CPSC 314
Computer Graphics

Dinesh K. Pai

Nuts and bolts of graphics programming

Announcements

- Today:
  - Assignment 1
  - Programming with WebGL and GLSL

2

- Assignment 1 demo

3

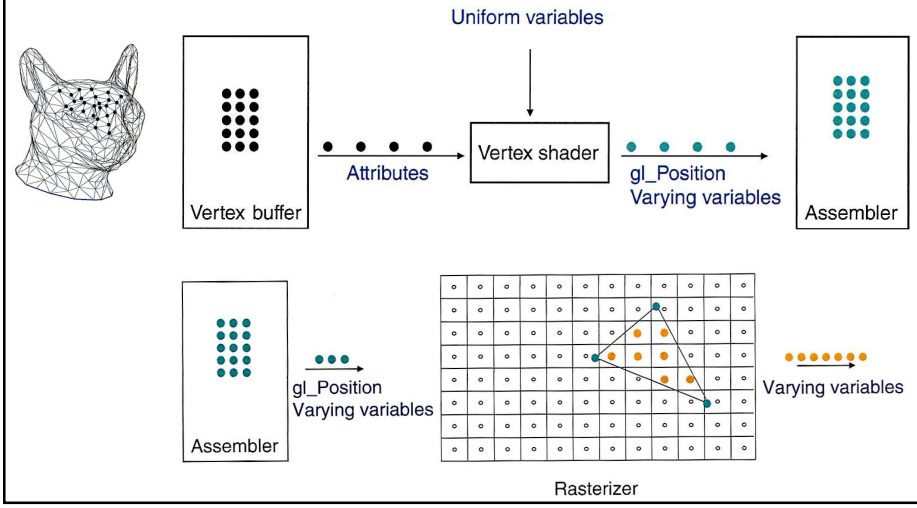
Assignment 1

- First thing: download template from repository and get it running locally on your computer.
- There are lots of details in the template that you can ignore till later in the course. Skim the general structure. Look for comments “HINT” or “YOUR WORK”
- Make small modifications (a few of lines of code) to the shaders, and understand how to pass information from a JavaScript program to the shaders

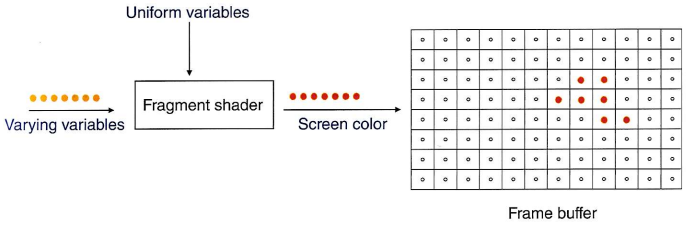
4

Recap Pipeline: Vertex Shader

- Read last lecture and Textbook Chapter 1!



Recap Pipeline: Fragment Shader



6

A closer look at GLSL shaders

Handy reference:

https://www.khronos.org/files/webgl/webgl-reference-card-1_0.pdf

Pages 3 and 4 cover GLSL

7

GLSL

- OpenGL shading language
- C-like, w. data types and functions useful for graphics
  - vec3, vec4, dvec4, mat4, sampler2D …
    (OpenGL data are floats unless qualified)
  - <matrix-vector multiplication>, reflect, refract
- Used for both vertex shaders and fragment shaders, with small differences
- WebGL 1.0 uses GLSL 1, compatible with Open GL ES 2.0. We use this in our course.
- WebGL 2.0 has recently been released with many advanced features (compatible with Open GL ES 3.0) but not widely available yet.

8

Summary of Key GLSL Concepts (1)

- ‘uniform’ type qualifier: Same for all vertices
- ‘attribute’ type qualifier: per vertex data
- ‘varying’ type qualifiers: configure data flow in the pipeline.
  - Output of vertex shader, input to fragment shader (after interpolation)
- gl_Position is built-in output variable that must be set before rasterization

9

Summary of Key GLSL Concepts (2)

- Support for geometry, vector and matrix arithmetic
  - length, distance, dot, cross, normalize, reflect
- Compiled by WebGL, at runtime

10

Three.js support

- THREE.ShaderMaterial() lets you set shaders, uniforms
- Built-in uniforms and attributes. See https://threejs.org/docs/#api/renderers/webgl/WebGLProgram
- Some vertex attributes
  - position, normal, and uv
- Some uniforms
  - modelView matrix and cameraPosition

11

```
/**
 * UBC CPSC 314, Vjan2015
 * Outline of a Three.js program for this course
 */
//  SCENE
var scene = new THREE.Scene();
//  RENDERER
var renderer = new THREE.WebGLRenderer();
//  CAMERA
var camera = new THREE.PerspectiveCamera(30, 1, 0.1, 1000);
//  SHADERS
var gemMaterial = new THREE.ShaderMaterial({
   uniforms: { gemPosition: gemPosition},
   vertexShader: <VertexShaderSource>,
   fragmentShader: <FragmentShaderSource>
});
//  OBJECT GEOMETRY
var gemGeometry = new THREE.SphereGeometry(1, 32, 32);
//  OBJECT MESH
var gem = new THREE.Mesh(gemGeometry, gemMaterial);

scene.add(gem);

// SETUP UPDATE CALL-BACK
function update() {
  requestAnimationFrame(update);
  renderer.render(scene, camera);
}
update();
```

12

Minimalist shaders

```
vertex shader
-------------

uniform vec3 gemPosition;
varying vec3 color;

void main() {
  color = normal;
  gl_Position = projectionMatrix * modelViewMatrix * vec4(position, 1.0);
}

fragment shader
---------------

varying vec3 color;
void main() {
  gl_FragColor = vec4(normalize(color), 1.0);
}
```

13

ShaderMaterial Example

```
var material = new THREE.ShaderMaterial( {

        uniforms: {
                time: { type: "f", value: 1.0 },
                resolution: { type: "v2", value: new THREE.Vector2() }
        },
        attributes: {
                vertexOpacity: { type: 'f', value: [] }
        },
        vertexShader: document.getElementById( 'vertexShader' ).textContent,
        fragmentShader: document.getElementById( 'fragmentShader' ).textContent

} );
```

https://threejs.org/docs/#api/materials/ShaderMaterial

14

Animation (infinite) Loop

```
// SETUP UPDATE CALL-BACK
function update() {
  requestAnimationFrame(update); // next frame
  renderer.render(scene, camera);
}

// Do this last
update();
```

15

Debugging your program

- Debugging GLSL programs can be challenging. Keep calm. Many problems are due to strict typing. E.g., float literals must use decimal point
- Good news: easy to run and see results. No compilation step. Test code as you write it.
- Browsers provide some tools for JavaScript debugging, but not for GLSL programs
  - Toggle console with, e.g., <F12>
  - Reload page with CTRL-R

16

Next

- Geometry 1: Points and Vectors
- Homework: read Textbook Chapter 2

17