PROGRAMMING ASSIGNMENT 1

• Is out! Posted on Piazza
• Due 23:59:59, Sep 25th
• Grace days: 3 per term – use wisely
  • Weekend doesn’t count

• It will take time to set up the environment
• You will not be able to complete all until Lecture 4

• ENJOY
ENVIRONMENT

• Write code in any text editor
  • Notepad++ (win)
  • Sublime text (any platform)
  • vim (linux)

• Handin

• After it’s handed in, TAs will set up face-to-face time

• Labs starting Tuesday
PIAZZA

• Up and running
• Please sign up

• Link to Assgn 1 will be posted there
  • And website
WHAT IS RENDERING?

Generating image from a 3D scene
WHAT IS RENDERING?

Generating image from a 3D scene

Let’s think HOW.
SCENE

- A coordinate frame
- 3D objects
- 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?
SHAPES: TRIANGLE MESHES

• Triangle = 3 vertices

• Mesh = \{vertices, triangles\}

• Example
SCENE

• How to describe a scene?
SCENE

- How to describe a scene?
- Local Transformations
SKETCH OF A RENDERING PIPELINE

• Scene
  • Coordinate frame
  • 3D models
    • Coordinates
    • Local transforms
    • properties (color, material)
• Lights
• Camera
SKETCH OF A RENDERING PIPELINE

• **Scene**
  - Coordinate frame
  - 3D models
    - Coordinates
    - properties (color, material)
  - Lights
  - Camera

• **Camera View**
  - 2D positions of shapes
  - Depth of shapes
  - Normals

• **Image**
  - Shape pixels
  - Their color
  - Which pixel is visible
OPENGL/WEBGL

• Open Graphics Library
• One of the most popular libraries for 2D/3D rendering
• A software interface to communicate with graphics hardware
• Cross-language API
OPENGL RENDERING PIPELINE

Vertices and attributes → Vertex Shader → Vertex Post-Processing → Rasterization

Fragment Shader → Per-Sample Operations → Framebuffer
OPENGL RENDERING PIPELINE

Scene
Vertices and attributes

Camera Coords
Vertex Shader
Fragment Shader

Device Coords
Vertex Post-Processing
Per-Sample Operations

Rasterization
Framebuffer

Image
Vertex Shader

Vertices and attributes

- Vertex Shader
- Vertex Post-Processing
- Rasterization
- Framebuffer

- Fragment Shader
- Per-Sample Operations
**VERTEX SHADER**

- Vertices are stored in vertex buffer
- Each one is processed by vertex shader
- Outputs 2D position
- May compute per-vertex variables (normal, etc.)
RASTERIZATION

Vertices and attributes

Vertex Shader ➔ Vertex Post-Processing ➔ Rasterization

Fragment Shader ➔ Per-Sample Operations ➔ Framebuffer
RASTERIZATION

Places three 2D vertices on a virtual screen
Fills up the space between them
Interpolates per-vertex variables to get per-fragment vars
Vertices and attributes → Vertex Shader → Vertex Post-Processing → Rasterization → Framebuffer

- - - → Fragment Shader → Per-Sample Operations → Framebuffer

FRAGMENT SHADER
FRAGMENT SHADER

- Each fragment is passed through Fragment Shader
- Here it computes fragment color
FRAGMENT SHADER

• Can simulate different materials and lights
PIPELINE: MORE DETAILS

Vertices and attributes

Vertex Shader → Vertex Post-Processing → Rasterization

Fragment Shader → Per-Sample Operations → Framebuffer
PIPELINE: MORE DETAILS

- **Vertex Shader**
  - Modelview transform
  - Per-vertex attributes

- **Vertex Post-Processing**
  - Viewport transform
  - Clipping

- **Rasterization**
  - Scan conversion
  - Interpolation

- **Fragment Shader**
  - Texturing/...
  - Lighting/shading

- **Per-Sample Operations**
  - Depth test
  - Blending

- **Framebuffer**