GPU Architecture

GPU

- What is GPGPU?
  - General-Purpose computing on a Graphics Processing Unit
  - Using graphic hardware for non-graphic computations

- Prefect for massive parallel processing on data paralleled applications

GPU vs CPU

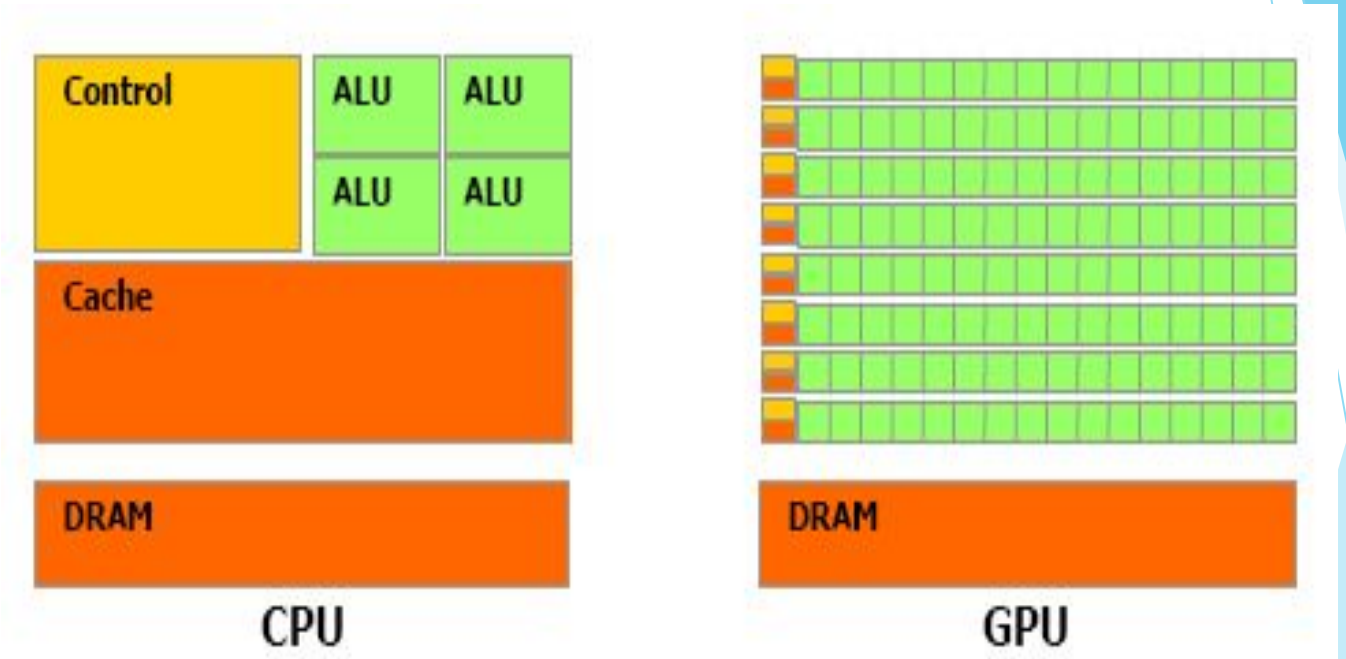
- A GPU is tailored for highly parallel operation while a CPU executes programs serially
- For this reason, GPUs have many parallel execution units and higher transistor counts, while CPUs have few execution units and higher clockspeeds
- GPUs have much deeper pipelines (several thousand stages vs 10-20 for CPUs)
- GPUs have significantly faster and more advanced memory interfaces as they need to shift around a lot more data than CPUs

