WebGLot

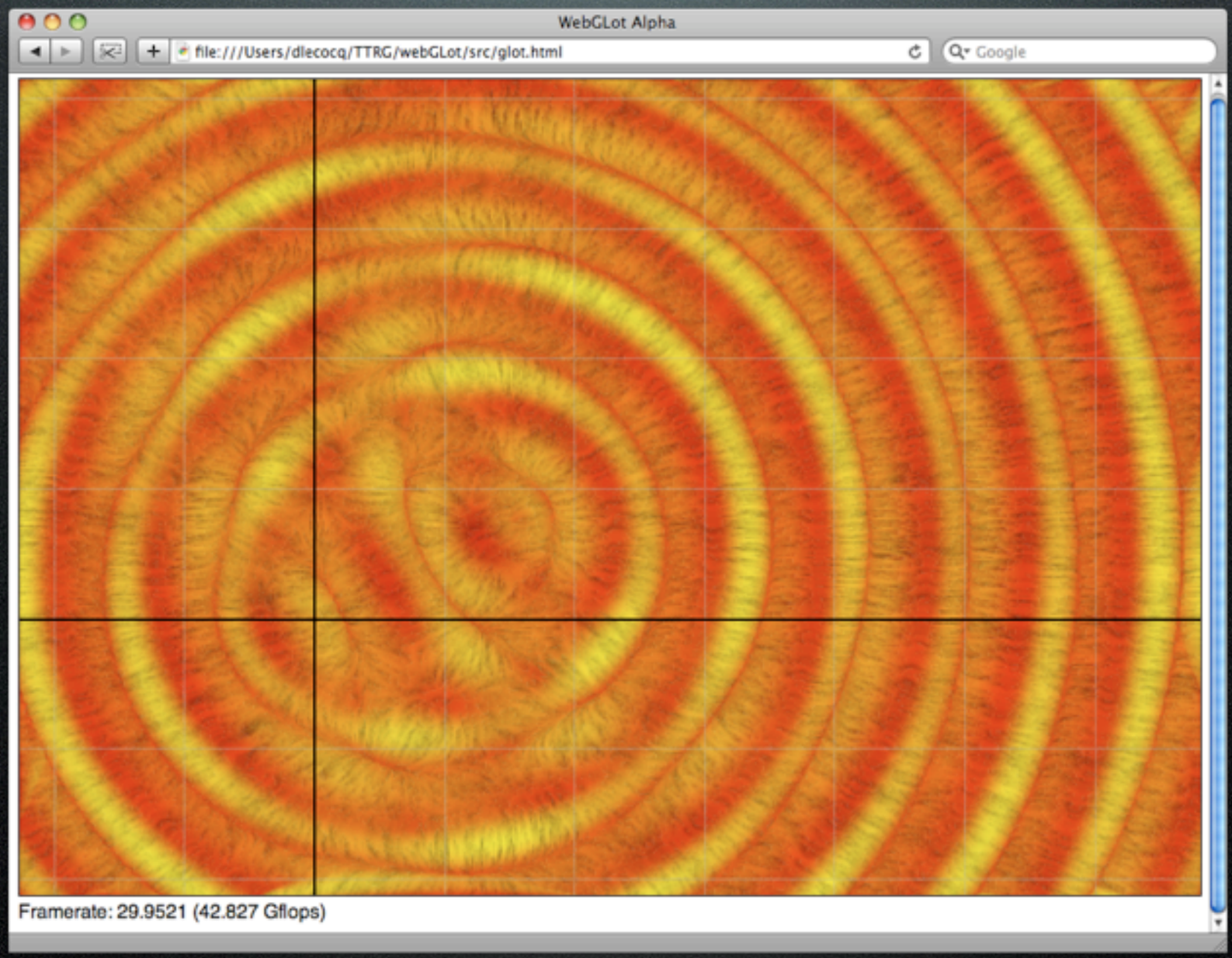
High-Performance Visualization in the Browser

