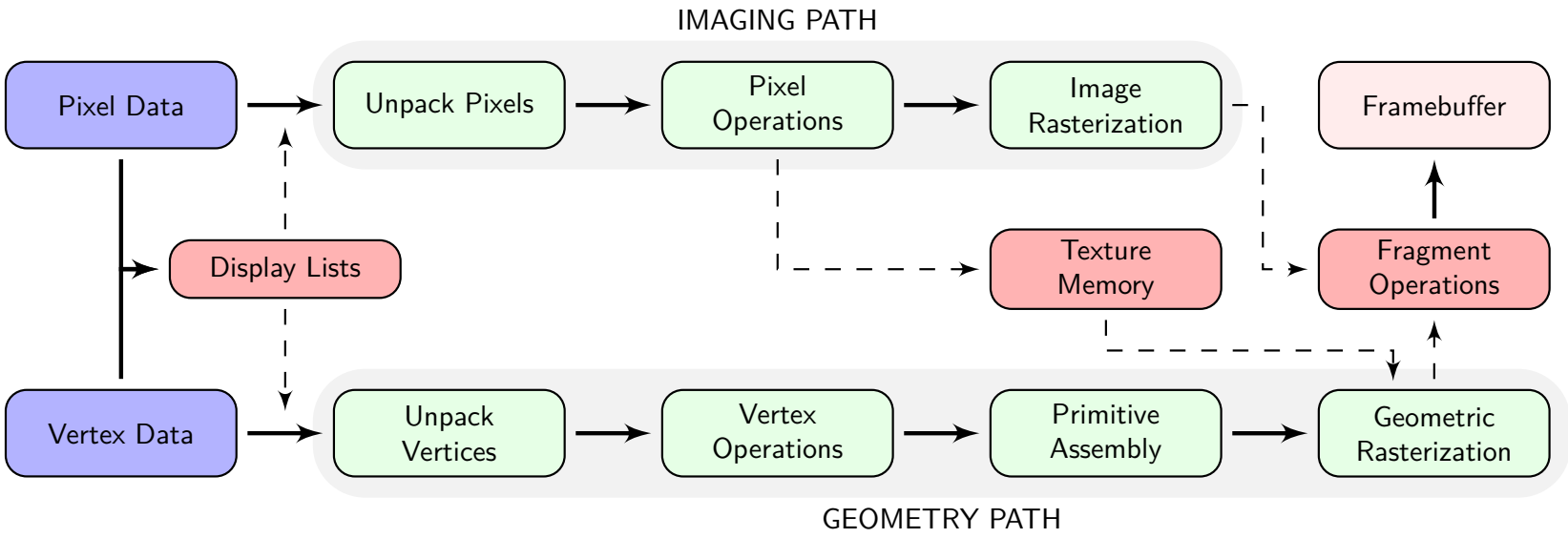
Potok renderowania



Listy wyświetlania

Listy wyświetlania *(display lists)* przechowują skompilowane instrukcje OpenGL. Pozwalają znacznie zwiększyć szybkość renderingu grafiki.

