WebGL Fragment Shader Profiler

Terry Sun, CIS565 Final Project

Project Idea

- A tool for profiling fragment shaders!
- A Chrome extension (!) which interacts with the shaders on a page and can profile them (semi-automatically) to show you which sections are taking longer.
  - Integrate with existing extension ShaderEditor (based of FF dev tools)
- Mouse over the page to see hotspots in particular pixel.

Motivation

- There are **many** tools for profiling Javascript WebGL applications, but none of them target the shaders.
- Shaders do a lot of heavy lifting, and can get very, very complicated.

Profiling

- WebGL Disjoint Timer Query API
  - New! In Chrome Canary
  - … which is not built for Linux, so I'm using a chromium build
  - Actually quite easy and looks like it works pretty well.

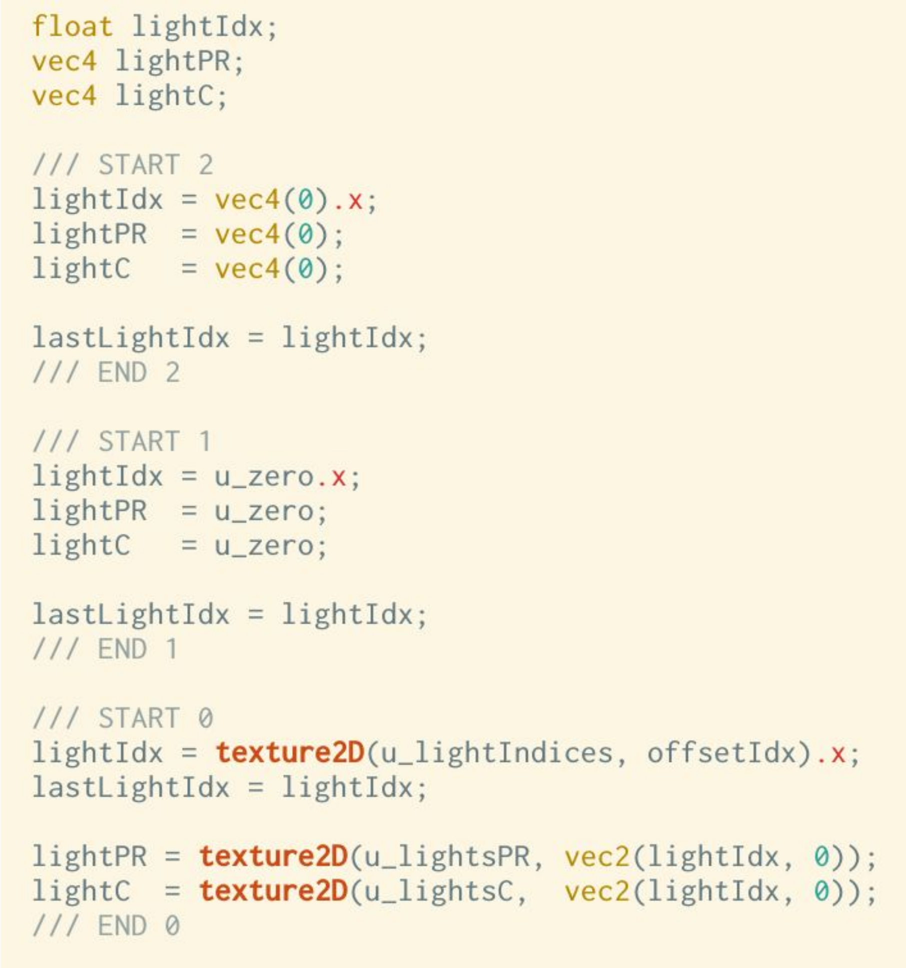
Profiling

- Measure performance impact of sections of code by modifying the shader to omit those parts, then comparing performance of the new shader.
  - User-provided markup in shaders
  - Auto-replace certain function calls (using the AST?)
    - texture2D, trig, loop bodies, user-defined functions
  - Replace big textures with 1x1 textures

