A Closer Look At GPUs

By Kayvon Fatahalian and Mike Houston

Presented by Richard Stocker

Contents

- What is a GPU and Why do we need it?
- The Graphics Processing Pipeline
- Shader Programming
- Characteristics and Challenges
- GPU Architecture
- GPU vs CPU Architecture
- GPU on Chip Memory
- Convergence of CPUs and GPUs

What is a GPU?

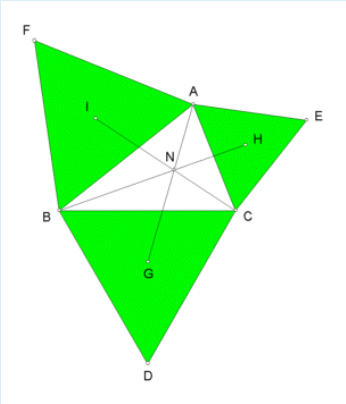
- GPU stands for Graphics Processing Unit
- A GPU is the main chip inside a computer which calculates and generates the positioning of graphics on a computer screen
- Today's graphical needs requires high performance GPUs to compute the required hundreds of gigaflops (FLoating point Operations Per Second) of modern day applications

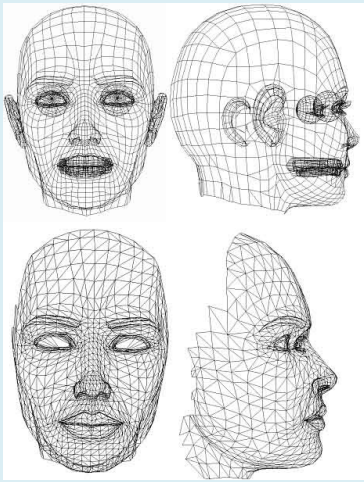
Graphics Processing Pipeline

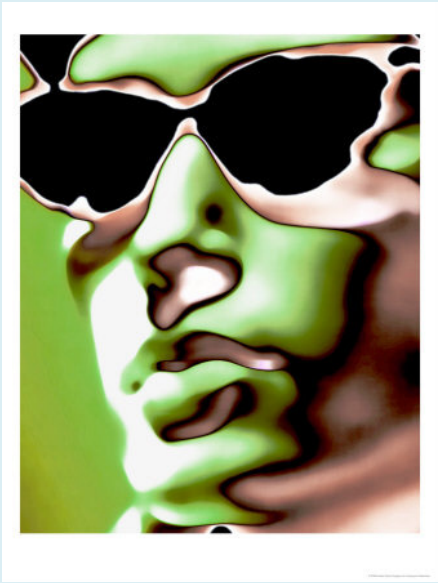
- Graphics systems need to have an appropriate balance between:
  - maximising performance
  - maintaining an expressive but simple interface for describing the graphics computations

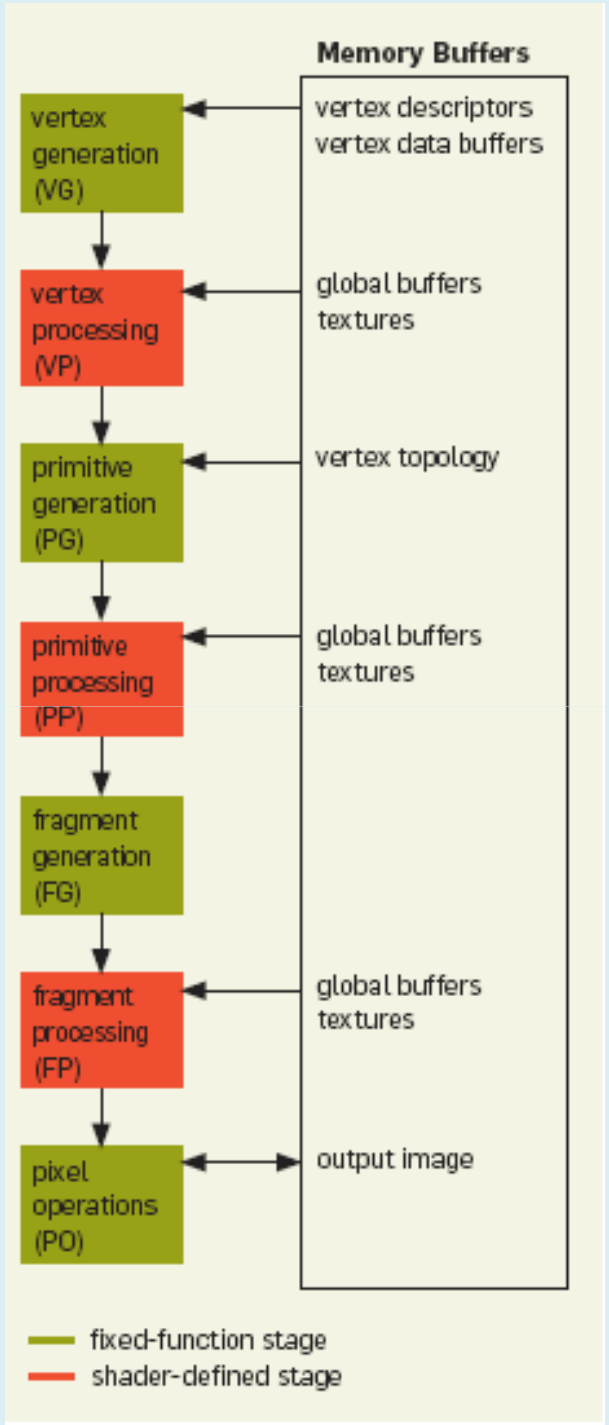
- Graphics processing pipeline is the method adopted by application programming interfaces Direct3D and OpenGL to maintain this balance