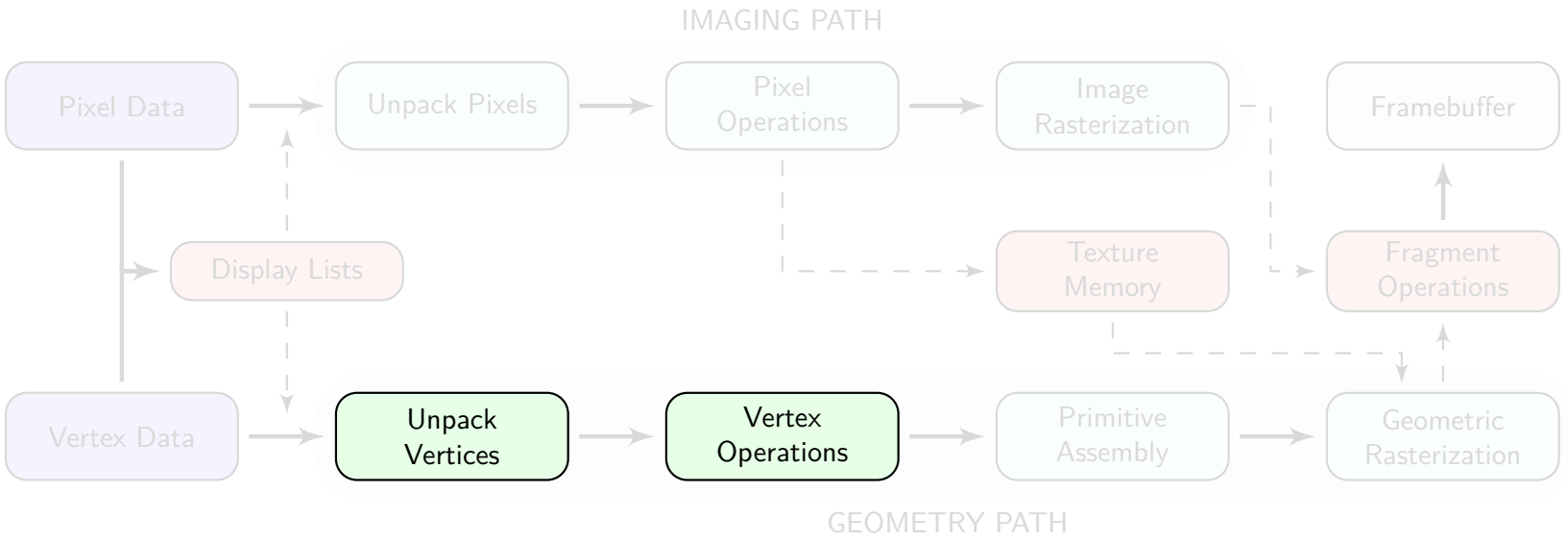
Potok renderowania



Operacje na wierzchołkach

Operacje na wierzchołkach *(vertex operations)* obejmują transformacje wszystkich wierzchołków i normalnych do współrzędnych globalnych (przez macierz GL_MODELVIEW). Jeśli włączone jest oświetlenie – kalkulacja oświetlenia i wyznaczenie kolorów wierzchołków.

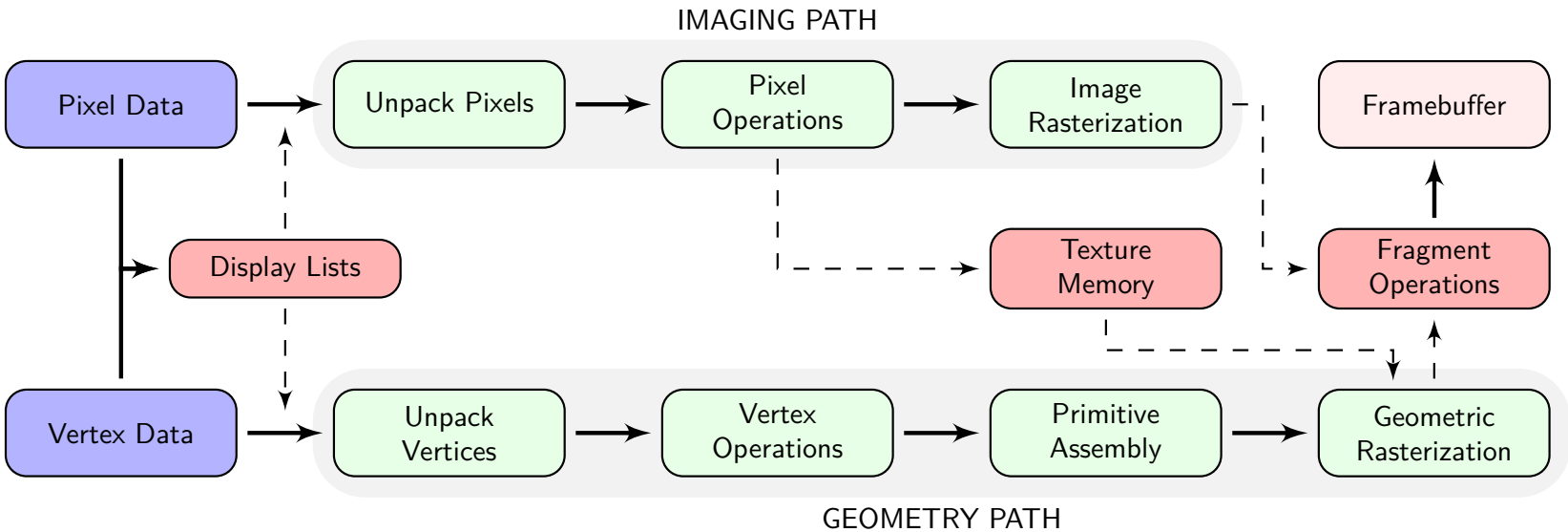
Potok renderowania



Operacje na wierzchołkach

Operacje na wierzchołkach *(vertex operations)* obejmują transformacje wszystkich wierzchołków i normalnych do współrzędnych globalnych (przez macierz GL_MODELVIEW). Jeśli włączone jest oświetlenie – kalkulacja oświetlenia i wyznaczenie kolorów wierzchołków.