Dan Lecocq

King Abdullah University of Science and Technology

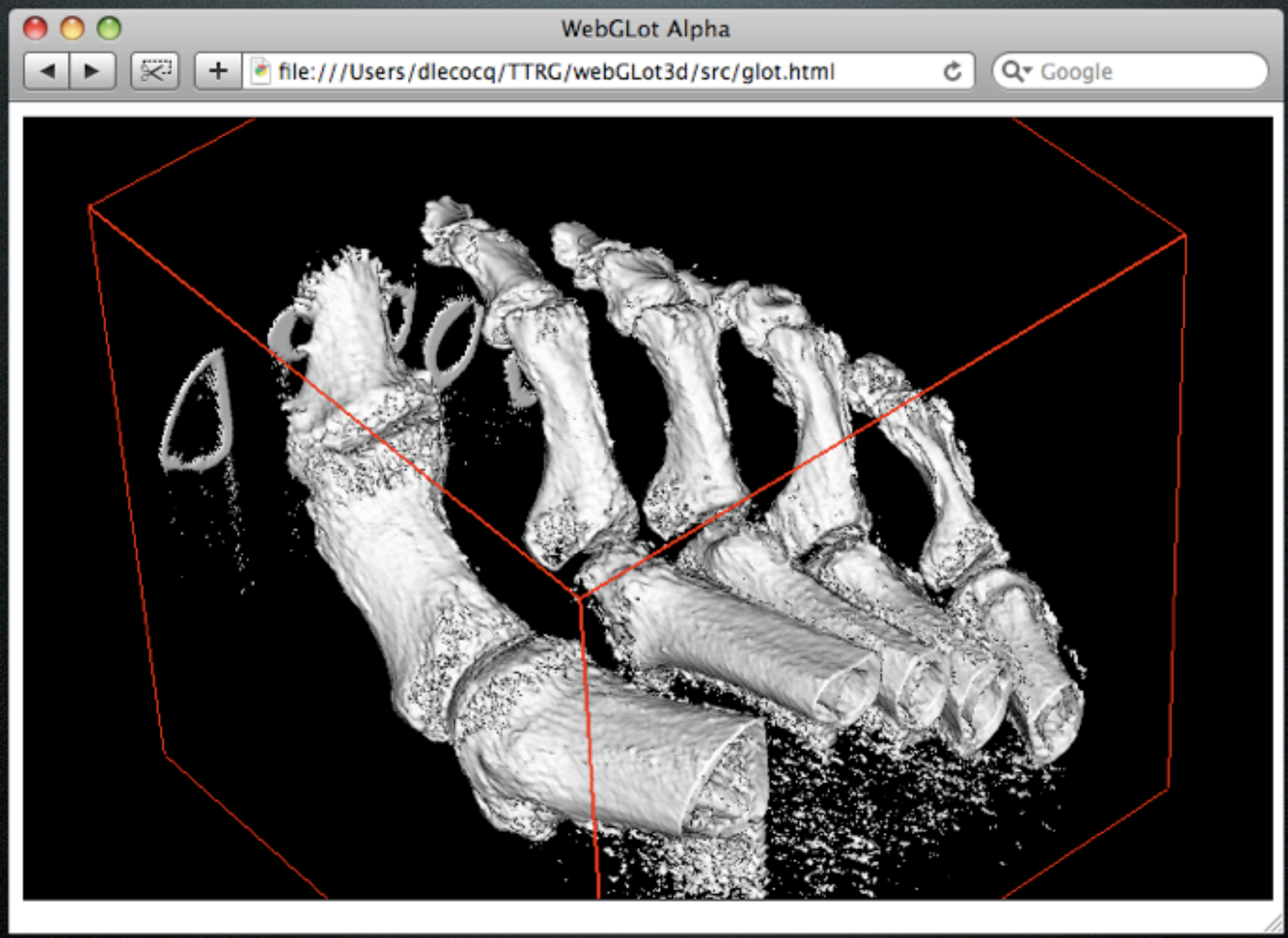
SIGGRAPH 2010

WebGLot is a Library...



