Computer Graphics Hardware
An Overview
Graphics System

Input devices

CPU/Memory

GPU

Monitor
Raster Graphics System

- Raster: An array of picture elements
- Based on raster-scan TV technology
- The screen (and a picture) consists of discrete pixels, and each pixel has a small display area
Frame Buffer

- Frame buffer: the memory to hold the pixel properties (color, alpha, depth, stencil mask, etc)
- Properties of a frame buffer that affect the graphics performance:
  - Size: screen resolution
  - Depth: color level
    - 1 bit/pixel: black and white
    - 8 bits/pixel: 256 levels of gray or color pallet index
    - 24 bits/pixel: 16 million colors
  - Speed: refresh speed
A dedicated processor for graphics processing.
Graphics Bus Interface

PCI based technology

- Graphics Memory/Frame buffer
- Graphics Processor
- Video Controller

Other Peripherals

PCIe (8 GB/s)

System Bus

CPU
Main Memory
Graphics Accelerators
What do GPUs do?

- Graphics processing units (GPUs) are massively parallel processors
  - Process geometry/pixels and produce images to be displayed on the screen
  - Can also be used to perform general purpose computation (via CUDA/OpenGL)
- Evolved from simple video scan controllers, to special purpose processors that implement a simple pipeline with fixed graphics functionality, to complex many-core architectures that contain several deep parallel pipelines
  - Example: nvidia’s Kepler GK110 contains 15x192 cores and 7.1 billions transistors
  - A graphics card can easily have more than 2GB of video memory
Computer Graphics Hardware
An Overview
CPU/GPU Performance Gap

![Graph showing the performance gap between CPUs and GPUs over time](image-url)
Architecture
- 7.1B Transistors
- 15 SMX units
- > 1 TFLOP FP64
- 1.5 MB L2 Cache
- 384-bit GDDR5
- PCI Express Gen3
nVidia TITAN X (2018)

Architecture

- 12 B Transistors
- 28 SMXs
- 11 TFlops
- 3 MB L2 Cache
- 384-bit GDDR5
- PCI Express Gen3
# nVidia Latest GPUs

<table>
<thead>
<tr>
<th>GPU</th>
<th>Titan X (GP102)</th>
<th>GeForce GTX 1080 (GP104)</th>
<th>Titan X (GM100)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMs</td>
<td>28</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>CUDA Cores</td>
<td>3584</td>
<td>2560</td>
<td>3072</td>
</tr>
<tr>
<td>Base Clock</td>
<td>1417 MHz</td>
<td>1607 MHz</td>
<td>1000 MHz</td>
</tr>
<tr>
<td>GPU Boost Clock</td>
<td>1531 MHz</td>
<td>1733 MHz</td>
<td>1075 MHz</td>
</tr>
<tr>
<td>GFLOPs (Base Clock)</td>
<td>10,157</td>
<td>8228</td>
<td>6144</td>
</tr>
<tr>
<td>Texture Units</td>
<td>224</td>
<td>160</td>
<td>192</td>
</tr>
<tr>
<td>Texel Fill Rate</td>
<td>342.9 GT/s</td>
<td>277.3 GT/s</td>
<td>192 GT/s</td>
</tr>
<tr>
<td>Memory Data Rate</td>
<td>10 Gb/s</td>
<td>10 Gb/s</td>
<td>7 Gb/s</td>
</tr>
<tr>
<td>Memory Bandwidth</td>
<td>480 GB/s</td>
<td>320 GB/s</td>
<td>336.5 GB/s</td>
</tr>
</tbody>
</table>
SMX or SM (Streaming Processor)

- Thousands of registers that can be partitioned among threads of execution
- Several caches:
  - *Shared memory* for fast data interchange between threads
  - *Constant cache* for fast broadcast of reads from constant memory
  - *Texture cache* to aggregate bandwidth from texture memory
  - *L1 cache* to reduce latency to local or global memory

- *Warp schedulers* that can quickly switch contexts between threads and issue instructions to warps that are ready to execute
- Execution cores for integer and floating-point operations:
  - Integer and single-precision floating point operations
  - Double-precision floating point
  - Special Function Units (SFUs) for single-precision floating-point transcendental functions
A PCI Express link between two devices consists of one or more lanes, which are dual simplex channels using two differential signaling pairs.[4][3]

Various slots on a computer motherboard, from top to bottom:
- PCI Express ×4
- PCI Express ×16
- PCI Express ×1
- PCI Express ×16
- Legacy PCI (32-bit, 5 V)

