# Multiprocessor Graphic Rendering

**Kerey Howard**

EEL 6897

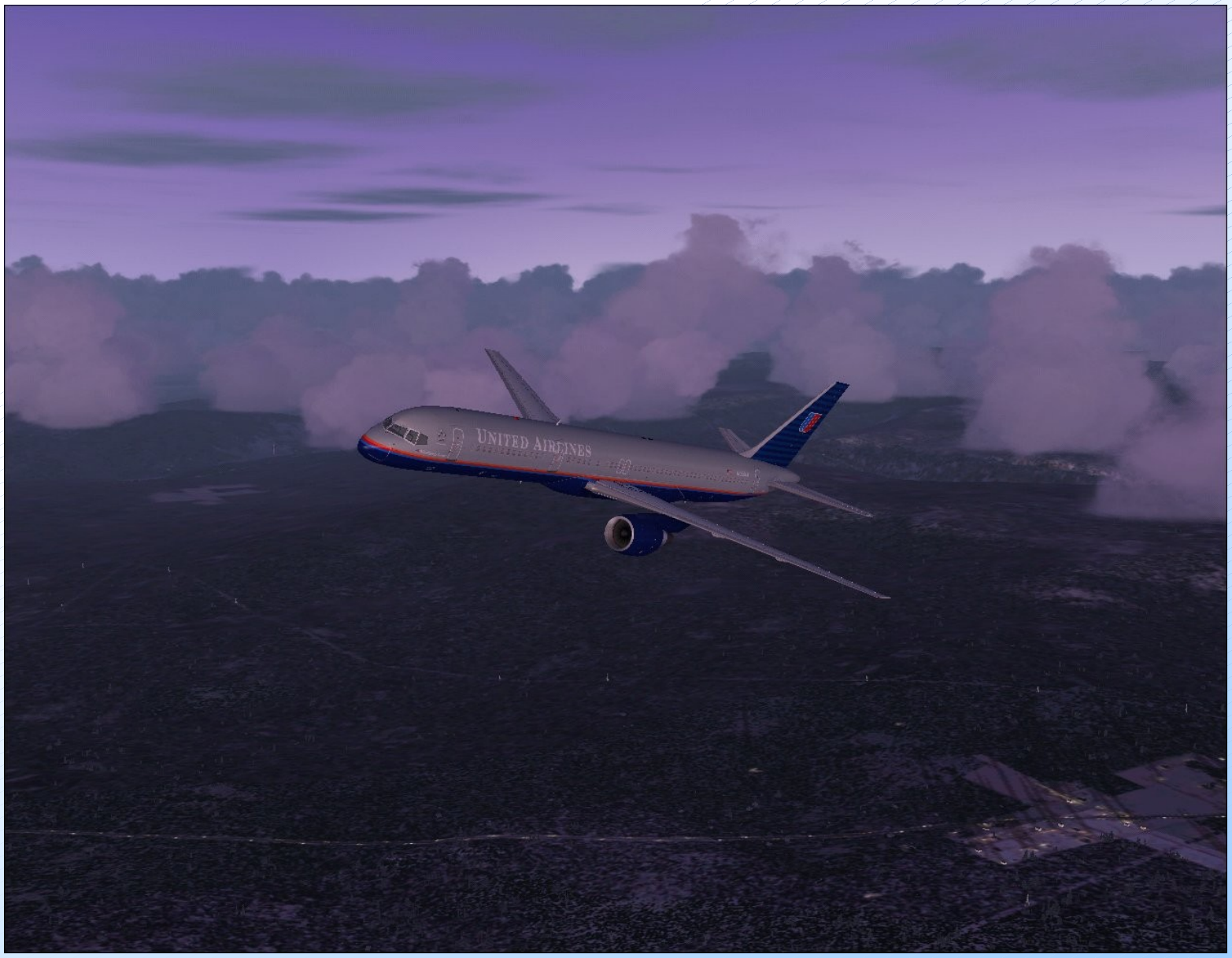
# Lecture Outline

- Real time Rendering Introduction
- Graphics API
- Pipeline
- Multiprocessing
- Parallel Processing
- Threading
- OpenGL with Java

# Real time Rendering Introduction

- Real-time:
  - Interactive
  - Range from 30Hz to 72Hz
- Rendering
  - Displaying Computer Graphics
  - Typically three-dimensional
- Examples:
  - Games
    - Half-life 2
    - Blizzard World of Warcraft
    - Microsoft Flight Simulator X
  - Simulators
    - Flight Safety (FAA flight simulation)
    - Lockheed Martin (DoD Air/Ground/Sea simulators)