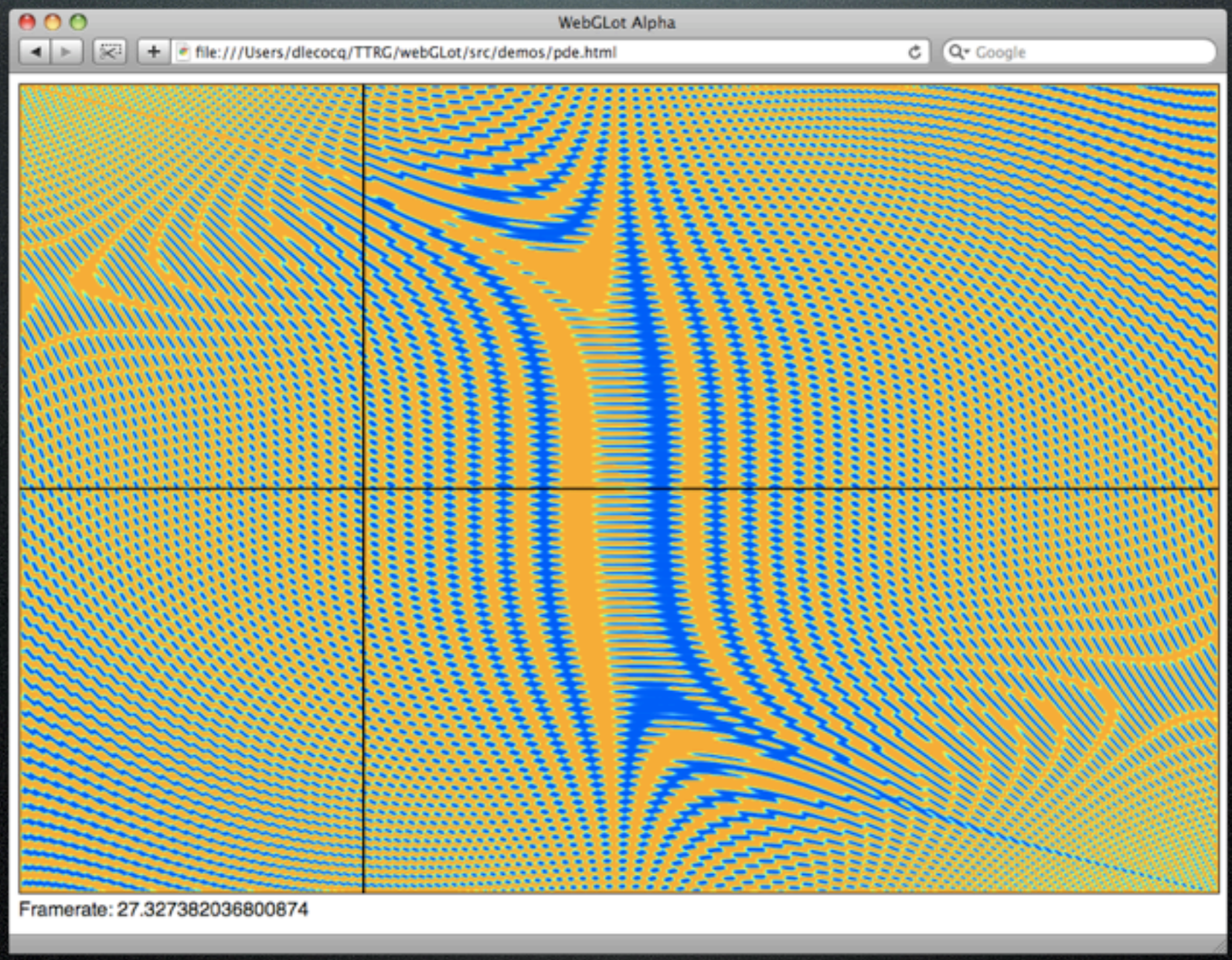
...for mathematical rendering

WebGLot is a Library...



...for data visualization

WebGLot is a Library...