Progress

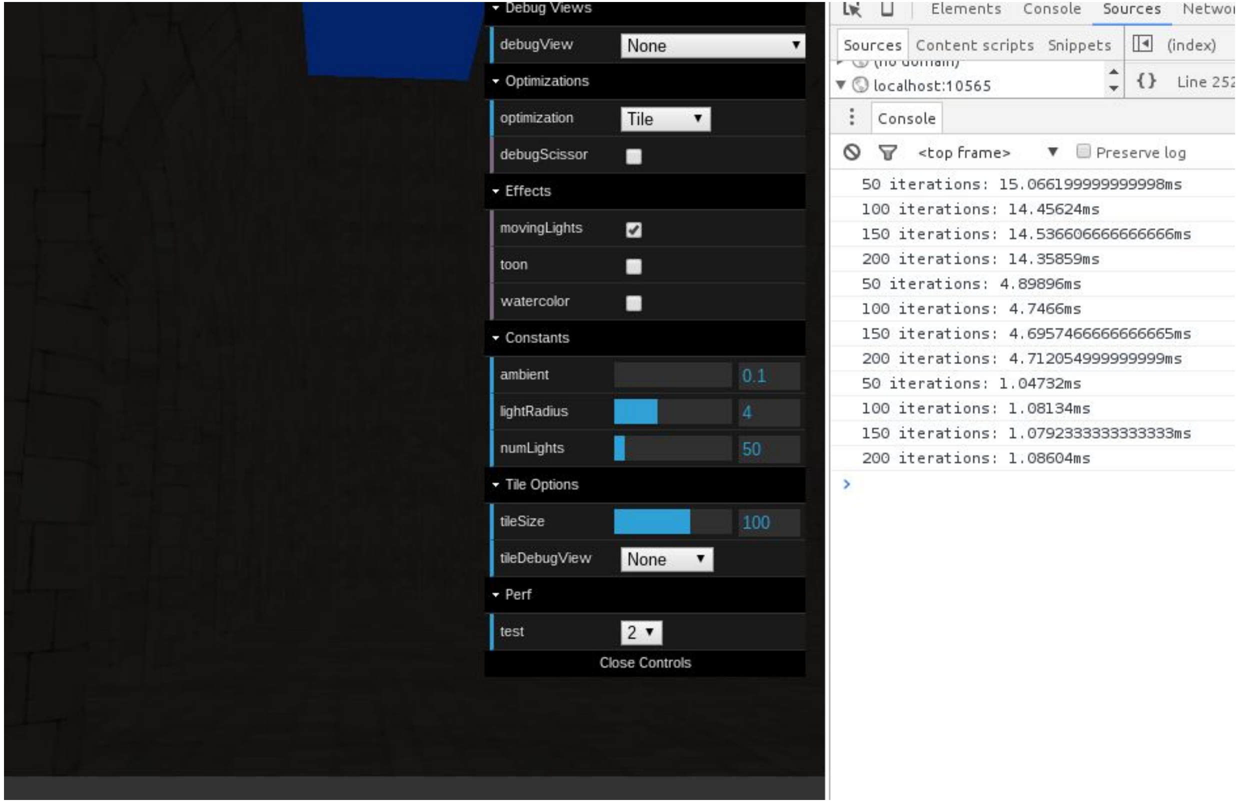
- Proof of concept
  - Using disjoint_timer_query
  - With user markup to modify shader
  - Re-compiling shader and taking updated timing data
  - Everything is more-or-less self-contained
    - Making API calls from deferred shader

Progress



```
float lightIdx;
vec4 lightPR;
vec4 lightC;

/// START 2
lightIdx = vec4(0).x;
lightPR  = vec4(0);
lightC   = vec4(0);

lastLightIdx = lightIdx;
/// END 2

/// START 1
lightIdx = u_zero.x;
lightPR  = u_zero;
lightC   = u_zero;

lastLightIdx = lightIdx;
/// END 1

/// START 0
lightIdx = texture2D(u_lightIndices, offsetIdx).x;
lastLightIdx = lightIdx;

lightPR = texture2D(u_lightsPR, vec2(lightIdx, 0));
lightC  = texture2D(u_lightsC,  vec2(lightIdx, 0));
/// END 0
```

Progress



debugView: None

optimization: Tile

debugScissor: □

movingLights: ☑

toon: □

watercolor: □

ambient: 0.1

lightRadius: 4

numLights: 50

tileSize: 100

tileDebugView: None

test: 2

```
50 iterations: 15.066199999999998ms
100 iterations: 14.45624ms
150 iterations: 14.536606666666666ms
200 iterations: 14.35859ms
50 iterations: 4.89896ms
100 iterations: 4.7466ms
150 iterations: 4.6957466666666665ms
200 iterations: 4.712054999999999ms
50 iterations: 1.04732ms
100 iterations: 1.08134ms
150 iterations: 1.0792333333333333ms
200 iterations: 1.08604ms
```

Milestones

- Milestone 1
  - Play with shader analysis/modification
  - Test mouse interaction
- Milestone 2
  - Automatic(ish) shader modification
  - Single-pixel analysis
  - Generate nice output/graphs
  - Look at ShaderEditor (if time)

Milestones

- Milestone 3
  - Integrate with ShaderEditor Chrome extension
  - Generate nice output/graphs
- Final Presentation
  - Fix all of the things that are broken
  - Polish / nicer output and analysis