The Rendering Pipeline –
A First Look

Your Tasks Until Monday

**Assignment 0**
- Refresher of linear algebra
- Set up programming environment on lab computers

**Reading (in Shirley: Introduction to CG)**
- Math refresher: Chapters 2, 4
  - Optional (for now): 2.5-2.9
- Background on graphics: Chapter 1
The Rendering Pipeline

**What is it? All of this:**
- Abstract model for sequence of operations to transform a geometric model into a digital image
- An abstraction of the way graphics hardware works
- The underlying model for application programming interfaces (APIs) that allow the programming of graphics hardware
  - *OpenGL*
  - *Direct 3D*

*Actual implementations of the rendering pipeline will vary in the details*
Rendering Pipeline

**Advantages of a pipeline structure**
- Logical separation of the different components, modularity
- Easy to parallelize:
  - *Earlier stages can already work on new data while later stages still work with previous data*
  - *Similar to pipelining in modern CPUs*
  - *But much more aggressive parallelization possible (special purpose hardware!)*
  - *Important for hardware implementations!*
- Only local knowledge of the scene is necessary

Rendering Pipeline

**Disadvantages:**
- Limited flexibility
- Some algorithms would require different ordering of pipeline stages
  - *Hard to achieve while still preserving compatibility*
- Only local knowledge of the scene is available
  - *Shadows*
  - *Global illumination*
The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer

Rasterization → Fragment Processing

Geometry Processing

Geometry Database

Needs to represent models for
- Geometric primitives
- Relations between different primitives (transformations)
- Object materials
- Light sources
- Camera
Geometric Primitives

**Different philosophies:**
- Collections of complex shapes
  - Spheres, cones, cylinders, tori, …
- One simple type of geometric primitive
  - Triangles or triangle meshes
- Small set of complex primitives with adjustable parameters
  - E.g. “all polynomials of degree 2”
  - Splines, NURBS (details in CPSC 424)
  - Fractals

Geometric Primitives

**Mathematical representations:**
- Explicit functions
- Parametric functions
- Implicit functions
Explicit Functions

**Curves:**
- $y$ is a function of $x$: $y := \sin(x)$
- Only works in 2D

**Surfaces:**
- $z$ is a function of $x$ and $y$: $z := \sin(x) + \cos(y)$
- Cannot define arbitrary shapes in 3D

Parametric Functions

**Curves:**
- 2D: $x$ and $y$ are functions of a parameter value $t$
- 3D: $x$, $y$, and $z$ are functions of a parameter value $t$

\[
C(t) := \begin{pmatrix} \cos(t) \\ \sin(t) \\ t \end{pmatrix}
\]
Parametric Functions

**Surfaces:**
- Surface $S$ is defined as a function of parameter values $s$, $t$
- Names of parameters can be different to match intuition:

$$S(\phi, \theta) := \begin{pmatrix} \cos(\phi) \cos(\theta) \\ \sin(\phi) \cos(\theta) \\ \sin(\theta) \end{pmatrix}$$

Geometry Database

**Implicit Surfaces:**
- Surface is defined implicitly via the roots of a function
- E.g:

$$S(x, y, z) : x^2 + y^2 + z^2 - 1 = 0$$
Geometry Database

**Triangles and Triangle Meshes:**

---

The Rendering Pipeline

Geometry Processing

- Geometry Database
- Model/View Transform.
- Lighting
- Perspective Transform.
- Clipping

Rasterization

- Scan Conversion
- Texturing

Fragment Processing

- Depth Test
- Blending

Frame-buffer
Modeling and Viewing Transformation

**Modeling transformation:**
- Map points from *object coordinate system* to *world coordinate system*
- Same as placing objects

**Viewing transformation:**
- Map points from *world coordinate system* to *camera (or eye) coordinate system*
- Same as placing camera
**Viewing Transformation: Camera Placement**

**Modeling and Viewing Transformation**

**Types of transformations:**
- Rotations, scaling, shearing
- Translations

Other transformations (not handled by rendering pipeline):
  - Freeform deformation
**Modeling and Viewing Transformation**

**Linear transformations**
- Rotations, scaling, shearing
- Can be expressed as a 3x3 matrix
- E.g. rotation:

\[
\begin{pmatrix}
  x' \\
  y' \\
  z'
\end{pmatrix} =
\begin{pmatrix}
  \cos(\phi) & -\sin(\phi) & 0 \\
  \sin(\phi) & \cos(\phi) & 0 \\
  0 & 0 & 1
\end{pmatrix}
\cdot
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]

**Affine transformations**
- Linear transformations + translations
- Can be expressed as a 3x3 matrix + 3 vector
- E.g. rotation + translation:

\[
\begin{pmatrix}
  x' \\
  y' \\
  z'
\end{pmatrix} =
\begin{pmatrix}
  \cos(\phi) & -\sin(\phi) & 0 \\
  \sin(\phi) & \cos(\phi) & 0 \\
  0 & 0 & 1
\end{pmatrix}
\cdot
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix} +
\begin{pmatrix}
  t_x \\
  t_y \\
  t_z
\end{pmatrix}
\]