...for some GPGPU applications

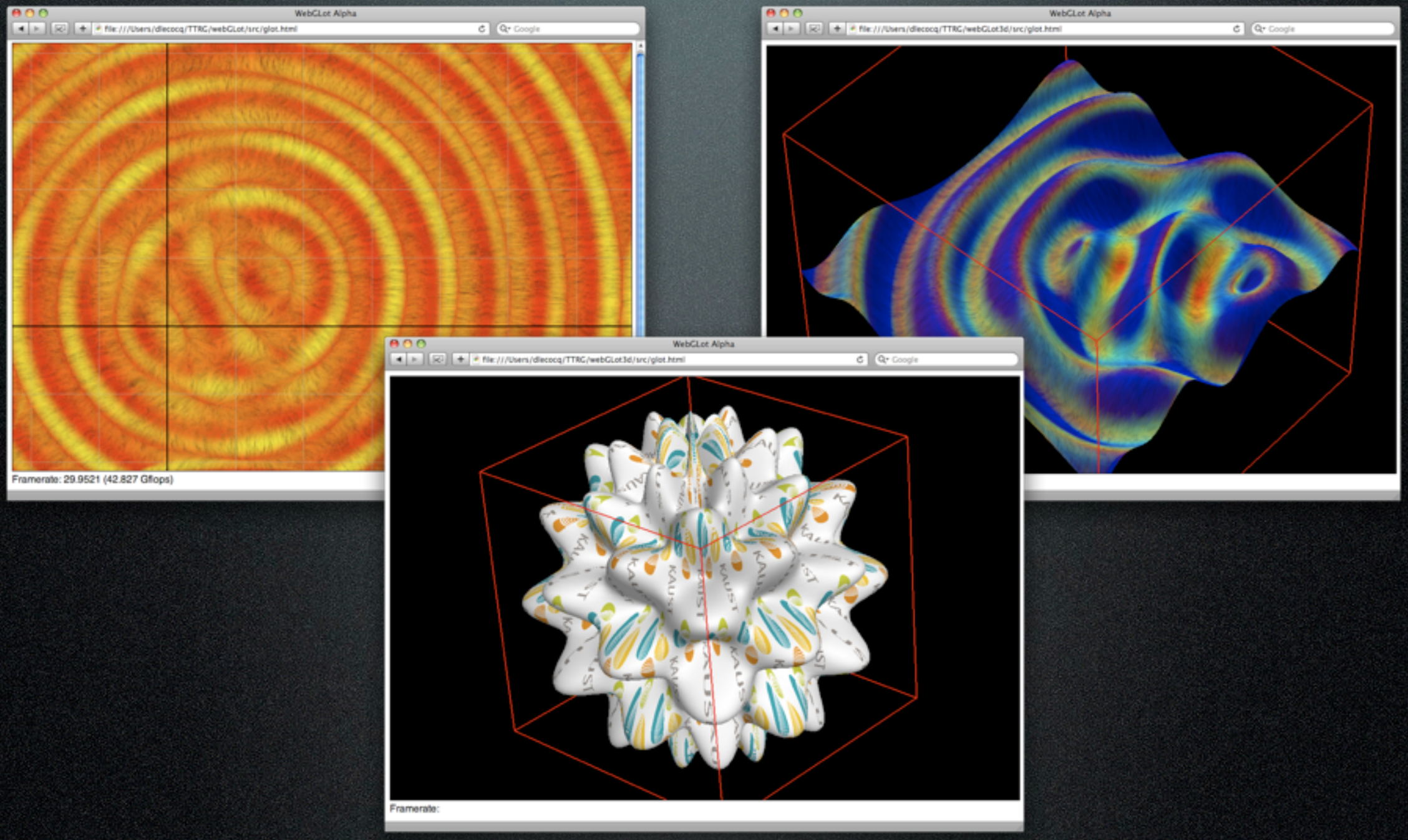
Started Out as Math

- WebGL + plot = WebGLot
- Library for interactive demos
- Frustrated with other tools
  - evaluated on CPU
  - Grapher.app, MATLAB, etc.