Memory interfaces

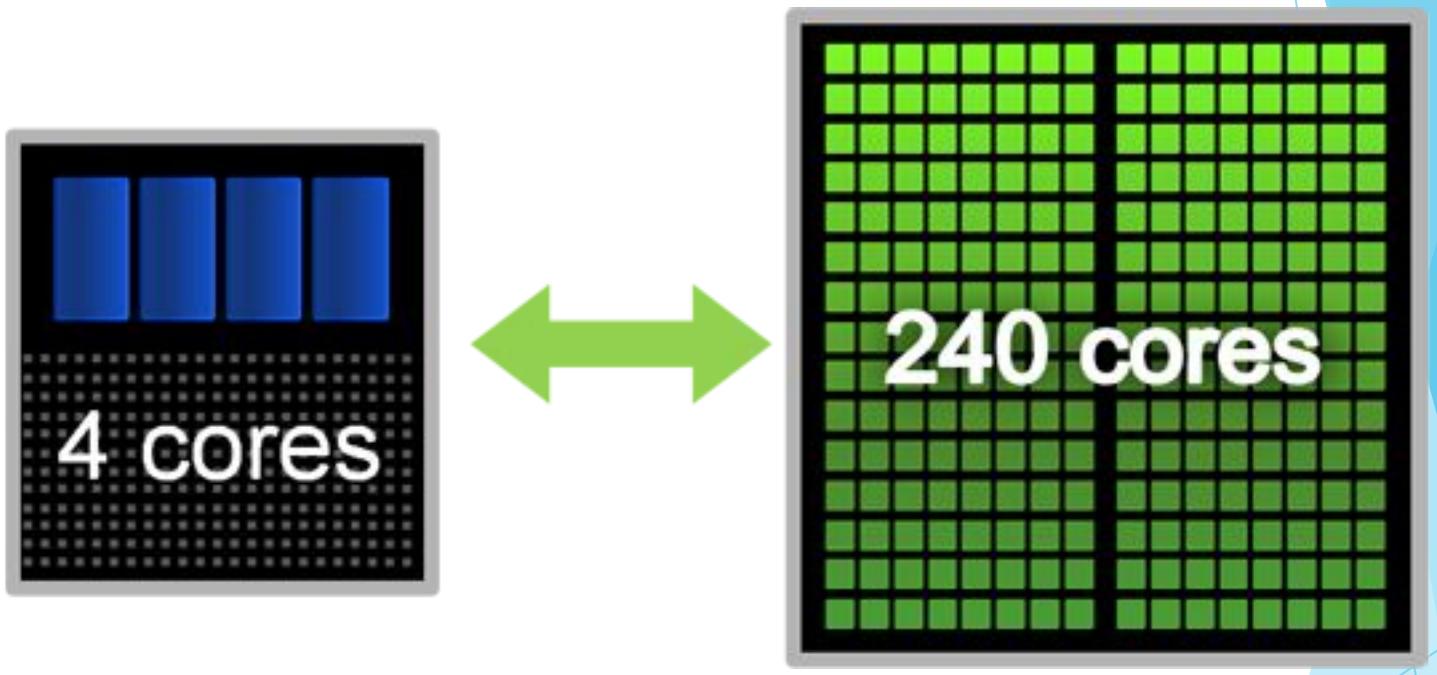
- Available Memory Bandwidth in Different Parts of the Computer System

Bandwidth	Component
GB/sec 35	GPU Memory Interface
GB/sec 8	PCI Express Bus
GB/sec 6.4	CPU Memory Interface