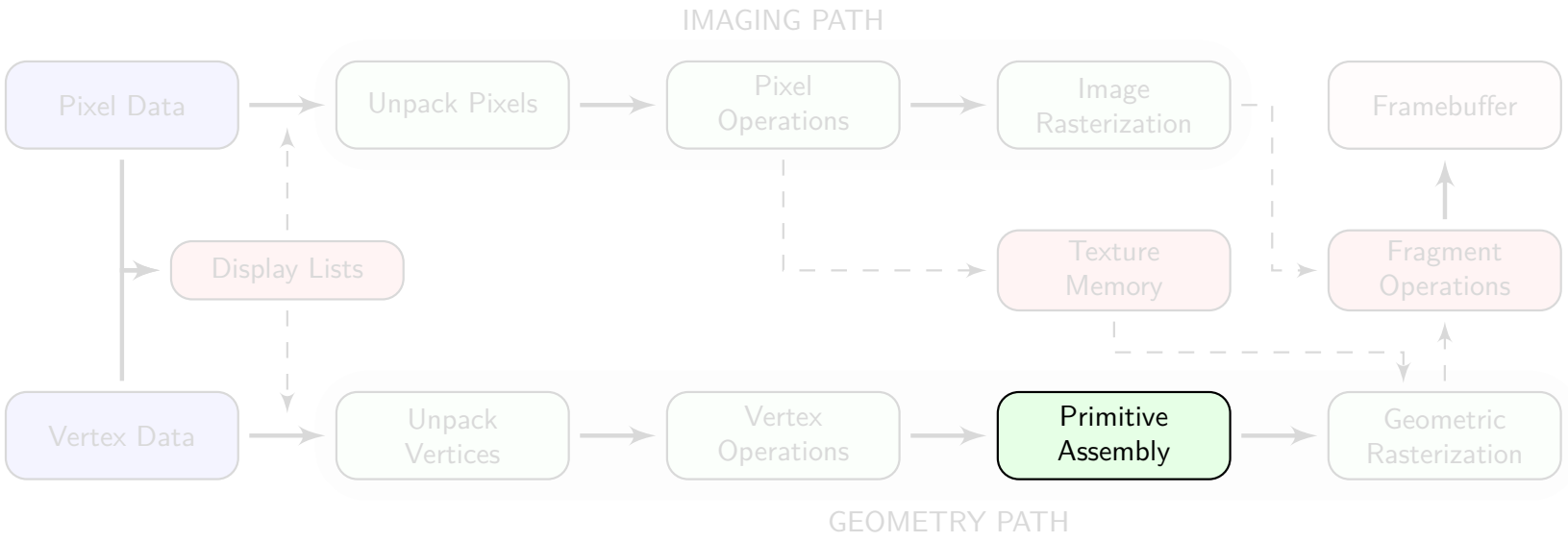
Potok renderowania



Składanie prymitywów

Składanie prymitywów *(primitive assembly)* – prymitywy (punkty, linie i wielokąty) są transformowane przez macierz rzutowania i obcinane do bryły widzenia. Następuje rzutowanie perspektywiczne i usuwanie powierzchni niewidocznych (face-culling).

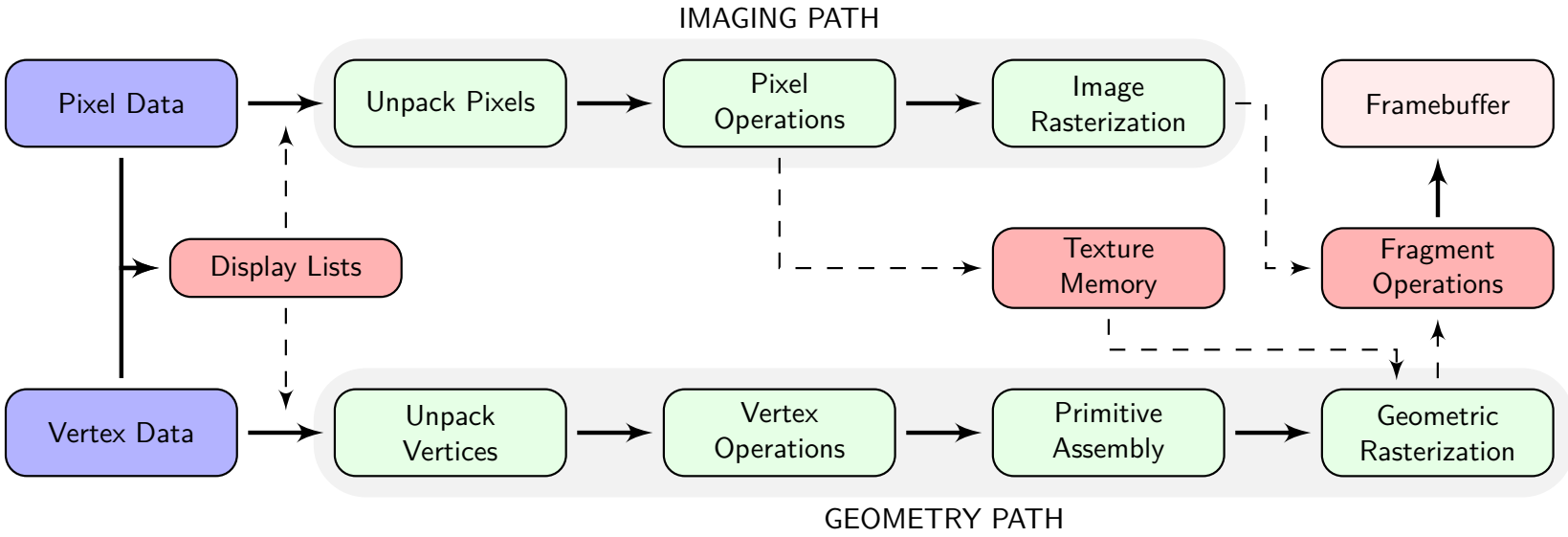
Potok renderowania



Składanie prymitywów

Składanie prymitywów *(primitive assembly)* – prymitywy (punkty, linie i wielokąty) są transformowane przez macierz rzutowania i obcinane do bryły widzenia. Następuje rzutowanie perspektywiczne i usuwanie powierzchni niewidocznych (face-culling).