### PCI Express link performance [27][29]

<table>
<thead>
<tr>
<th>PCI Express version</th>
<th>Line code</th>
<th>Transfer rate[a]</th>
<th>Throughput[a]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>x1</td>
</tr>
<tr>
<td>1.0</td>
<td>8b/10b</td>
<td>2.5 GT/s</td>
<td>250 MB/s</td>
</tr>
<tr>
<td>2.0</td>
<td>8b/10b</td>
<td>5 GT/s</td>
<td>500 MB/s</td>
</tr>
<tr>
<td>3.0</td>
<td>128b/130b</td>
<td>8 GT/s</td>
<td>984.6 MB/s</td>
</tr>
<tr>
<td>4.0 (expected in 2017)</td>
<td>128b/130b</td>
<td>16 GT/s</td>
<td>1.969 GB/s</td>
</tr>
<tr>
<td>5.0 (far future)[28]</td>
<td>128b/130b</td>
<td>32 / 25 GT/s</td>
<td>3.9 / 3.08 GB/s</td>
</tr>
</tbody>
</table>
Why are GPU’s so fast?

- Entertainment Industry has driven the economy of these chips?
  - Males age 15-35 buy $10B in video games / year
- Moore’s Law ++
- Simplified design (stream processing)
- Single-chip designs.
Modern GPU has more ALU’s

Figure 1-2. The GPU Devotes More Transistors to Data Processing
A Specialized Processor

- Very Efficient For
  - Fast Parallel Floating Point Processing
  - Single Instruction Multiple Data Operations
  - High Computation per Memory Access

- Not As Efficient For
  - Double Precision
  - Logical Operations on Integer Data
  - Branching-Intensive Operations
  - Random Access, Memory-Intensive Operations
The Rendering Pipeline

- The process to generate two-dimensional images from given virtual cameras and 3D objects
- The pipeline stages implement various core graphics rendering algorithms
- Why should you know the pipeline?
  - Necessary for programming GPUs
  - Understand various graphics algorithms
  - Analyze performance bottleneck

Diagram:
- Host interface → vertex processing → triangle setup → pixel processing → memory interface
The Rendering Pipeline

- The basic construction – three conceptual stages
- Each stage is a pipeline and runs in parallel
- Graphics performance is determined by the slowest stage
- Modern graphics systems:
  - Software
  - Hardware

Diagram:

- Application
  - Geometry
    - Rasteriazer
      - Image
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 with all their associated information (normals, texture coordinates, per vertex color etc).

![Diagram](diagram.png)
The vertex processing stage receives vertices from the host interface in object space and outputs them in screen space.

This may be a simple linear transformation, or a complex operation involving morphing effects.

Normals, texcoords etc are also transformed.

No new vertices are created in this stage, and no vertices are discarded (input/output has 1:1 mapping).
In this stage geometry information becomes raster information (screen space geometry is the input, pixels are the output)

Prior to rasterization, triangles that are backfacing or are located outside the viewing frustum are rejected

Some GPUs also do some hidden surface removal at this stage
Triangle Setup (cont)

- A fragment is generated if and only if its center is inside the triangle.
- Every fragment generated has its attributes computed to be the perspective correct interpolation of the three vertices that make up the triangle.
Each fragment provided by triangle setup is fed into fragment processing as a set of attributes (position, normal, texcoord etc), which are used to compute the final color for this pixel.

The computations taking place here include texture mapping and math operations.

Typically the bottleneck in modern applications
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.
- On modern GPUs, z and color are compressed to reduce framebuffer bandwidth (but not size).
Programmability in the GPU

- Vertex and fragment processing, and now triangle set-up, are programmable
- The programmer can write programs that are executed for every vertex as well as for every fragment
- This allows fully customizable geometry and shading effects that go well beyond the generic look and feel of older 3D applications
The Graphics Pipeline

Application Stage → 3D Triangles → Geometry Stage → 2D Triangles → Rasterization Stage → Pixels

For each triangle vertex:
- Transform 3D position into screen position
- Compute attributes

Rasterization Stage:
- For each triangle:
  - Rasterize triangle
  - Interpolate vertex attributes across triangle
  - Shade pixels
  - Resolve visibility
Diagram of a modern GPU

Input from CPU

Host interface

Vertex processing

Triangle setup

Pixel processing

Memory Interface

64bits to memory

64bits to memory

64bits to memory

64bits to memory
The Quest for Realism

[courtesy: nvidia]