CPU vs. GPU - Hardware



CPU vs. GPU - Hardware



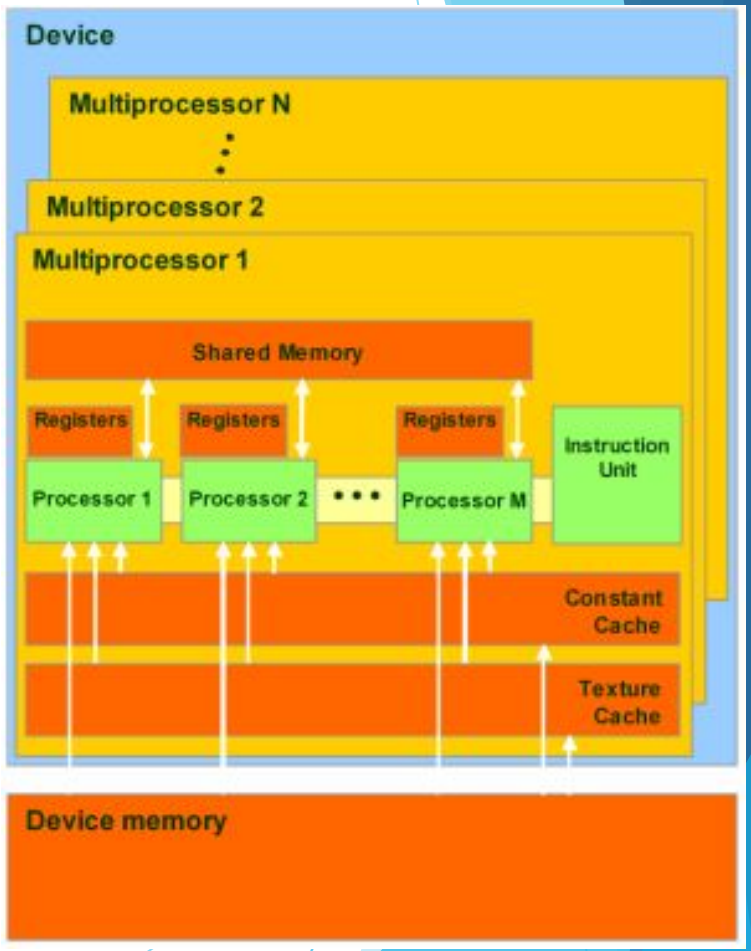
Multi-core vs. Many-core

Memory Architecture

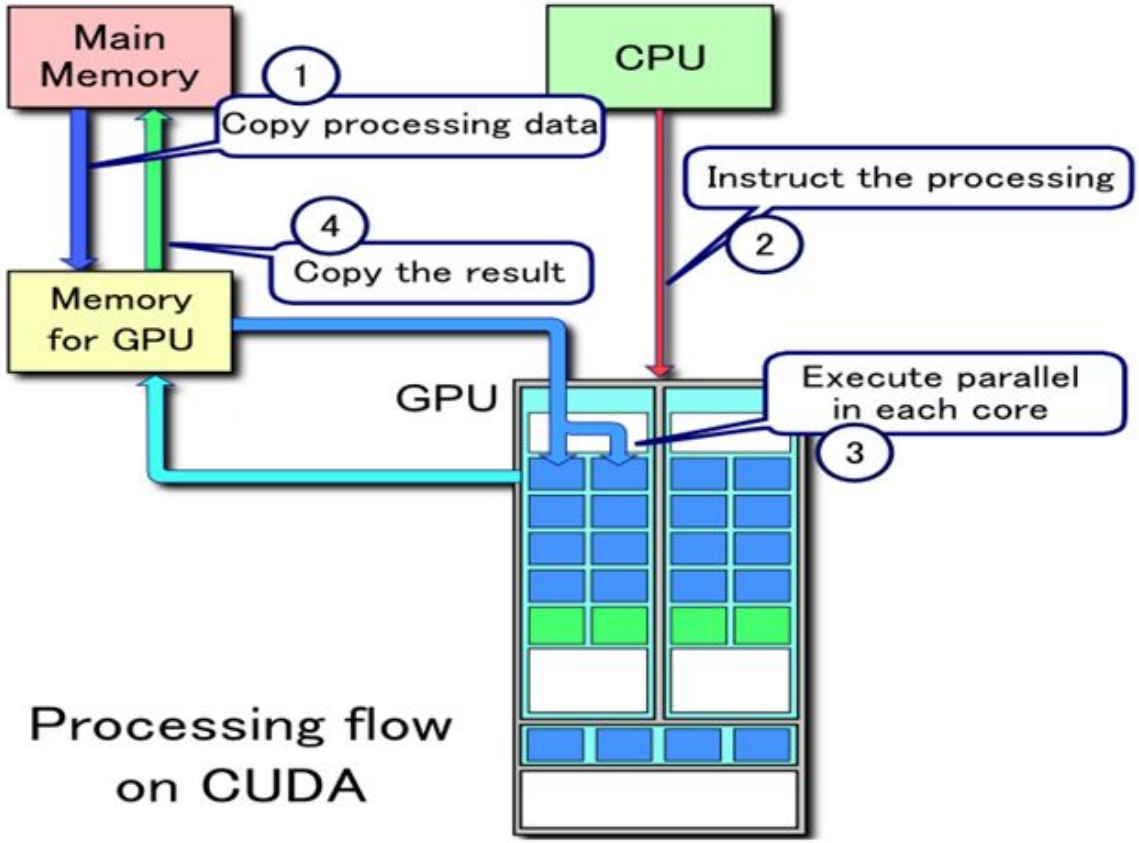
Fastest Memory =

Registers & Shared Memory

Lowest Memory = Global Memory



CPU & GPU Connection

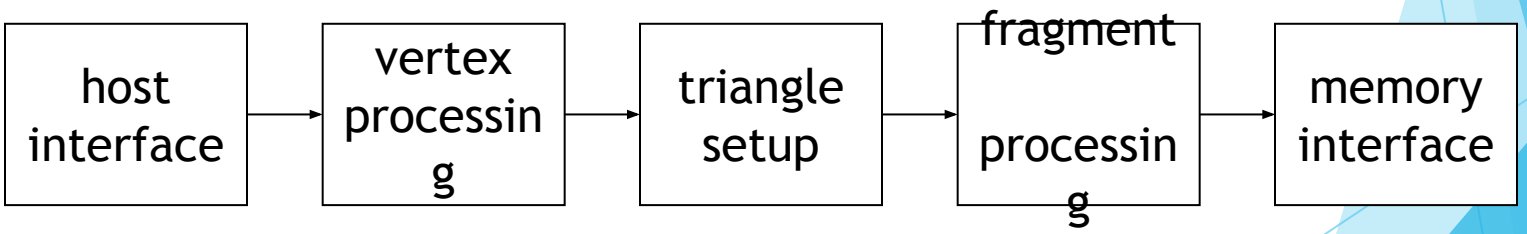


CUDA is **a parallel computing platform and programming model developed by Nvidia for general computing on its own GPUs**

The GPU pipeline

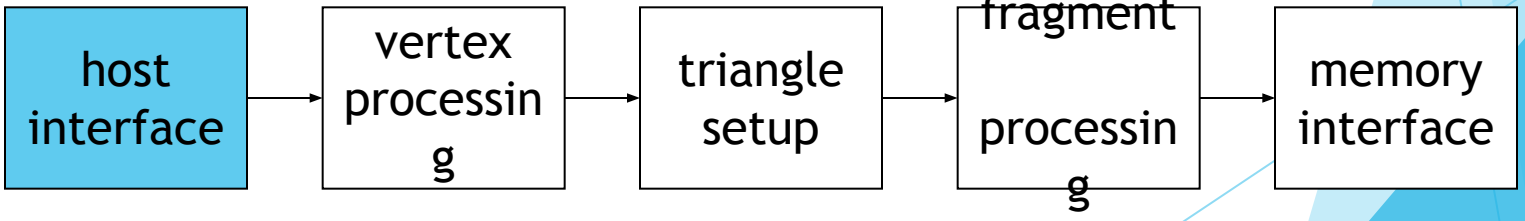
