<table>
<thead>
<tr>
<th>Course Syllabus</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Image processing</td>
</tr>
<tr>
<td>II. Rendering</td>
</tr>
<tr>
<td>III. Modeling</td>
</tr>
<tr>
<td>IV. Animation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3D Rendering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greg Humphreys</td>
</tr>
<tr>
<td>University of Virginia</td>
</tr>
<tr>
<td>CS 445, Fall 2003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Where Are We Now?</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Image processing</td>
</tr>
<tr>
<td>II. Rendering</td>
</tr>
<tr>
<td>III. Modeling</td>
</tr>
<tr>
<td>IV. Animation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>The Everyday World</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>
The Nature of Realism

Toyota Supra by Kevin Hulsey

Café Express by Richard Estes, 1975

Rendering

- Generate an image from geometric primitives

Geometric Primitives  Rendering  Raster Image

3D Rendering Example

What issues must be addressed by a 3D rendering system?
# Overview

- 3D scene representation
- 3D viewer representation
- Visible surface determination
- Lighting simulation

## How is the 3D scene described in a computer?

### 3D Scene Representation

- Scene is usually approximated by 3D primitives
  - Point
  - Line segment
  - Polygon
  - Polyhedron
  - Curved surface
  - Solid object
  - etc.

### 3D Point

- Specifies a location
3D Point

- Specifies a location
  - Represented by three coordinates
  - Infinitely small

```c
typedef struct {
  Coordinate x;
  Coordinate y;
  Coordinate z;
} Point;
```

3D Vector

- Specifies a direction and a magnitude

```c
typedef struct {
  Coordinate dx;
  Coordinate dy;
  Coordinate dz;
} Vector;
```

### Dot product of two 3D vectors

- \( V_1 \cdot V_2 = dx_1 dx_2 + dy_1 dy_2 + dz_1 dz_2 \)
- \( V_1 \cdot V_2 = ||V_1|| ||V_2|| \cos(\theta) \)
3D Line Segment

- Linear path between two points

\[ P = P_1 + t (P_2 - P_1), \quad (0 \leq t \leq 1) \]

typedef struct {
    Point P1;
    Point P2;
} Segment;

3D Line Segment

- Use a linear combination of two points

\[ P = P_1 + t (P_2 - P_1), \quad (0 \leq t \leq 1) \]

3D Ray

- Line segment with one endpoint at infinity

\[ P = P_1 + t V, \quad (0 \leq t < \infty) \]

typedef struct {
    Point P1;
    Vector V;
} Ray;

3D Line

- Line segment with both endpoints at infinity

\[ P = P_1 + t V, \quad (-\infty < t < \infty) \]

typedef struct {
    Point P1;
    Vector V;
} Line;
3D Plane

- A linear combination of three points

$$P_1 \cdot P_2 \cdot P_3$$

**Implicit representation:**
- $$P \cdot N + d = 0$$, or
- $$ax + by + cz + d = 0$$

$$N$$ is the plane “normal”
- Unit-length vector
- Perpendicular to plane

```c
typedef struct {
    Vector N;
    Distance d;
} Plane;
```

3D Plane

- A linear combination of three points

$$P_1 \cdot P_2 \cdot P_3$$

**Implicit representation:**
- $$P \cdot N + d = 0$$, or
- $$ax + by + cz + d = 0$$

$$N = (a,b,c)$$

```c
typedef struct {
    Vector N;
    Distance d;
} Plane;
```

3D Polygon

- Area “inside” a sequence of coplanar points
  - Triangle
  - Quadrilateral
  - Convex
  - Star-shaped
  - Concave
  - Self-intersecting

Points are in counter-clockwise order
- Holes (use > 1 polygon struct)

3D Sphere

- All points at distance “r” from point “$$(c_x, c_y, c_z)$$”

**Implicit representation:**
- $$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 = r^2$$

**Parametric representation:**
- $$x = r \cos(\phi) \cos(\theta) + c_x$$
- $$y = r \cos(\phi) \sin(\theta) + c_y$$
- $$z = r \sin(\phi) + c_z$$

```c
typedef struct {
    Point center;
    Distance radius;
} Sphere;
```
3D Geometric Primitives

- More detail on 3D modeling later in course
  - Point
  - Line segment
  - Polygon
  - Polyhedron
  - Curved surface
  - Solid object
  - etc.

Overview

- 3D scene representation
  » 3D viewer representation
- Visible surface determination
- Lighting simulation

Camera Models

- The most common model is pin-hole camera
  - All captured light rays arrive along paths toward focal point without lens distortion (everything is in focus)
  - Sensor response proportional to radiance

Camera Parameters

- What are the parameters of a camera?
### Camera Parameters

- **Position**
  - Eye position (px, py, pz)

- **Orientation**
  - View direction (dx, dy, dz)
  - Up direction (ux, uy, uz)

- **Aperture**
  - Field of view (xfov, yfov)

- **Film plane**
  - *Look at* point
  - View plane normal

### Moving the camera

![View Frustum](image)

### Overview

- 3D scene representation
- 3D viewer representation
- **Visible surface determination**

- Lighting simulation

### Visible Surface Determination

- The color of each pixel on the view plane depends on the radiance emanating from visible surfaces

- Simplest method is ray casting

How can the front-most surface be found with an algorithm?
Ray Casting

- For each sample …
  - Construct ray from eye position through view plane
  - Find first surface intersected by ray through pixel
  - Compute color of sample based on surface radiance

Visible Surface Determination

- For each sample …
  - Construct ray from eye position through view plane
  - Find first surface intersected by ray through pixel
  - Compute color of sample based on surface radiance

Rendering Algorithms

Rendering is a problem in sampling and reconstruction!

More efficient algorithms utilize spatial coherence!
Overview
• 3D scene representation
• 3D viewer representation
• Visible surface determination
  » Lighting simulation

Lighting Simulation
• Lighting parameters
  ▪ Light source emission
  ▪ Surface reflectance
  ▪ Atmospheric attenuation
  ▪ Camera response

How do we compute the radiance for each sample ray?

Lighting Simulation
• Direct illumination
  ▪ Ray casting
  ▪ Polygon shading
• Global illumination
  ▪ Ray tracing
  ▪ Monte Carlo methods
  ▪ Radiosity methods

More on these methods later!

Summary
• Major issues in 3D rendering
  ▪ 3D scene representation
  ▪ 3D viewer representation
  ▪ Visible surface determination
  ▪ Lighting simulation

• Concluding note
  ▪ Accurate physical simulation is complex and intractable
  ▪ Rendering algorithms apply many approximations to simplify representations and computations
Next Lecture

- Ray intersections
- Light and reflectance models
- Indirect illumination

For assignment #3, you will write a ray tracer!