# *Illustration: FSX Screen Capture*



# *Illustration: Halflife 2 ep2 Screen Capture*

# *Illustration: Halflife 2 Screen Capture*

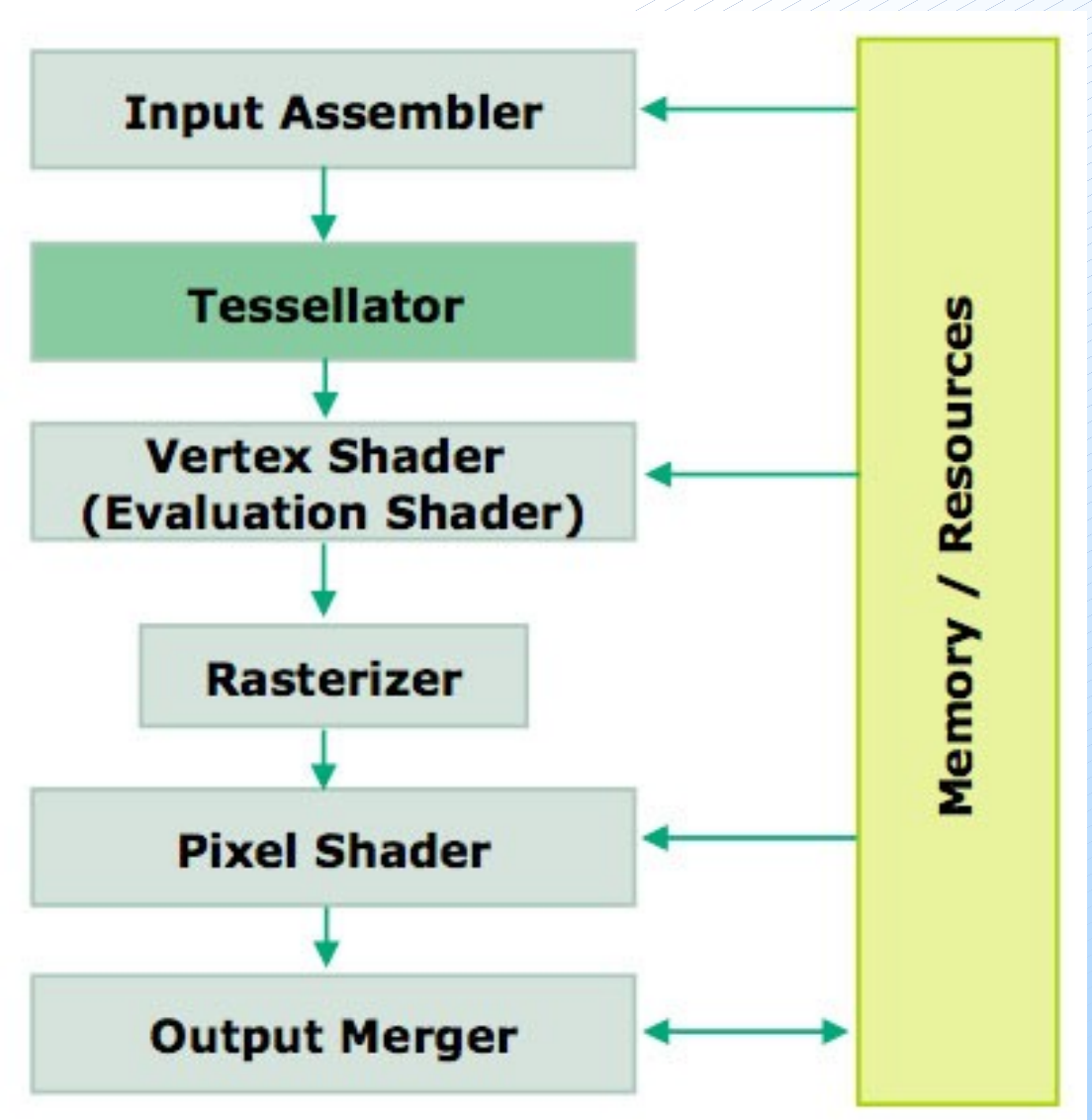
# Graphics API

- OpenGL
  - Open Graphics Library
  - http://www.opengl.org/
  - Native Multi OS support
  - Hardware Accelerated
  - GLSL Shader Language
- Direct 3D
  - Part of the Microsoft Direct X API package
  - http://www.microsoft.com/directx
  - Windows support only (DirectX 10 only on Vista)
  - Hardware Accelerated
  - HLSL Shader Language

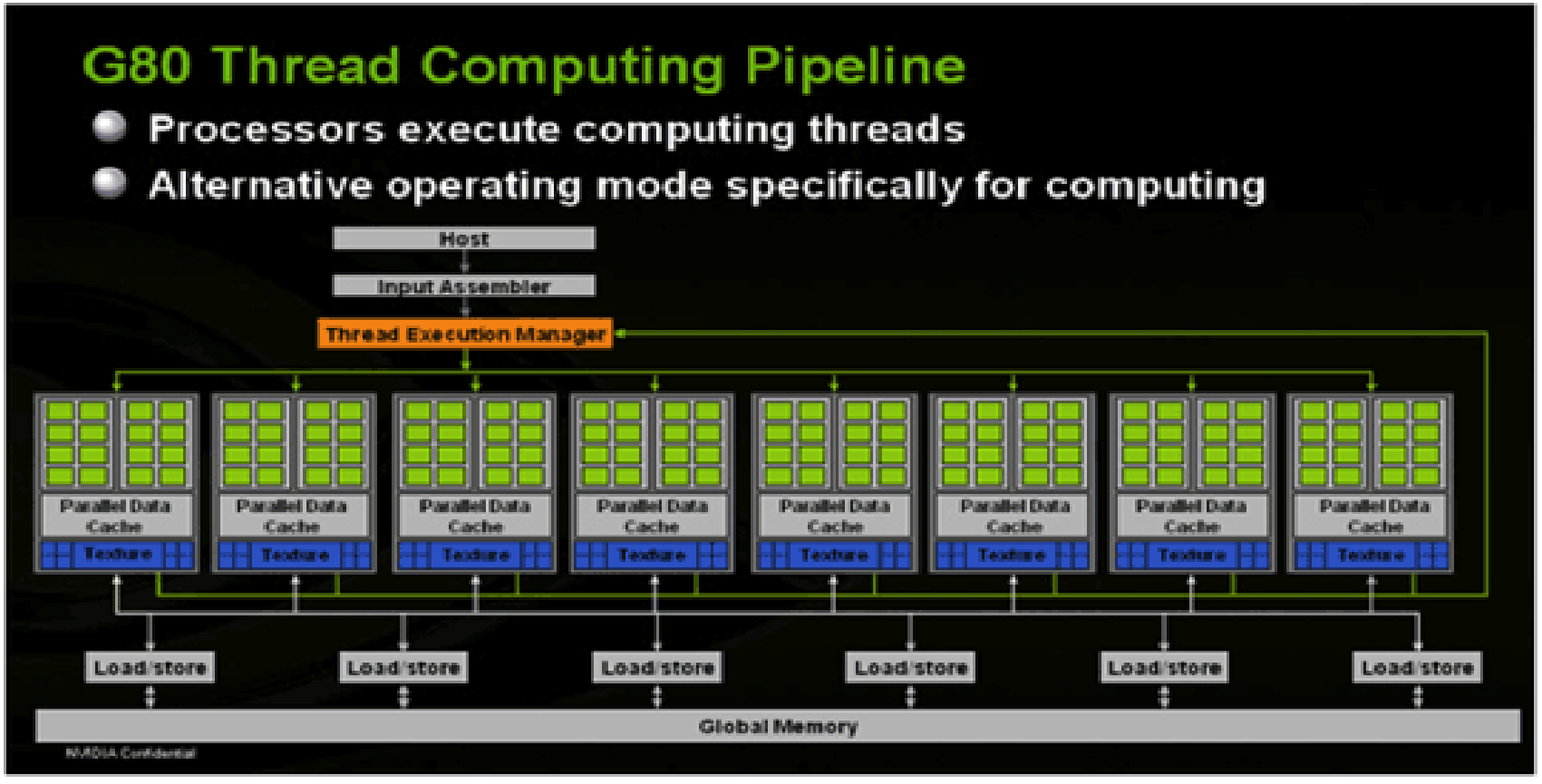
# Pipeline

- Render Pipeline
  - Slowest Stage determines Rendering Speed (fps)
- Application Stage
  - Input Control
  - Collision Detection
- Geometry Stage
  - Model & View Transform
  - Lighting
  - Projection
  - Clipping
  - Screen Mapping
- Rasterizer Stage
  - Assign Colors to all pixels
  - Anti-Aliasing, Z-buffer,and other filtering