- The GPU receives geometry information from the CPU as an input and provides a picture as an output

- Let's see how that happens



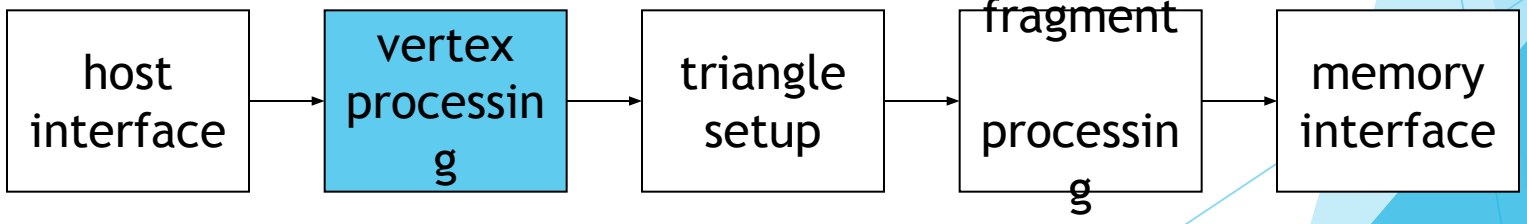
Host interface

- The host interface is the communication bridge between the CPU and the GPU
- It receives commands from the CPU and also pulls geometry information from system memory
- It outputs a *stream* of vertices in object space.



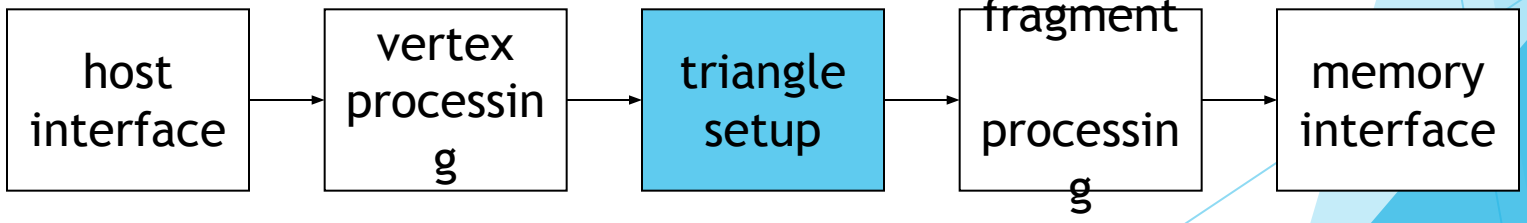
Vertex Processing

- The vertex processing stage receives vertices from the host interface in object space and outputs them in screen space



