The Rendering Pipeline – A Second Look

Part 1: 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

Explicit Functions

Mathematical representations:
- Explicit functions
- Parametric functions
- Implicit 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

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

**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.: \[ S(x, y, z) : x^2 + y^2 + z^2 - 1 = 0 \]

**Triangles and Triangle Meshes:**

**The Rendering Pipeline**

**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
Modeling Transformation: Object Placement

Viewing Transformation: Camera Placement

Modeling and Viewing Transformation

Types of transformations:
- Rotations, scaling, shearing
  ![Illustration of transformations](image1.png)
- 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:

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

Modeling and Viewing Transformation

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

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

- Another representation: 4x4 homogeneous matrix

The Rendering Pipeline
**Lighting**

**Complex Lighting and Shading**

**The Rendering Pipeline**

**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
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

Pinhole Camera - Camera Obscura

The Rendering Pipeline

Part 2: Rasterization & Fragment Processing

The Rendering Pipeline – A Second Look
**Problem:**
- Line is infinitely thin, but image has finite resolution
- Results in steps rather than a smooth line
  - Jagged
  - Aliasing
- One of the fundamental problems in computer graphics

**Color interpolation**
- Linearly interpolate per-pixel color from vertex color values
- Treat every channel of RGB color separately

**Example:**
- Red
- Green
- Blue
The Rendering Pipeline

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

Scan Conversion
- Texturing
- Depth Test
- Blending
- Frame Buffer

Rasterization

With Hidden Line Removal

Without Hidden Line Removal

Hidden Line Removal

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

Display Technology

Color CRTs

Display Technology

Trinitron CRTs
Display Technology

**Liquid Crystal Displays (LCD)**

---

**Coming Up...**

**Thursday, Sep 13:**
- Geometric Transformations (Affine)

**Tuesday, Sep 18:**
- Geometric Transformations (Perspective)