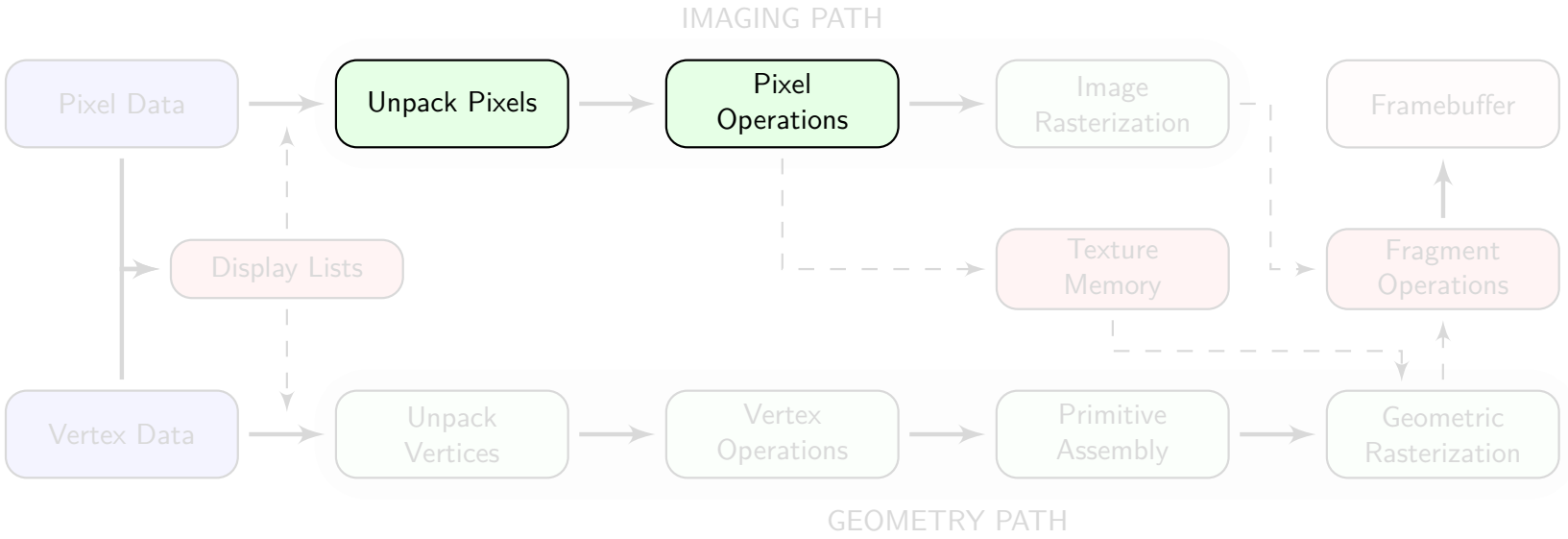
Potok renderowania



Operacje na pikselach

Operacje na pikselach czyli odczyt pikseli z pamięci użytkownika. Dodatkowo wykonywane jest skalowanie, obcinanie i inne operacje na bitmapach. Przetworzone dane składowane są w **pamięci tekstur** lub trafiają bezpośrednio do **rasteryzacji**.