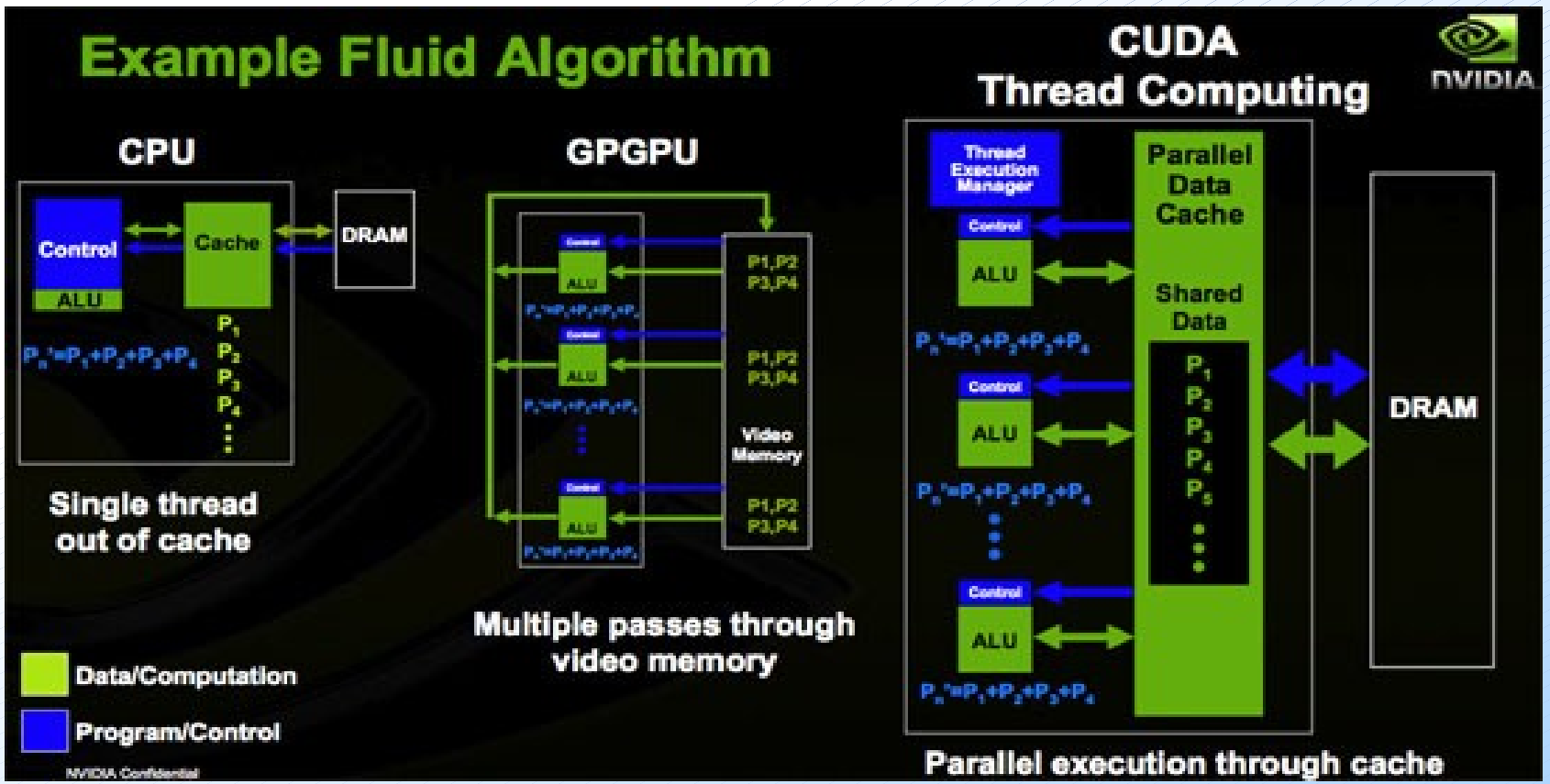
# *Illustration: Graphics Pipeline*



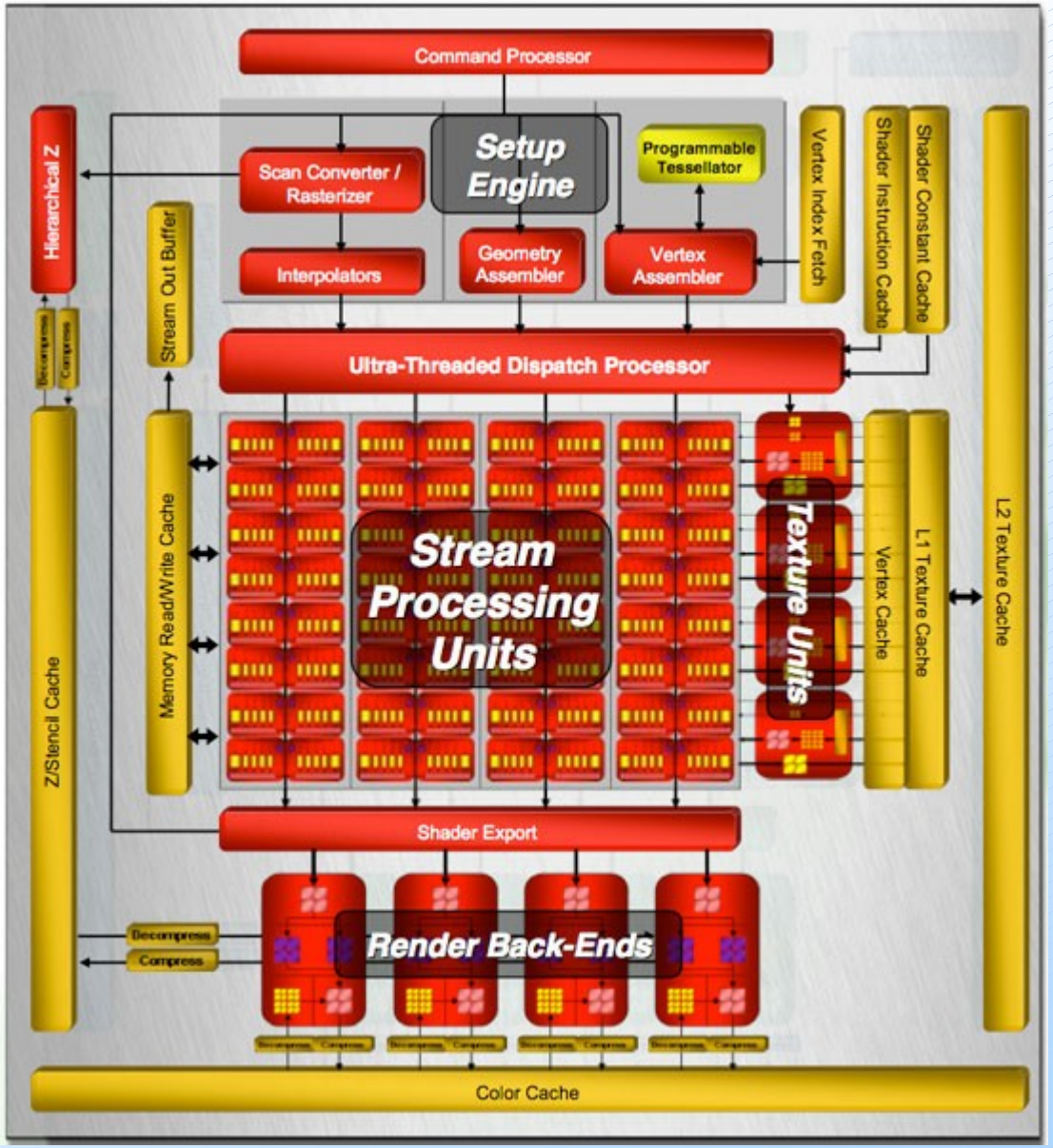
# *Illustration: Nvidia G80*



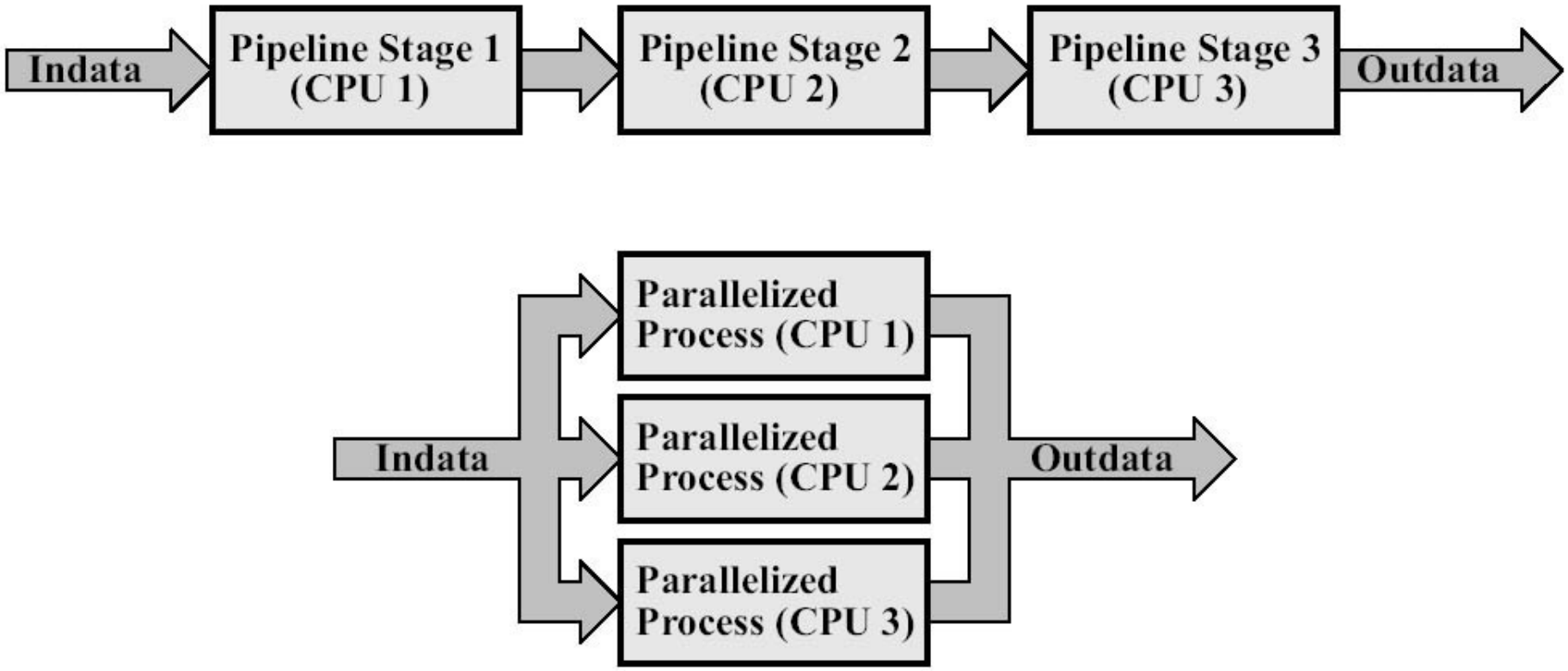
# Illustration: Nvidia G80



# Illustration: