COMPUTER GRAPHICS

California State University, Fresno

What is Shading?

- Calculating how a 3D object should look, taking lighting, vertex and color into account

- Depends on:

  - mesh data
    - vertices, colors, textures, etc.
  - lighting properties
    - type of light, position of light, etc.
  - shading algorithm used

- How is it done?

  Basically by running a script on your graphics card. This will be a program used to determine the final surface properties of an object or image

Simpler Shader Definition

- http://madebyevan.com/webgl-water/

- A program that can affect:
  - the position of a vertex
  - the color of a pixel
  - or both

- This can include arbitrarily complex descriptions of:
  - light absorption
  - light diffusion
  - texture mapping
  - reflection
  - refraction
  - shadowing
  - surface displacement
  - post-processing effects

Shader Languages

- HLSL, GLSL, Cg (C for Graphics), CUDA, OpenCL
- Compile to Assembly
- Shader compilers are on the GPU
- Compiling is done at runtime

Shading Languages

- Shader programs may be written for different platforms
  - can operate on GPU

- Different platforms use different shading languages, e.g.
  - High-Level Shading Language (HLSL)
    - uses C-like code
  - OpenGL Shading Language (GLSL)
    - uses C-like code
  - Nvidia Cg
    - uses assembly-like code
  - Pixar’s RenderMan
    - the pioneer among shaders

Shader Types

- **Vertex Shader**
- **Pixel Shader** (Fragment Shader in GLSL)
- Geometry Shader
- Compute Shader

Types of Shaders

- Vertex Shaders
  - allows programmer to control transformation & lighting operations (T&L)

- Pixel Shading
  - The output of a vertex shader provides input to a pixel shader
  - allows programmer to ultimately decide final pixel colors before rasterization

Vertex Shaders

- http://jeremybouny.fr/ocean/demo/
- https://www.clicktorelease.com/code/bumpy-metaballs/

Shader Code

- Any data you want those functions to have access to must be provided to the GPU

- There are 4 ways a shader can receive data.
  - Attributes and Buffers
  - Uniforms
  - Textures
  - Varyings

Data Types

- Attributes and Buffers
  - Buffers are arrays of binary data you upload to the GPU
  - Usually buffers contain
    - Positions
    - Normals
    - Texture coordinates
    - Vertex
    - Colors etc.

Data Types … cntd.

- Uniforms
  - Uniforms are effectively global variables you set before you execute your shader program.

- Textures
  - Textures are arrays of data you can randomly access in your shader program.
  - The most common thing to put in a texture is image data but textures are just data and can just as easily contain something other than RGB.

- Varyings
  - Varyings are a way for a vertex shader to pass data to a fragment shader.
  - Depending on what is being rendered, points, lines, or triangles, the values set on a varying by a vertex shader will be interpolated while executing the fragment shader.

basic non-vector types:

bool: conditional type, values may be either true or false

int: a signed, two's complement, 32-bit integer

uint: an unsigned 32-bit integer

float: an IEEE-754 single-precision floating point number

Double: an IEEE-754 double-precision floating-point number