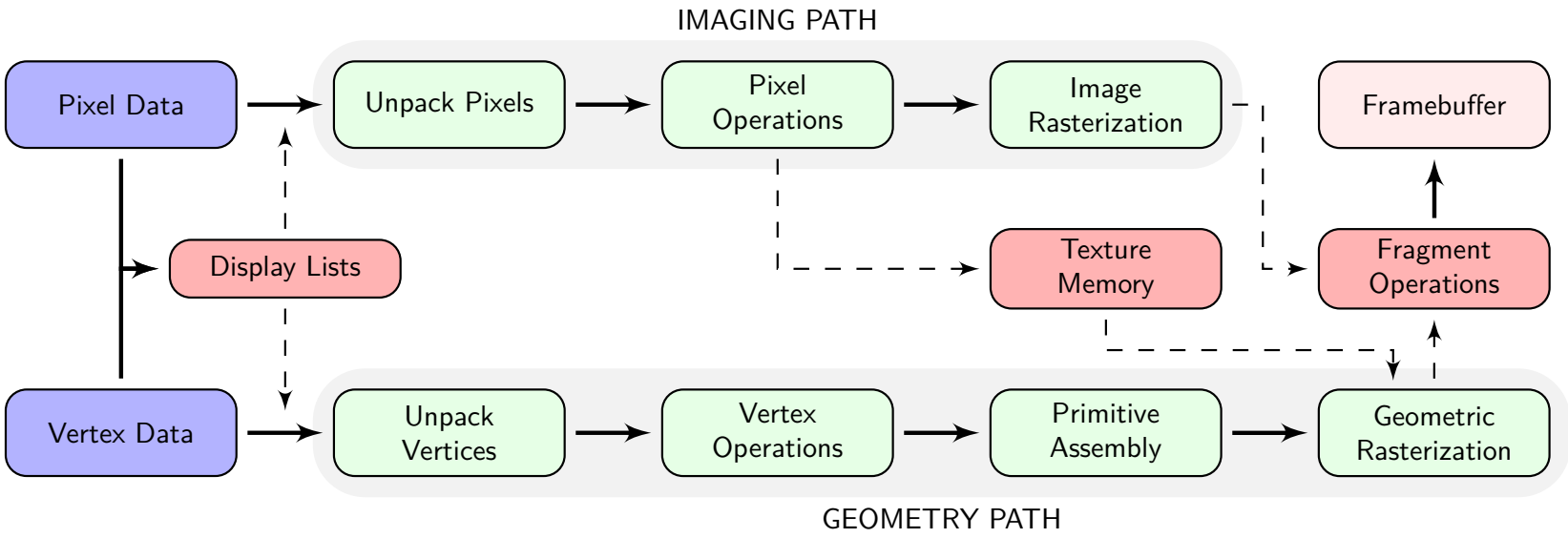
Potok renderowania



Operacje na pikselach

Operacje na pikselach czyli odczyt pikseli z pamięci użytkownika. Dodatkowo wykonywane jest skalowanie, obcinanie i inne operacje na bitmapach. Przetworzone dane składowane są w **pamięci tekstur** lub trafiają bezpośrednio do **rasteryzacji**.

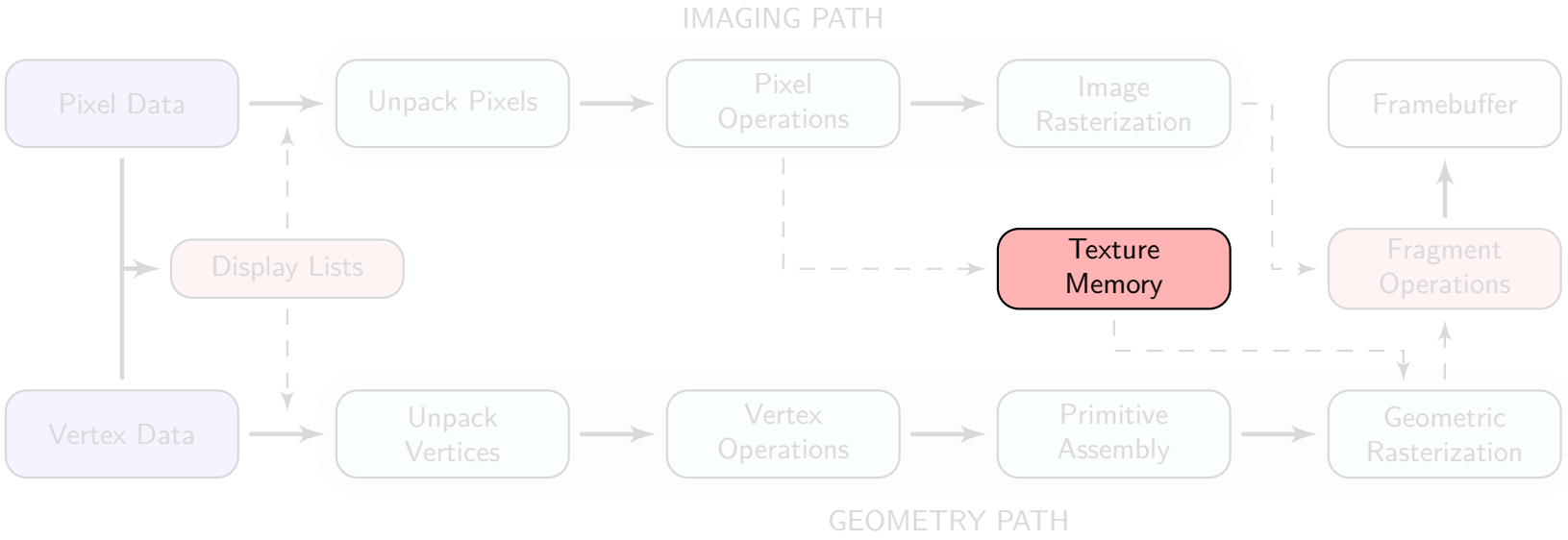
Potok renderowania



Pamięć tekstur

W **pamięci tekstur** *(texture memory)* przechowywane są tekstury nakładane na kształty geometryczne.

