WHAT IS RENDERING?

Generating image from a 3D scene

Let's think HOW.

SCENE

• Coordinate frame
• 3D models
• Their materials
• Lights
• Cameras

FRAME BUFFER

• Portion of RAM on videocard (GPU)
• What we see on the screen
• Rendering destination

SCREEN

• Displays what's in frame buffer
• Terminology:
  Pixel: basic element on device
  Resolution: number of rows & columns in device
  Measured in:
  • Absolute values (1K x 1K)
  • Density values (300 dots per inch)

SINGLE OBJECT

• How to describe a single piece of geometry?
• So far geometry has been constructed for you.

SHAPES: TRIANGLE MESHES

• Triangle: 3 vertices
• Mesh: [vertices, triangles]
• Example

SCENE

• How to describe a scene?
• Local Transformations

FRAME BUFFER

Scene
Coordinate frame
3D models
Their materials
Lights
Cameras

FINAL IMAGE

Scene
Coordinate frame
3D models
Their materials
Lights
Cameras

SKETCH OF A RENDERING PIPELINE

Scene
Coordinate frame
3D models
Their materials
Lights
Cameras

Frame buffer
Portion of RAM on videocard (GPU)
What we see on the screen
Rendering destination

FINAL IMAGE

Scene
Coordinate frame
3D models
Their materials
Lights
Cameras

SKETCH OF A RENDERING PIPELINE

Scene
Coordinate frame
3D models
Their materials
Lights
Cameras

Image
Camera View
• Depth of shapes
• Normals
• Which pixel is visible

Camera

Pixel

Resolution

Absolute values
Density values

Scene
3D objects
Materials
Lights
Cameras
Framebuffer
final image

Pixel: basic element on device
Resolution: number of rows & columns in device
Measured in:
• Absolute values (1K x 1K)
• Density values (300 dots per inch)

Scene
Coordinate Frame
3D objects
Materials
Lights
Cameras
Framebuffer
final image

What we see on the screen
Rendering destination

Scene
• How to describe a scene?
• Local Transformations

Scene
• A coordinate frame
• 3D objects
• Their materials
• Lights
• Cameras

Final image
MINIMAL VERTEX SHADE

- Transforms the vertices
- `gl_Position = modelViewMatrix * position;`

MINIMAL FRAGMENT SHADE

- Performs per-pixel shading
- `gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);`

FRAGMENT SHADE

- Final output
- `Has gl_FragCoord – 2D window coords
  - May set color!

VIEW COORDINATE SYSTEM

- Defined by Three.js
- `x y z`
- `1.0`

RASERIZATION - INTERPOLATION

- Interpolates vertex attributes

```
void main()
{
  // Transforming The Vertex
  gl_Position = modelViewMatrix * position;

  // Setting Each Pixel To Red
  gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
}
```

OPENGL RENDERING PIPELINE

- Pipeline from vertex to frame buffer
- `Fragment color = ...`
MODELING & VIEWING TRANSFORMATIONS

- Linear transformations:
  - Rotations, scaling, shearing
  - Can be expressed as 3x3 matrix + 3 vector
  - E.g. scaling (non-uniform)

- Affine transformations:
  - Linear transformations + translations
  - Can be expressed as 4x4 matrix + 4 vector
  - E.g. shear translation

- Another representation: 4d homogeneous matrix
**PERSPECTIVE TRANSFORMATION**
- In computer graphics:
  - Image plane conceptually in front of center of projection
- Perspective transformation is one of projective transformations
- Linear & affine transformations also belong to this class
- All projective transformations can be expressed as 4x4 matrix operations

**SCAN CONVERSION/RASTERIZATION**
- Convert continuous 2D geometry to discrete
- Raster display – discrete grid of elements
- Terminology:
  - Screen Space: Discrete 2D Cartesian coordinate system of the screen/plane

**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

**COLOR INTERPOLATION**
- Linearly interpolate per-pixel color from source color values
  - Treat every channel of RGB color separately

**CLIPPING**
- Removing invisible geometry
- Geometry outside viewing frustum
- Camera model:
  - Pinhole camera (single view point)
  - More complex camera models exist, but are less common in CG
- Optimization
### TEXTURING

- **Issues:**
  - Computing 3D/2D map (low distortion)
  - 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)

### LIGHTING

- **Complex Lighting and Shading**

### BLENDING

- **Frame Buffer:** video memory on graphics board that holds resulting image and used to display it

### DEPTH TEST / HIDDEN SURFACE REMOVAL

- **Issues:**
  - Remove invisible geometry
  - Parts that are hidden behind other geometry
  - Possible implementations:
    - Pixel level decision
      - Object space decision
      - E.g. intersection order for ray tracing
      - E.g. intersection order for ray tracing
  - Possible implementations:
    - Color space decision
      - Object space decision
      - E.g. intersection order for ray tracing

### REFLECTION/SHADOWS