Pipeline Diagram









Shader Programming

- Shader programming is used in transformation effects in graphics such as generating facial expression, under water effects, soap bubbles etc.

- Shader functions are used during the processing stages of pipeline processing; vertex, primitive and fragment processing.

- The Shader function is a C-Like function (High-level language), they operate on a single input-entity and serially produce multiple output-entity records.

Characteristics and Challenges

- Opportunities for parallelism
  - Graphics computing there is vast potential areas for parallelism
  - However dynamic and fine grained dependencies complicates programming the parallelism
  - Fixed-function stages requires waiting until other functions have been processed creating parallelisation difficulties
  - dynamic memory addresses makes pre-fetching input data difficult
- Instruction stream sharing
  - Simultaneous shader invocations means GPU core designs need to take into account algorithms for multiple pipeline scheduling
- Extreme Variations in Pipeline Load
  - Applications need to alter the configuration of the pipelines to accommodate different types of graphics

GPU Architecture

- GPUs contain multiple cores that utilise hardware multithreading and SIMD (Single Input, Multiple Data).
- Having multiple cores and using SIMD processing means that GPUs have ALOT of Arithmetic Logic Units (ALU)
- GPUs utilize a wider SIMD width over CPUs, GPUs tend to have a width 32-64 compared to a the SIMD width of 4 chosen by CPU designers
- These wider SIMD allow cores to be packed densely with ALUs

GPU and CPU Architecture comparison

Type	Processor	Cores/Chip	ALUs/Core[3]	SIMD width	Max T[4]
GPUs	AMD Radeon HD 4870	10	80	64	25
	NVIDIA GeForce GTX 280	30	8	32	128
CPUs	Intel Core 2 Quad[1]	4	8	4	1
	STI Cell BE[2]	8	4	4	1
	Sun UltraSPARC T2	8	1	1	4

GPU On Chip Memory

- The huge parallelism of the GPU architecture lays a heavy burden on memory and bandwidth of data transmission, prompting GPU designers to implement high bandwidth with a cost of high latency data access.
- GPUs however only have small read-only caches, which only purpose is to filter requests from the memory controller and reduce bandwidth on main memory.
- GPUs do contain large on chip storage to hold entity streams, execution contexts and thread data.

GPU and CPU Convergence

- The parallel nature of graphics processing has allowed the design of GPUs to execute large numbers of operations all in parallel
- CPUs still have to handle the serial nature of many programs existing today, so can't commit to full time massive parallelism
- As programmers are now likely to start developing programs which can be processed on massively parallel levels, the architectures of GPU and CPUs will eventually converge

Summary

- GPUs are Graphics processing units designed to bring bright and lively images to our screens
- GPUs need to strike a balance between maximising performance and maintaining an expressive but simple interface
- GPUs operate using shader Programming to construct the graphical data into pixels on the screen
- GPUs operate on a massively parallel level containing multiple cores and SIMD width with small cache space to minimise thrashing and reduce bandwidth on main memory
- Technological differences between GPUs and CPUs will ultimately fade away when software is developed to utilise these massively parallel chip designs