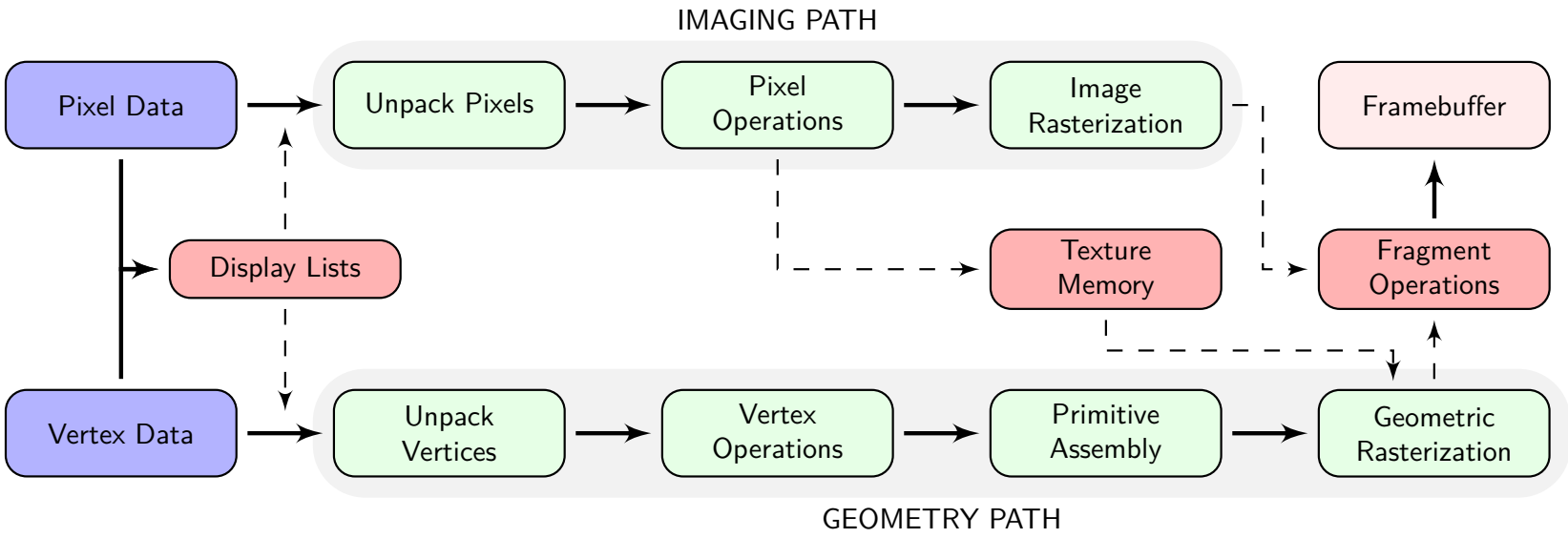
Potok renderowania



Pamięć tekstur

W **pamięci tekstur** *(texture memory)* przechowywane są tekstury nakładane na kształty geometryczne.

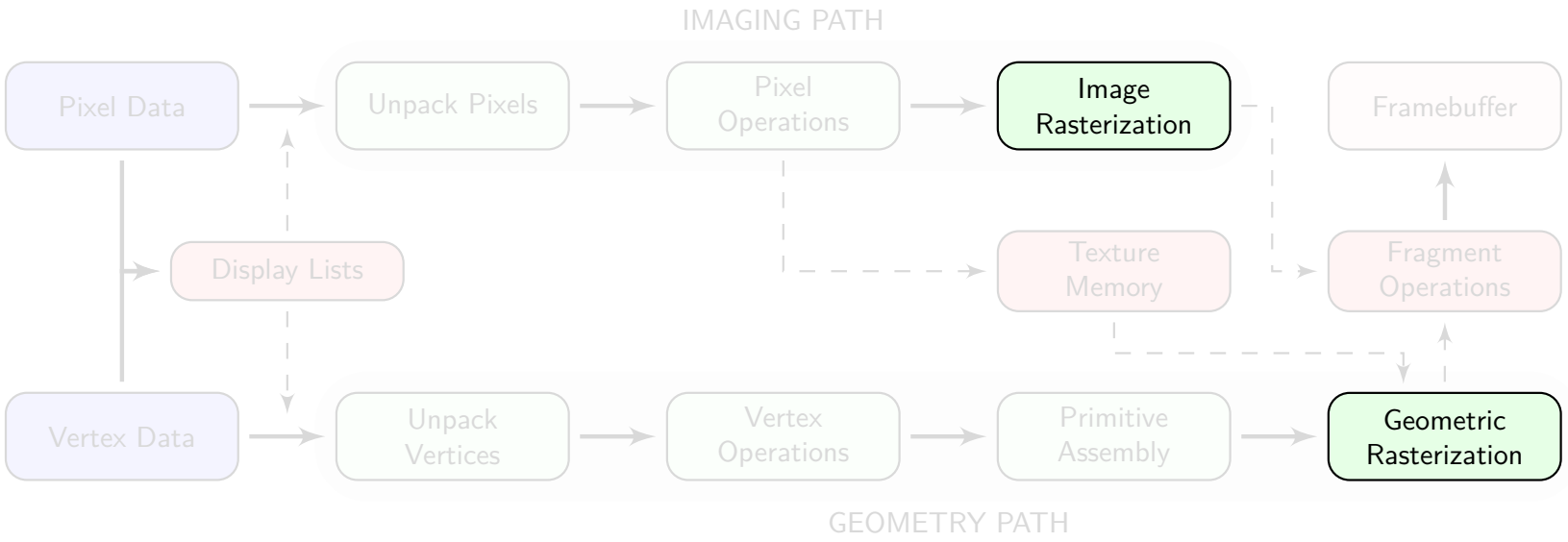
Potok renderowania



Rasteryzacja

Rasteryzacja *(rasterization)* – to konwersja kształtów geometrycznych i pikseli na fragmenty. Fragmenty przechowywane są w prostokątnej tablicy zawierającej informacje o kolorze, głębokości itp. Jeden fragment odpowiada jednemu pikselowi w buforze ramki.