ATI 2900



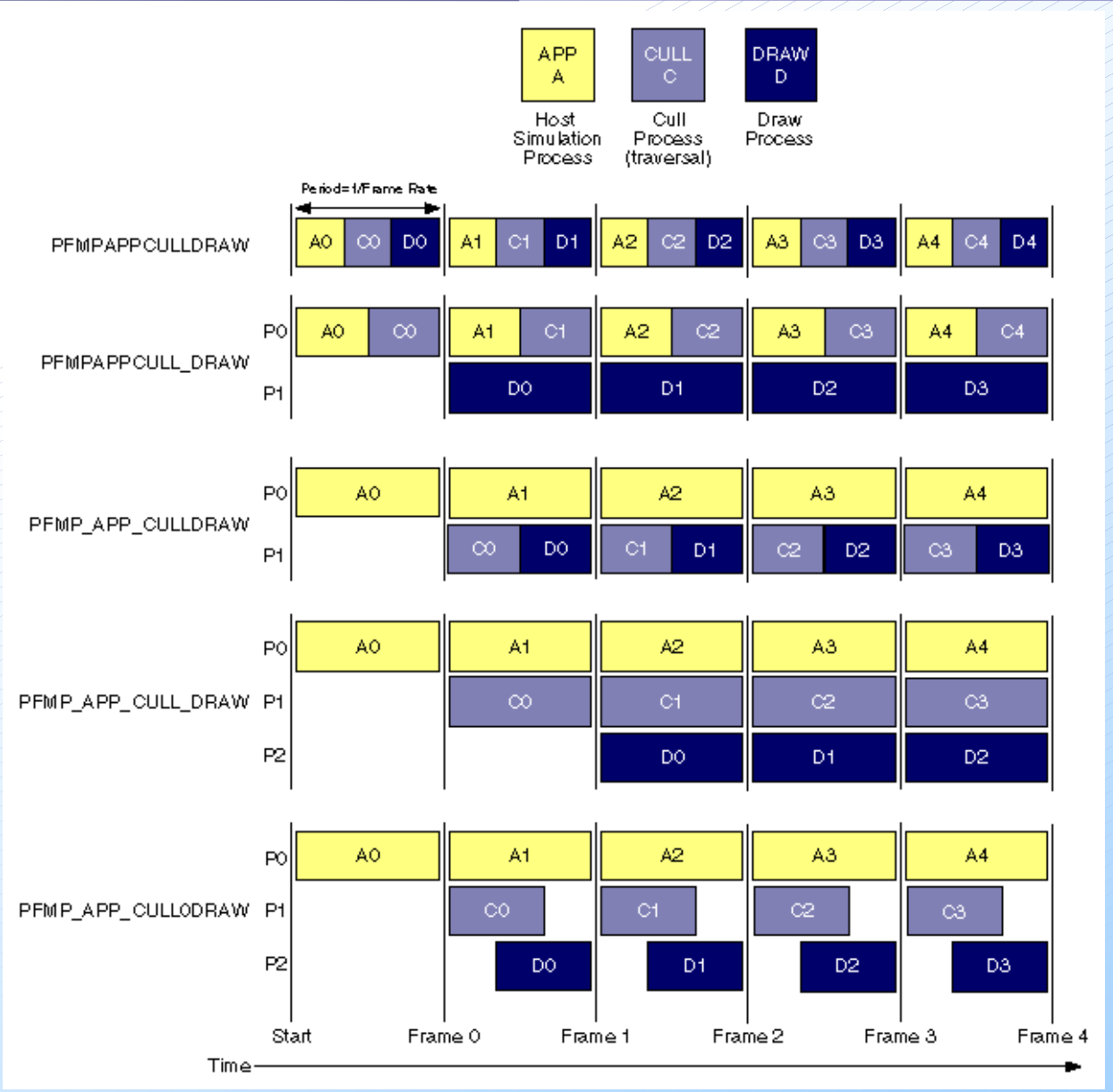
# *Illustration: Multiple Processor Utilization*



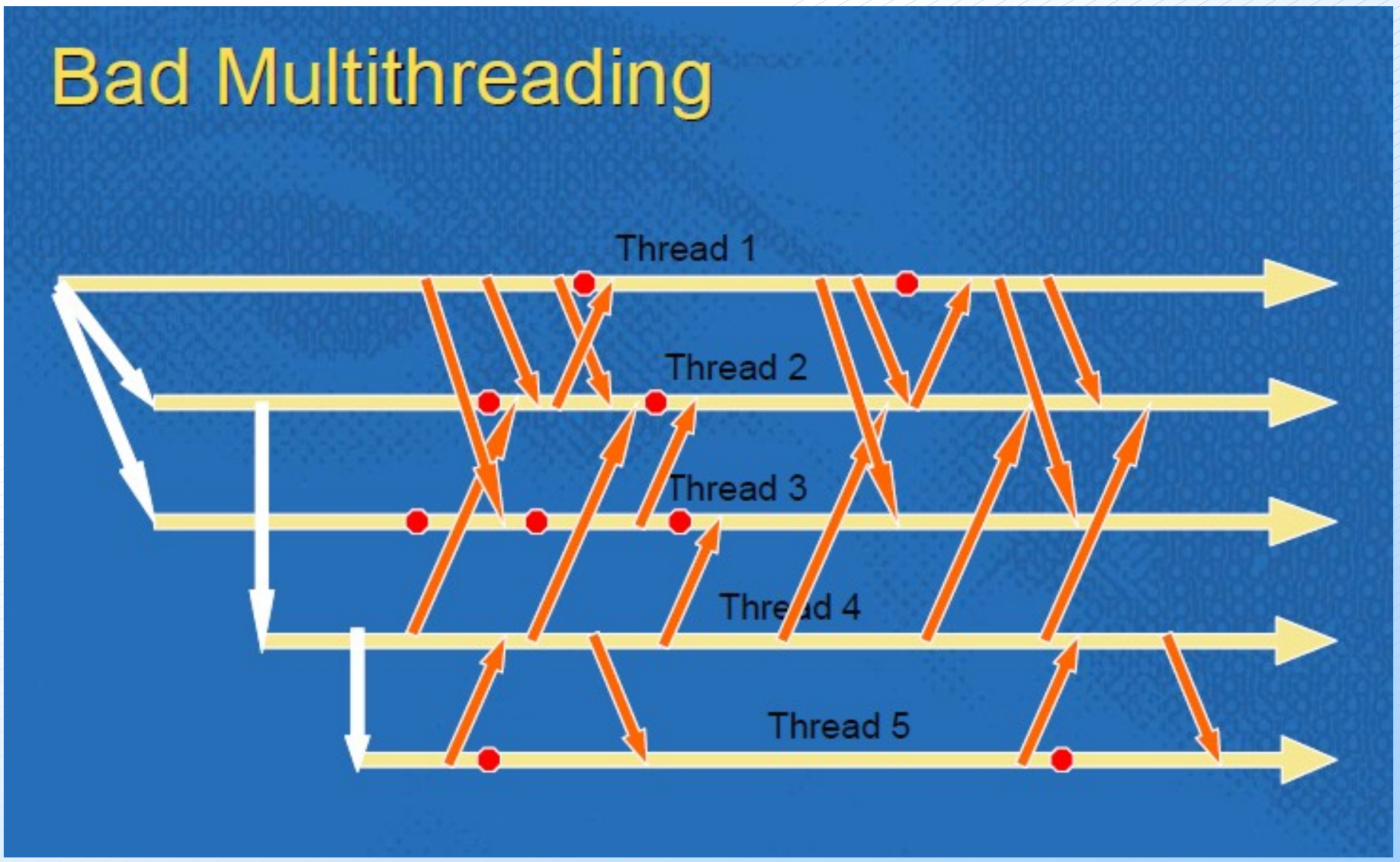
# Multiprocessing

- Temporal Parallelism
- Requires Multiple Processing Cores
  - No Hyperthreading
- Focus: Application Stage
  - APP is the Control
- Advantages:
  - Implementation (Stages are already divided)
  - Throughput is increased
  - Higher Frame rate
- Disadvantages:
  - Latency Increases
  - Synchronization Penalty

