Our Environment

Interaction/User input

Input → Processing Unit → Frame Buffer

Mathematical descriptions of curves, primitives…

Memory
Graphics Hardware: Processor

- Graphics Card
  - A highly parallel computer (stream processing, SIMD)
The Frame Buffer

- Array of pixels (the raster)
- **Resolution** determines number of pixels

**Resolution**  Resolution  Resolution

- **Depth** defines color precision
  - number of bits per pixel
  - 24 bit: 8 bit Red + 8 bit Green + 8 bit Blue = $2^{24}$ colors

<table>
<thead>
<tr>
<th>RGB</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0,0)</td>
<td>(0,1,1)</td>
</tr>
<tr>
<td>(0,1,0)</td>
<td>(0,0,1)</td>
</tr>
</tbody>
</table>
A Note on Color Spaces

The diagram illustrates the color gamuts of different color spaces:

- **ProPhoto RGB**
- **sRGB**
- **Epson 2200**

The diagram shows the x-y chromaticity coordinates with wavelengths (in nanometers) along the axes.
Chapter 2

1. Introduction
2. The Computer Graphics Pipeline
3. Object Representation
4. Object Transformation
5. 3D – Projections, Camera, and Lighting
6. Scene Representation and Interaction
7. Advanced Texturing and Shading
The Pipeline

- From vertices to pixels
- Fixed and programmable steps
- Implemented in hardware

- Concrete form is implementation specific
The Input

- Primitives, curves, vertices, … (geometry)
- Color, opacity, … (optical properties)

- Also: transformations, camera
The Input

- Objects
  - Information given per vertex
  - Connectivity defines faces, cells, inside/outside
  - Given as surfaces/shells
The Pipeline

Vertex Processor → Clipper and Assembler → Rasterizer → Fragment Processor
The Vertex Processor

- Operates on vertices
  - “Transform” the object and place it in relation to camera
  - Perform coordinate transformations
  - Process per-vertex attributes (colors, etc.)

- OpenGL:
  - Vertex Shader: Maps vertices to vertices
  - (Tessellation – optional)
  - Geometry Shader: Can create new primitives/vertices

Note: Tessellation and Geometry shaders can be considered part of the Vertex Processing or Primitive Assembly stage
The Vertex Processor

- Coordinate system transformation
The Vertex Processor

- Coordinate system transformation
The Vertex Processor

- Creation of new vertices/primitives
The Pipeline

1. Vertex Processor
2. Clipper and Assembler
3. Rasterizer
4. Fragment Processor
Viewing Volume

• Important to define mapping to 2D and clipping
Clipping and Assembling

- Assembles primitives from vertices
- Determines which objects are visible from camera
Clipping and Assembling

- Operations no longer per-vertex

- Clipping: Polygon-Polygon, Polygon-Line intersections
  - Rejects (culls) or accepts primitives
  - Clips parts outside of viewing volume
  - Reduces workload of rasterizer
  - Outputs normalized device coordinates
The Pipeline
Rasterization

- Working towards a representation that fits frame buffer
  - From normalized device coordinates to window coordinates
  - Identify pixels ‘inside’ of primitive
  - Convert primitives into sets of fragments
  - Interpolate attributes given by vertices
Rasterization

- Fragments
  - Carry more than ‘just’ color
  - Depth, alpha, texture coordinates,…
  - are potential pixels
The Pipeline

Vertex Processor → Clipper and Assembler → Rasterizer → Fragment Processor
The Fragment Processor

- Operates on fragments
  - Apply per-fragment operations (color, texture, …)
  - Hidden surface removal
  - Alpha compositing
  - Update frame buffer pixels
Using the Pipeline

• Make use of hardware features
  • Functionality implemented in graphics card driver
  • GPU (graphics processing unit) accelerates typical operations

• Abstraction from hardware/driver:
  • API (application programming interface)
  • Defines variables, algorithms, and routines

  • Large selection of APIs available for free download and use
API

Application → API → Driver

Input Device

OpenGL

Output Device
The Tools

- Two mainstream APIs for graphics:
  - OpenGL (WebGL, OpenGL ES, ...)
  - DirectX (Direct3D, DirectDraw, ...)

- OpenGL:
  - Open Standard
  - Multi-platform
  - http://www.opengl.org
  - C bindings (also Java, Python, ...)

- Download, compile, and run demos from
Demo

Short OpenGL pipeline demo
Pipeline Summary

Vertex Processor → Clipper and Assembler → Rasterizer → Fragment Processor

3D Triangles → Clipped Triangles → 2D Triangles → Fragments
Chapter 2 - Summary

• Programmable stages in computer graphics pipeline

• Operations are inherently parallel
  • **SIMD**: Single Instruction, Multiple Data

• Stages are implemented on graphics hardware
  • From vertices to pixels