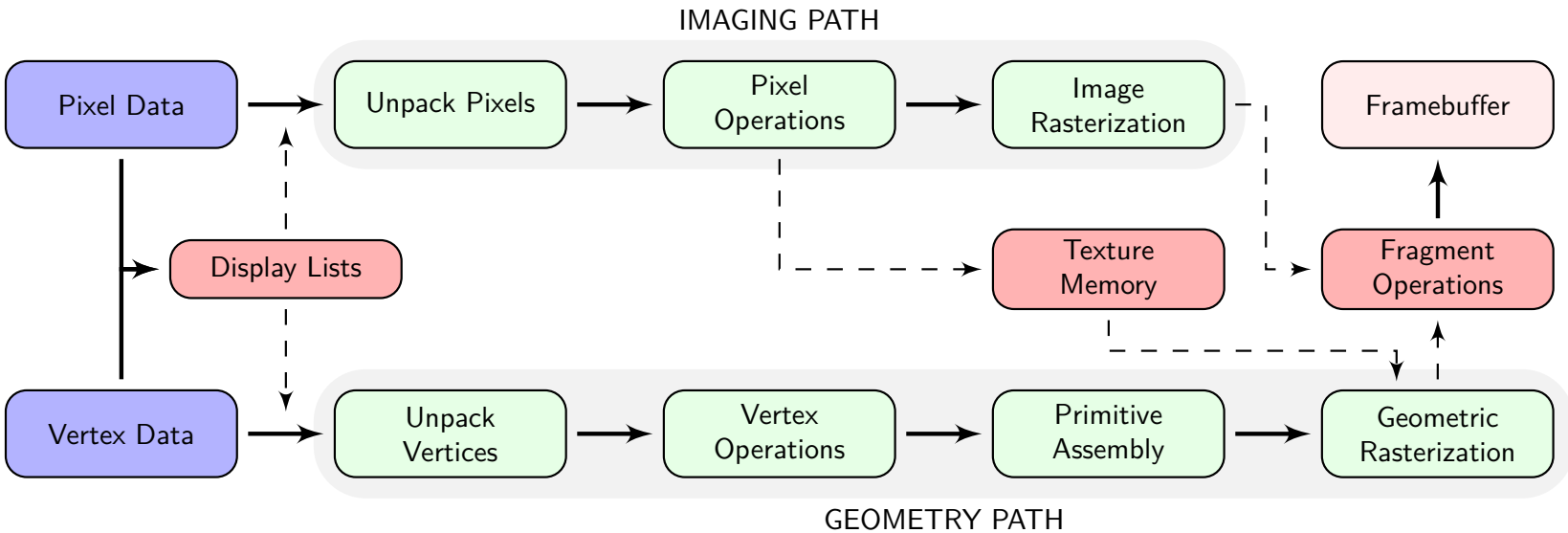
Potok renderowania



Rasteryzacja

Rasteryzacja *(rasterization)* – to konwersja kształtów geometrycznych i pikseli na fragmenty. Fragmenty przechowywane są w prostokątnej tablicy zawierającej informacje o kolorze, głębokości itp. Jeden fragment odpowiada jednemu pikselowi w buforze ramki.