Another representation: 4x4 homogeneous matrix
The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Geometry Processing

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer

Rasterization → Fragment Processing

Lighting

Wolfgang Heinrich
Complex Lighting and Shading

The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Geometry Processing

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer

Rasterization → Fragment Processing
Perspective Transformation

**Purpose:**
- Project 3D geometry onto a 2D image plane
- Simulates a camera

**Camera model:**
- Pinhole camera
- Other, more complex camera models also exist in computer graphics, but are less common
  - Thin lens cameras
  - Full simulation of lens geometry

Perspective Projection
**Perspective Transformation**

*Pinhole Camera:*
- Light shining through a tiny hole into a dark room yields upside-down image on wall

![Diagram of light shining through a small hole onto a wall with upside-down image.]
Pinhole Camera - Camera Obscura

Perspective Transformation

*In computer graphics:*

- Image plane is conceptually *in front* of the center of projection

  Perspective transformations belong to a class of operations that are called *projective transformations*

  Linear and affine transformations also belong to this class

  *All* projective transformations can be expressed as $4 \times 4$ matrix operations

Wolfgang Heidrich
The Rendering Pipeline

Geometry Database → Model/View Transform. → Lighting → Perspective Transform. → Clipping

Scan Conversion → Texturing → Depth Test → Blending → Frame-buffer

Rasterization → Fragment Processing

Scan Conversion

Wolfgang Heidrich
Scan Conversion

*Problem:*
- Line is infinitely thin, but image has finite resolution
- Results in steps rather than a smooth line
  - Jaggies
  - Aliasing
- One of the fundamental problems in computer graphics
Scan Conversion

**Color interpolation**

- Linearly interpolate per-pixel color from vertex color values
- Treat every channel of RGB color separately
Scan Conversion

**Color interpolation**

Example:

```
red  green  blue
```

![Diagram showing color interpolation example with red, green, and blue colors](image)

The Rendering Pipeline

![Diagram showing the rendering pipeline with steps from geometry database to framebuffer](image)
Texturing

Texture Mapping
Displacement Mapping

Reflection Mapping
Texturing

**Issues:**
- How to map pixel from texture (texels) to screen pixels
  - Texture can appear widely distorted in rendering
  - Magnification / minification of textures
- Filtering of textures
- Preventing aliasing (anti-aliasing)

---

The Rendering Pipeline

Geometry Database \(\rightarrow\) Model/View Transform. \(\rightarrow\) Lighting \(\rightarrow\) Perspective Transform. \(\rightarrow\) Clipping

Geometry Processing

Scan Conversion \(\rightarrow\) Texturing \(\rightarrow\) Depth Test \(\rightarrow\) Blending \(\rightarrow\) Frame-buffer

Rasterization \(\rightarrow\) Fragment Processing

Wolfgang Heidrich
Hidden Surface Removal

Depth Test / Hidden Surface Removal

**Remove invisible geometry**
- Parts that are hidden behind other geometry

**Possible Implementations:**
- Per-fragment decision
  - *Depth buffer*
- Object space decision
  - *Clipping polygons against each other*
  - *Sorting polygons by distance from camera*
Depth Test / Hidden Surface Removal

The Rendering Pipeline
Display Technology

Cathod Ray Tubes (CRTs)

Display Technology

Raster Scan Electron Beam
Display Technology

**Interlaced Scanning**

![Interlaced Scanning Diagram]

**Display Technology**

**Color CRTs**

![Color CRTs Diagram]
Display Technology

**Trinitron CRTs**

- electron gun assembly
- Trinitron slit type shadow mask
- shadow mask ("aperture grill", "harp")
- phosphor layer

Wolfgang Heidrich

Display Technology

**Liquid Crystal Displays (LCD)**

- polarization filter
- glass
- liquid crystal devices
- glass
- polarization filter

ON

OFF

Wolfgang Heidrich
High Dynamic Range Displays

Luminance (log cd/m²)

-6  -4  -2  0  2  4  6  8

starlight  moonlight  indoor lighting  sunlight

Range of Illumination

scotopic  mesopic  photopic

Visual Function

No colour vision
Poor acuity
Good colour vision
Good acuity

Human Simultaneous Visual Range

Conventional Display Luminance

HDR Display Principle

- Modulated LED array
- Conventional LCD
- Image compensation
Prototype Setup: Projector/LCD Panel

**Hardware setup:**
- Remove backlight from LCD panel
- Shine image from video projector onto back of panel
  - *(Fresnel lens for focusing)*
- Multiplies dynamic range of LCD and projector

**Measured:**
- Contrast: 50,000:1
- Intensity: 2,700 cd/m²

---

**Brightside Technologies / Dolby Commercial Display**

18” prototype: Zeetzen 5

37” commercial prototype DR-37P
LG Philips - “Local Area Luminance Control”

Coming Up...

Next week:
- Geometric Transformations (Affine, Perspective)