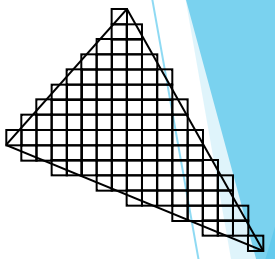
Triangle setup

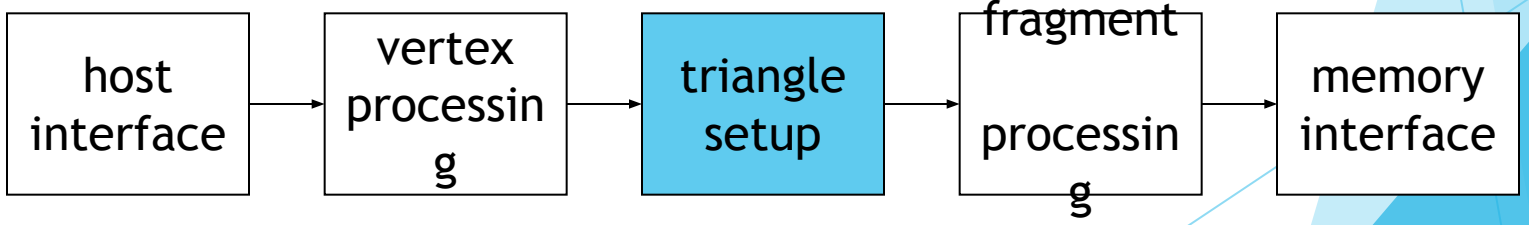
- In this stage geometry information becomes raster information (screen space geometry is the input, pixels are the output)



Triangle Setup

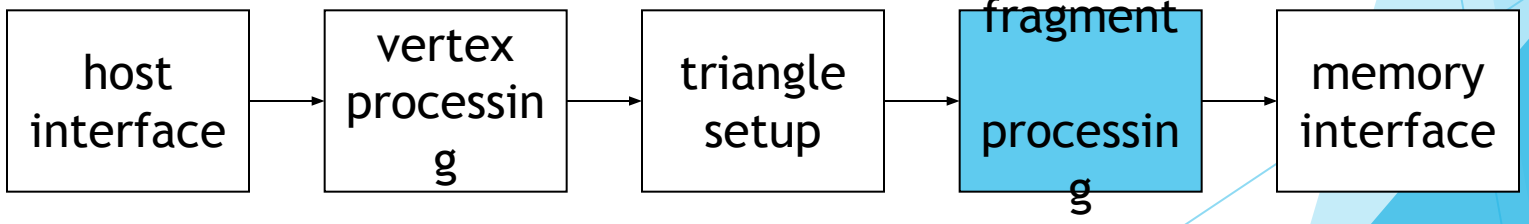
- A fragment is generated if and only if its center is inside the triangle





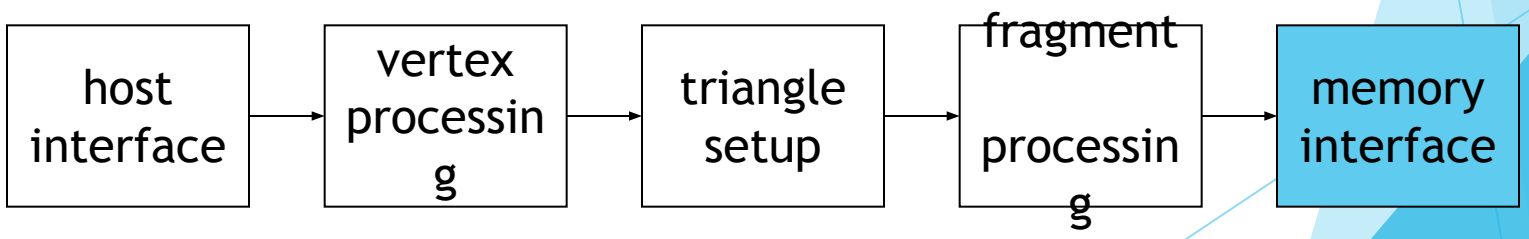
Fragment Processing

- Each fragment provided by triangle setup is fed into fragment processing as a set of attributes, which are used to compute the final color for this pixel