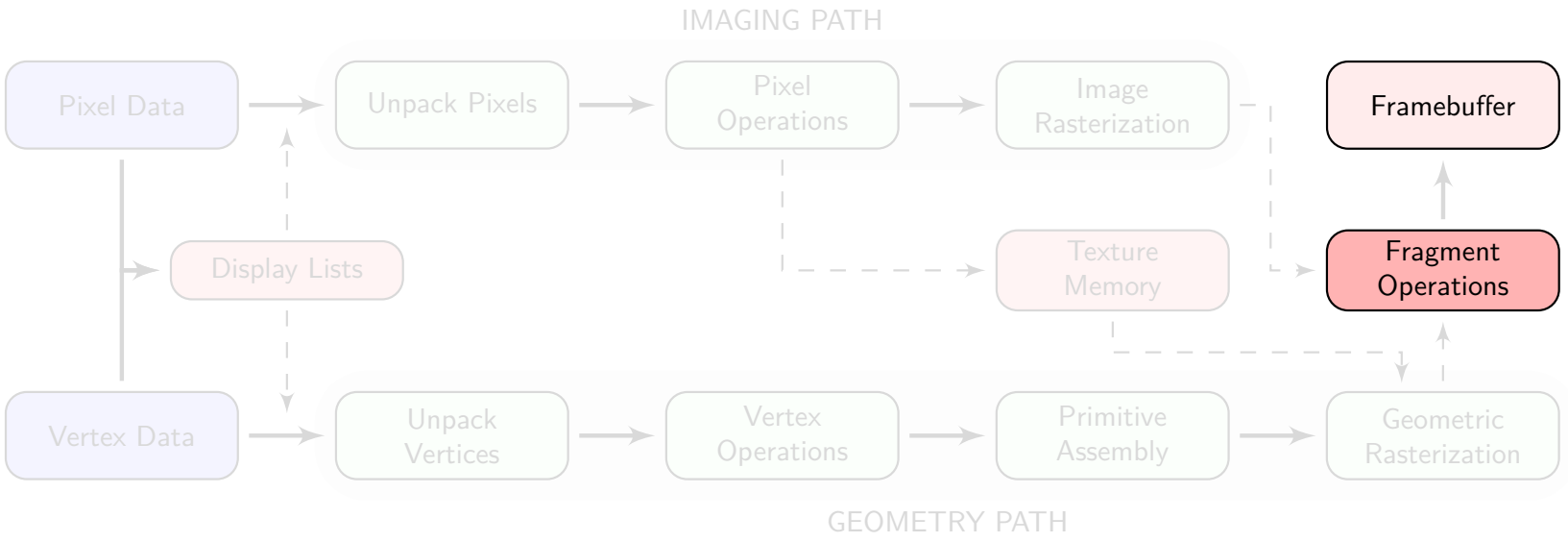
Potok renderowania



Operacje na fragmentach

Operacje na fragmentach *(fragment operation)* – pierwszym etapem jest generowanie tekseli na podstawie map bitowych tekstur. Następnie następuje kalkulacja mgły, testy (obcinanie, alfa, głębokości, itd.), blending, operacje bitowe. Wynikowe piksele umieszczane są w **buforze ramki**.

Potok renderowania



Operacje na fragmentach

Operacje na fragmentach *(fragment operation)* – pierwszym etapem jest generowanie tekseli na podstawie map bitowych tekstur. Następnie następuje kalkulacja mgły, testy (obcinanie, alfa, głębokości, itd.), blending, operacje bitowe. Wynikowe piksele umieszczane są w **buforze ramki**.

GLU (OpenGL Utility Library)

Biblioteka pomocnicza zawiera różne funkcje związane z generowaniem obiektów, operacjami na macierzach, przekształceniami perspektywy, teksturami, kwadrykami, obsługą powierzchni NURBS itp.

GLUT (OpenGL Utility Toolkit)

Biblioteka użytkowa, której głównym zadaniem jest zarządzanie systemem *wejścia/wyjścia* oraz generowanie i zarządzanie **oknami**. GLUT rozszerza możliwości standardowej biblioteki OpenGL, w znacznym stopniu usprawnia przenoszenie kodu na różne platformy systemowe oraz ułatwia proces uczenia się OpenGL'a.

GLUT (OpenGL Utility Toolkit)

Biblioteka użytkowa, której głównym zadaniem jest zarządzanie systemem *wejścia/wyjścia* oraz generowanie i zarządzanie **oknami**. GLUT rozszerza możliwości standardowej biblioteki OpenGL, w znacznym stopniu usprawnia przenoszenie kodu na różne platformy systemowe oraz ułatwia proces uczenia się OpenGL'a. Biblioteka zawiera funkcje do tworzenia i obsługi prostego interfejsu użytkownika niezależnego od platformy systemowej, generowaniem podstawowych kształtów brył itp. *Dawniej funkcje tę spełniała biblioteka AUX*.

Bibliografia



OpenGL Overview

http://www.opengl.org/about/overview/



Dave Shreiner

OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1 (7th Edition)

Addison-Wesley Professional, 2009



Jackie Neider, Tom Davis, Mason Woo

The OpenGL Programming Guide - The Redbook

http://www.opengl.org/documentation/red_book/



Richard S. Wright jr, Michael Sweet

OpenGL: księga eksperta

Helion, 1999



Kevin Hawkins, Dave Astle

OpenGL. Programowanie gier

Helion, 2003

Koniec części 1



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



Politechnika Świętokrzyska

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

" Program Rozwojowy Potencjału Dydaktycznego Politechniki Świętokrzyskiej w Kielcach: *kształcenie w nowoczesnych obszarach techniki* "

Program Operacyjny Kapitał Ludzki, Priorytet IV Szkolnictwo wyższe i nauka, Działanie 4.1. Wzmocnienie i rozwój potencjału dydaktycznego uczelni oraz zwiększenie liczby absolwentów kierunków o kluczowym znaczeniu dla gospodarki opartej na wiedzy, Poddziałanie 4.1.1. Wzmocnienie potencjału dydaktycznego uczelni.

Umowa UDA–POKL.04.01.01-00395/09-00