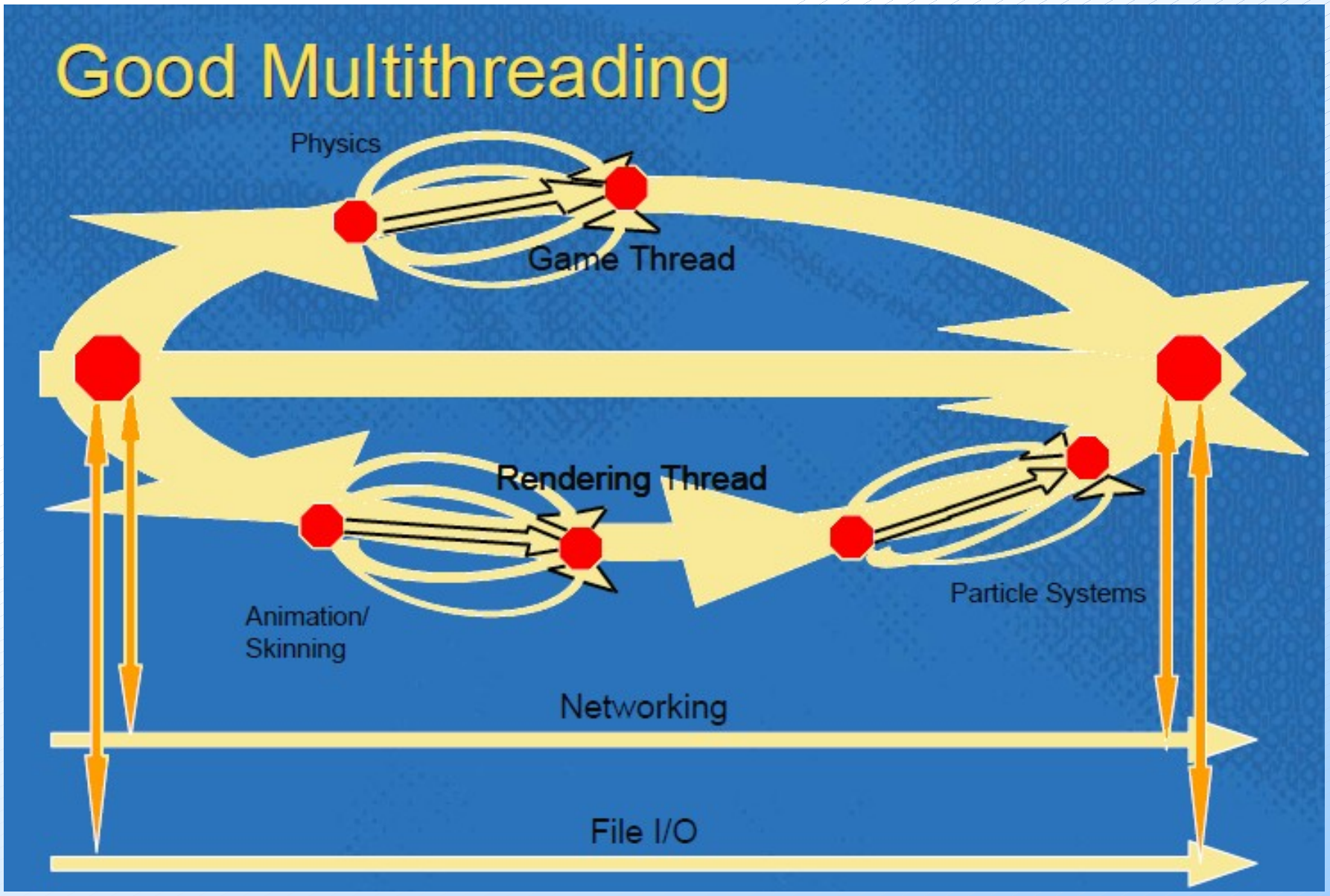
# Illustration: SGI Multiprocessing Models