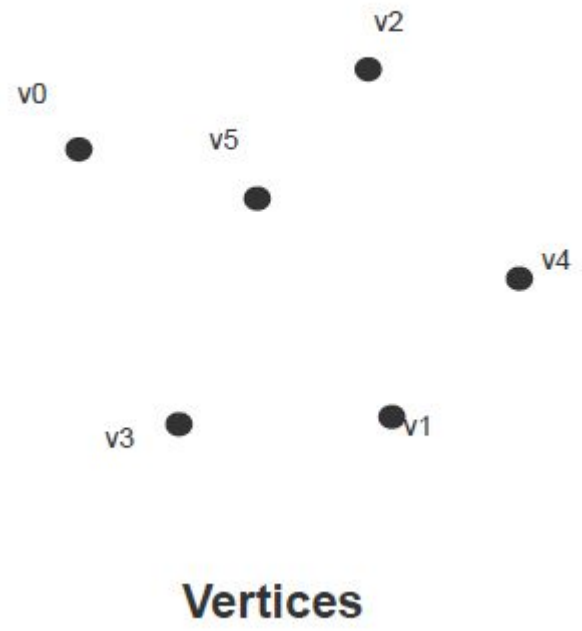
Memory Interface

- Fragment colors provided by the previous stage are written to the framebuffer
- Before the final write occurs, some fragments are rejected by the zbuffer, stencil and alpha tests



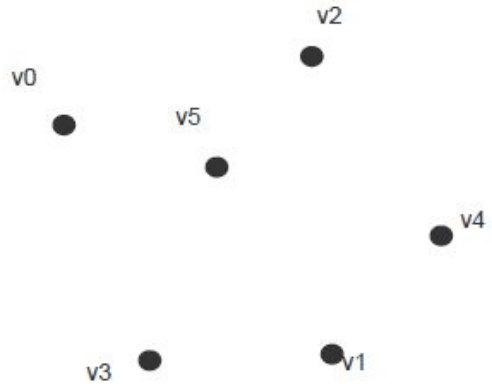
Vertex processing

Vertices are transformed into “screen space”

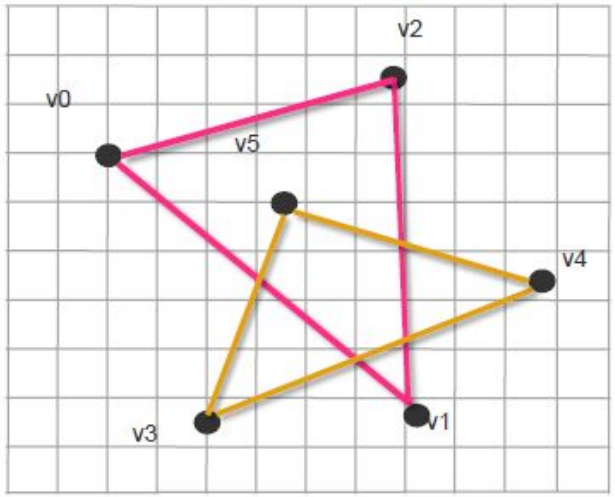


Triangle processing

Then organized into primitives that are clipped and culled…

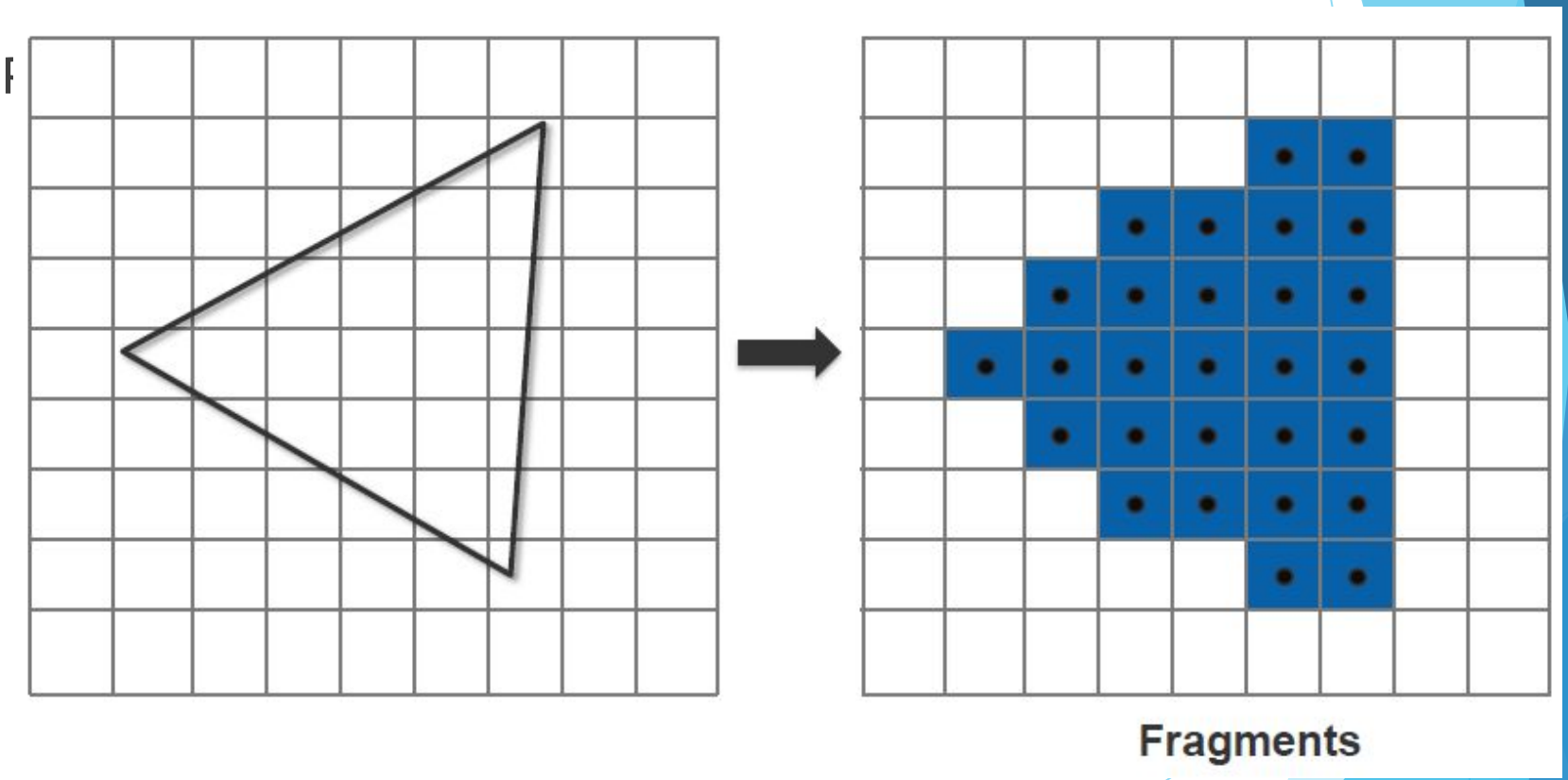


Vertices

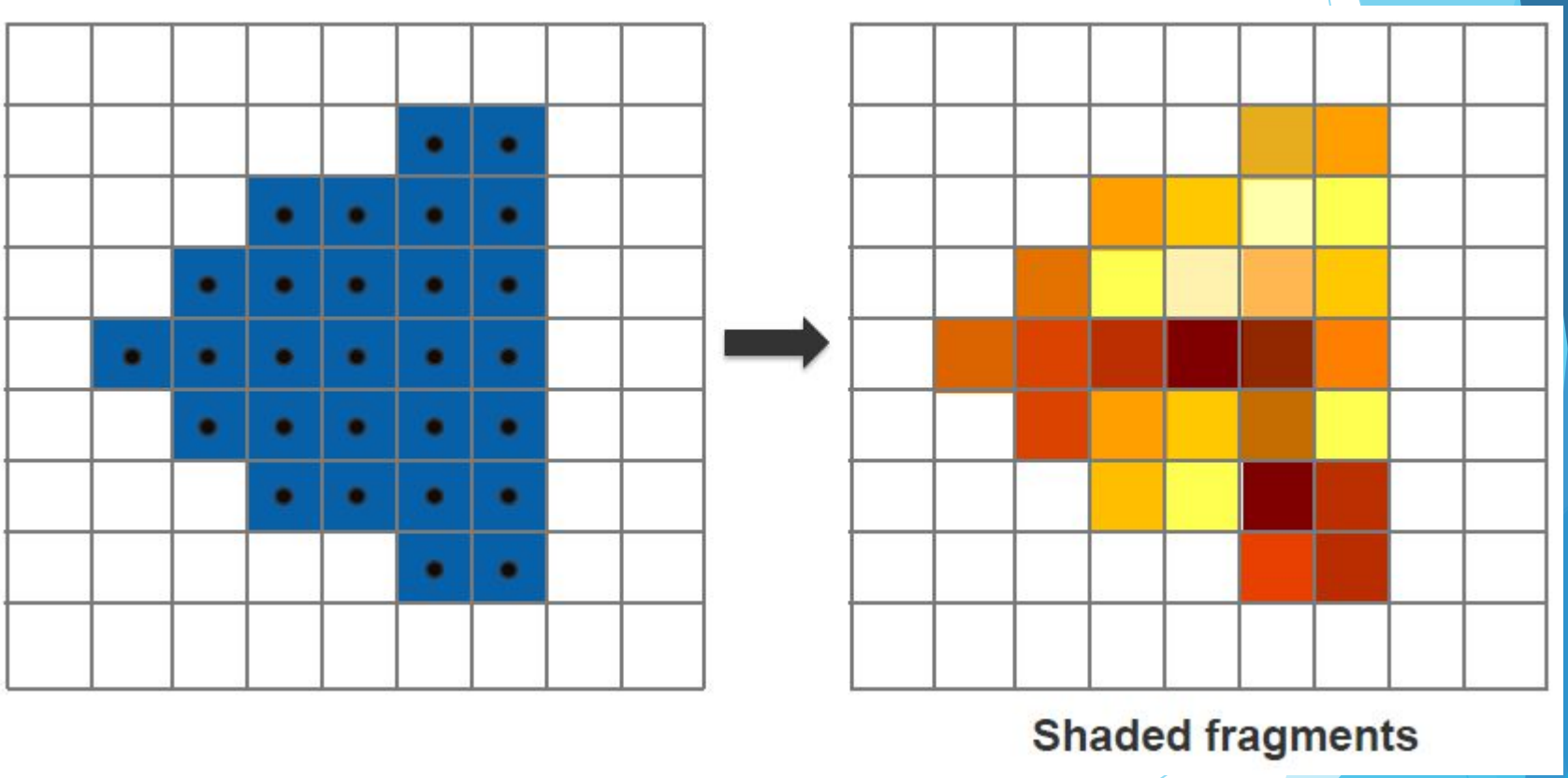


Primitives (triangles)

Rasterization