# *Illustration: Threading approaches*



# *Illustration: Threading approaches*



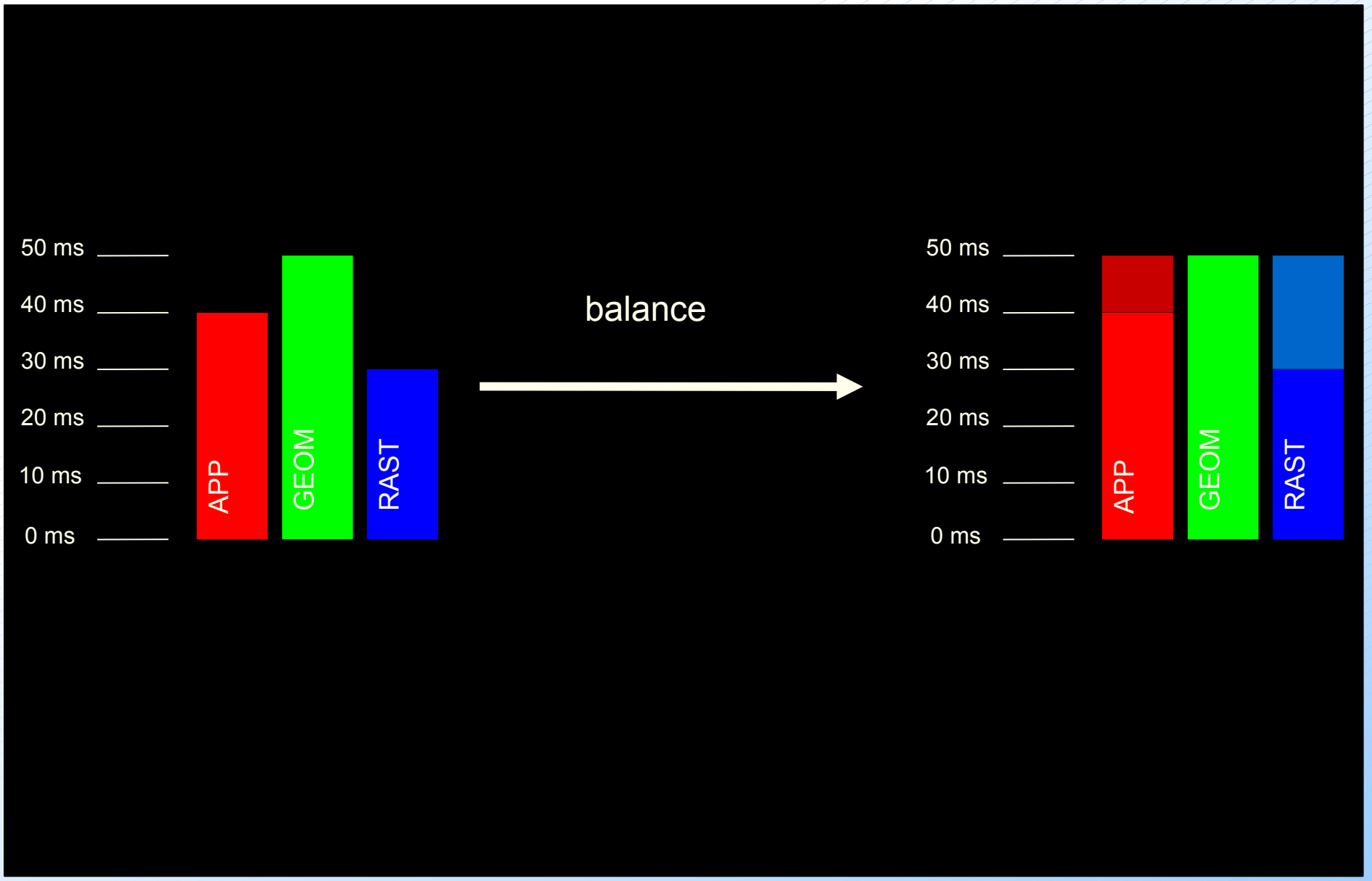
# Parallel Processing

- More complex algorithms
- Must allow for synchronization
- Typically Lower Latency
- Computes “work packages” in parallel
- Application Focused

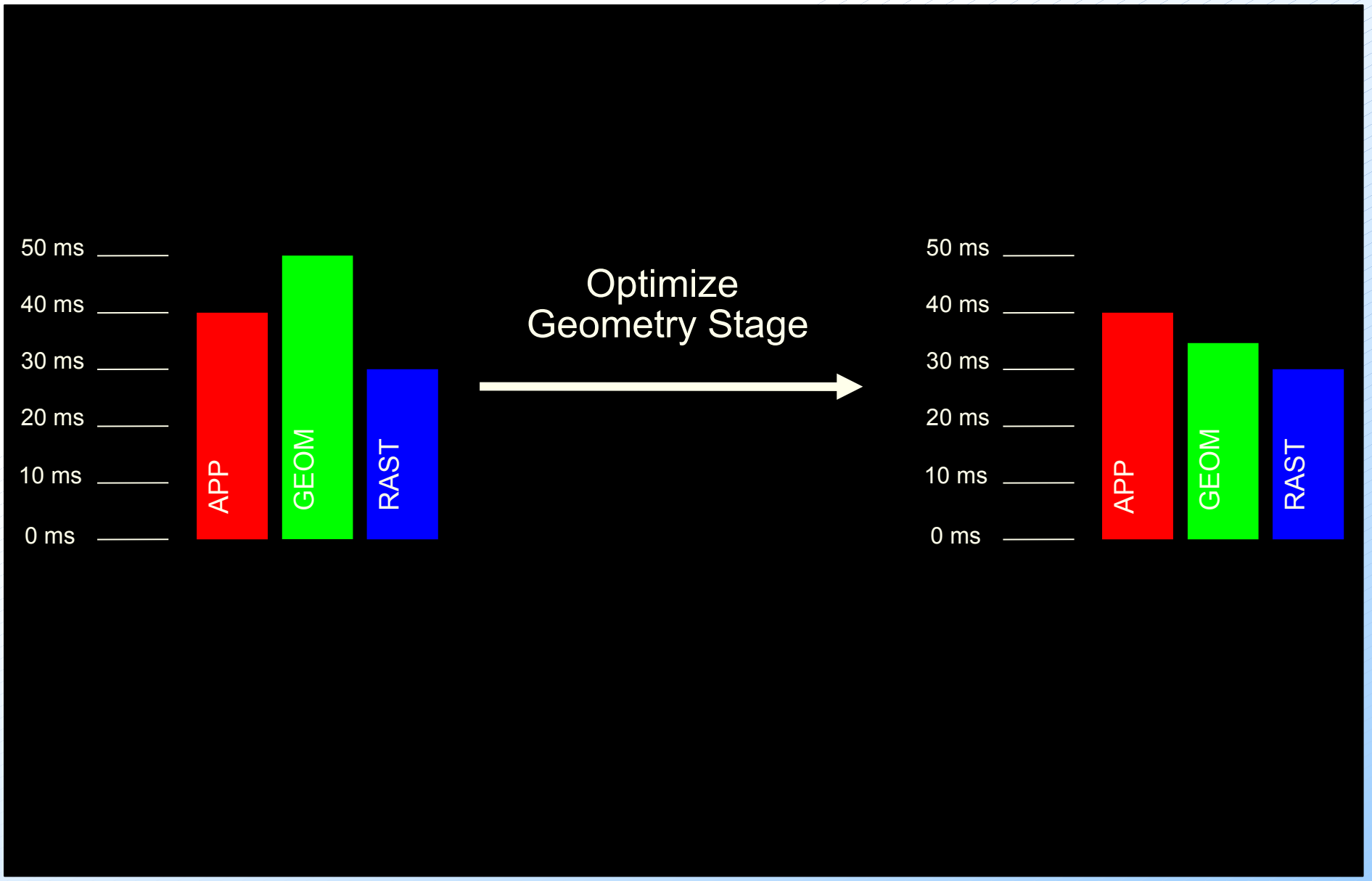
# Optimization

- Make it Run FIRST!
- Know the Architecture
- Find the Bottleneck (the slow stage)
- Optimize for Performance
  - Reduce latency
  - Increase Frame Rate
- Optimize for Quality
  - Make use of stall time

# *Illustration: Balancing the pipeline*



# Illustration: Optimization



# OpenGL: Java Implementation

- Can Java be used for Graphics Rendering?
- Can Java be used for Real-Time Games?