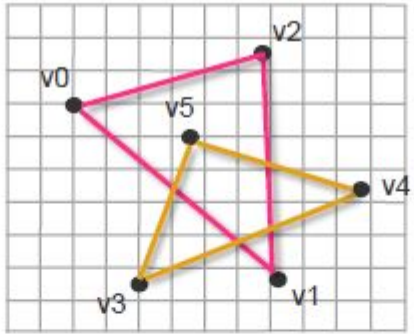


Fragment processing

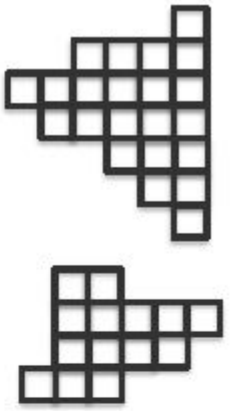


Pipeline entities

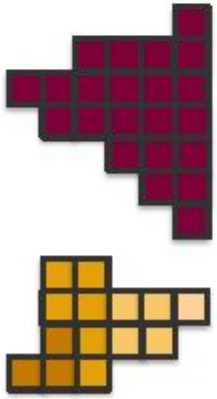
Vertices



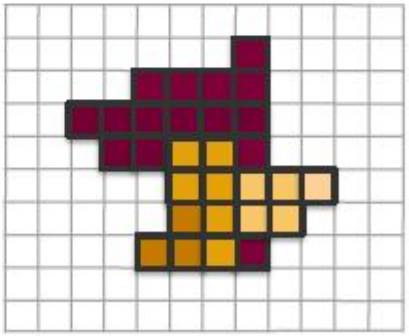
Primitives



Fragments



Fragments (shaded)



Pixels