# OpenGL: Java Implementation

- Java is too slow for games programming.
- Java has memory leaks.
- Java is too high-level.
- Java application installation is a nightmare.
- Java isn't supported on games consoles.
- No one uses Java to write 'real' games.
- Sun Microsystems isn't interested in supporting Java gaming.

- Almost all of these are substantially wrong.

# OpenGL: Java Implementation

- JOGL, a Java binding for OpenGL
- JOAL, a binding for OpenAL (a 3D audio library)
- JInput, a game devices API

# OpenGL: JOGL

- Advantages of Java based OpenGL
  - GLUT use a single-threaded model for event processing
  - java.awt/JOGL libraries will spawn multiple threads to handle events
- Disadvantage:
  - You need to manage your threads to avoid DL
- https://jogl.dev.java.net/

# OpenGL: JOGL

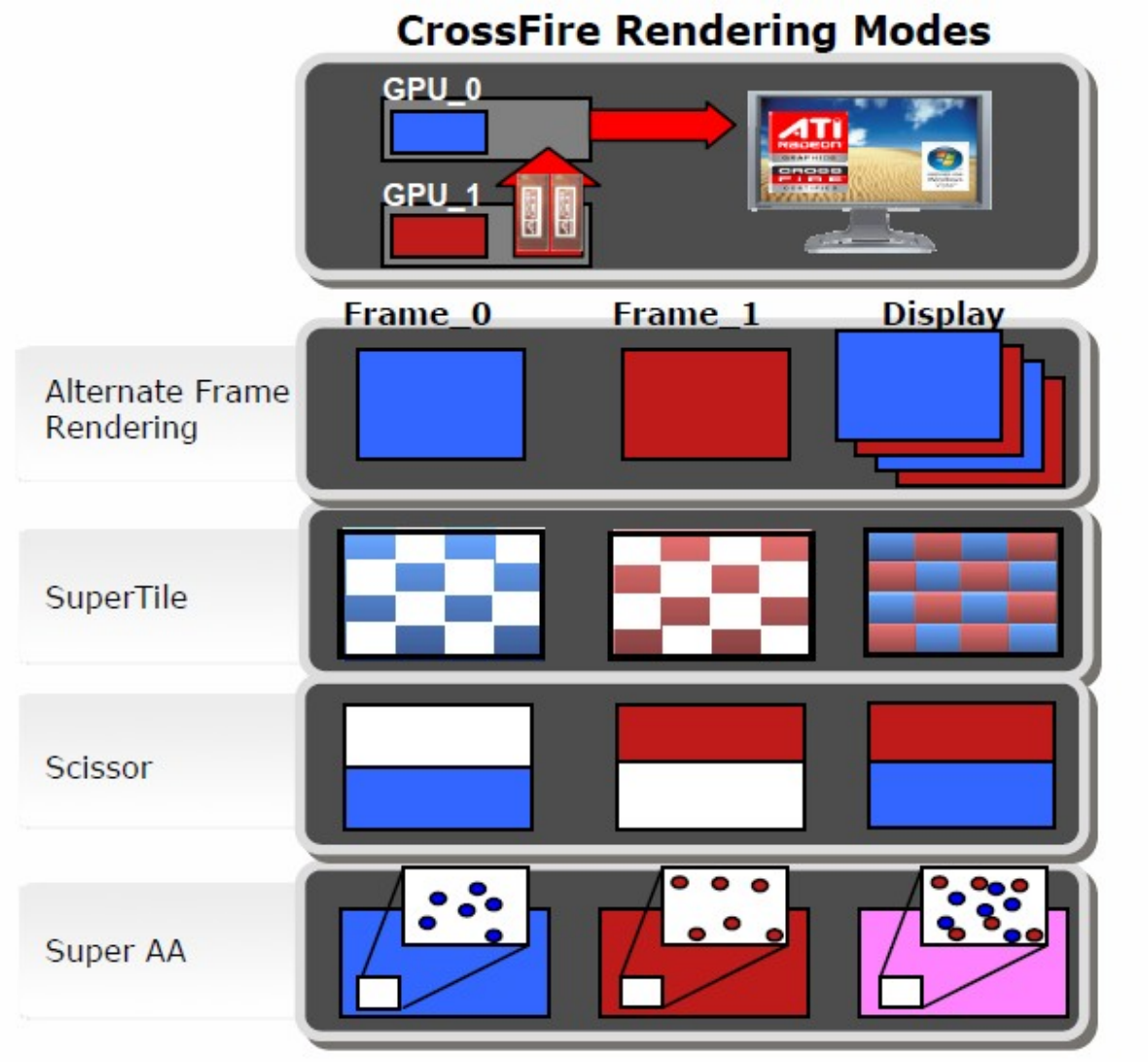
- Demos  - (If a recent Java is installed)

- https://jogl-demos.dev.java.net/

# Future Work

- Multidirectional Bus Speed
- Enhanced Multi-Core applications
  - CPU Based
  - GPU Based
- GPU Shaders  and Cube maps

# *Illustration: ATI Crossfire*



# References


- Akenine-Möller, T. REAL-TIME Rendering, A K Peters. Ltd.,2002
  http://www.realtimerendering.com
- Dawson, B. *Coding For Multiple Cores,* Microsoft Game Development,
  http://ati.amd.com/developer/brighton/07%20Coding%20for%20Multiple%20Cores.pdf
- Huddy, R. *ATI Radeon™HD 2000 SeriesTechnology*
  http://ati.amd.com/developer/AMDTechDay/2007/Architecture_Overview_RH.pdf
- Hwu, W., Kirk, D. *ECE 498 Programming Massively Parallel Processors*
  *http://courses.ece.uiuc.edu/ece498/al1/*
- SGI *OpenGL Performer Programmer's Guide*
  *http://techpubs.sgi.com/library/manuals/1000/007-1680-100/pdf/007-1680-100.pdf*
- *JOGL: A Beginner's Guide and Tutorial*
  *http://www.cs.umd.edu/~meesh/kmconroy/JOGLTutorial/*