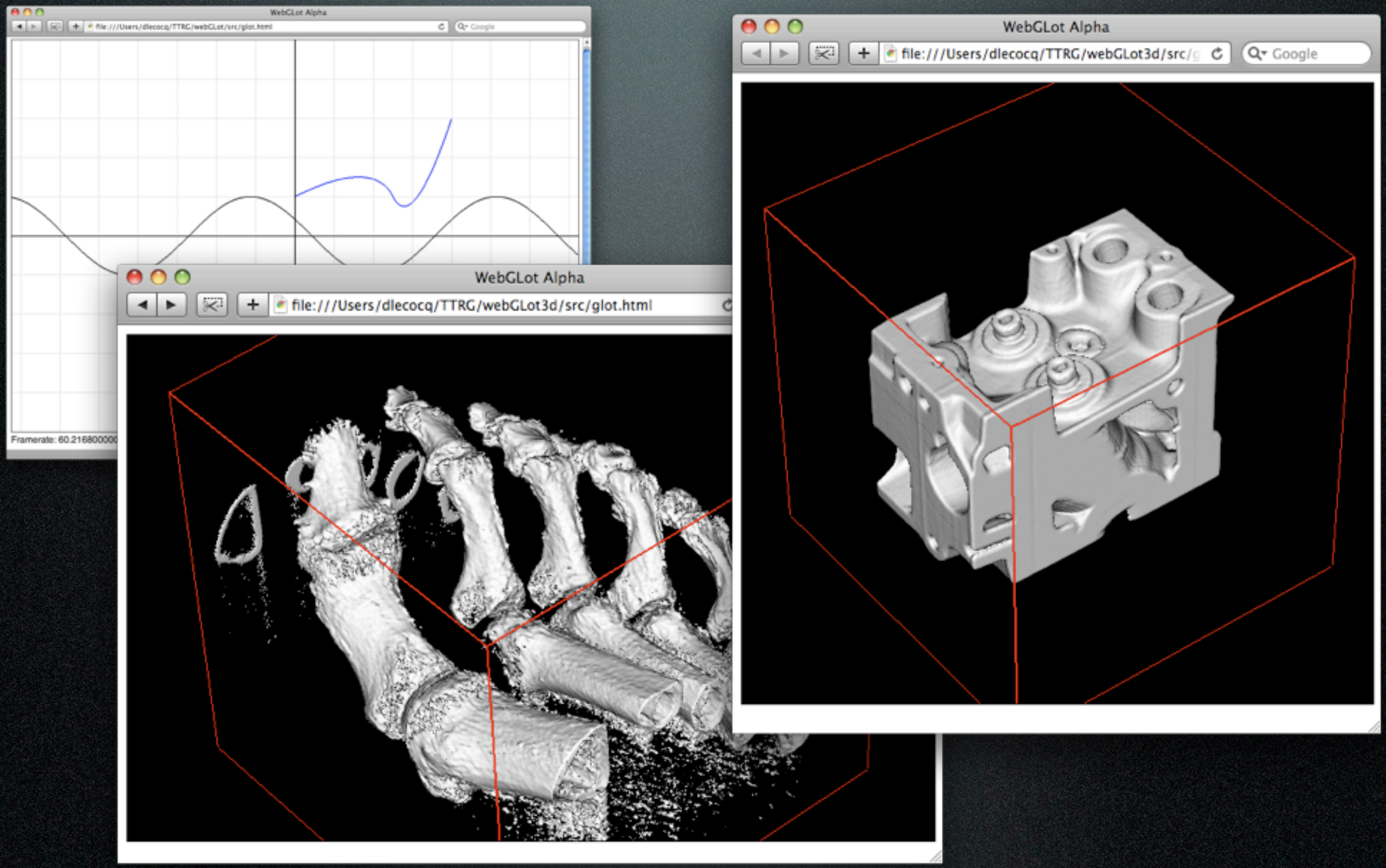
Why WebGLot

- **Fast** (GeForce 9400M)
  - Most primitives at 60fps
  - Complex primitives at 15-30fps
- **Portable**
  - Need browser, OpenGL impl.
  - No extensions, no Java, embedded

Mathematical Rendering



Data Visualization



Details

How the Magic Happens

WebGL

- JavaScript bindings to OpenGL ES 2.0
- Cross-platform
- Embeddable in HTML




Shader Generation

Uniform variables specified by the user

$\vec{f}(u, v)$ in C-like syntax

Coordinate transformations

p_surface.vert — src

p_surface.vert

```
2   uniform mat4 u_projectionMatrix;
3
4   attribute vec4 vTexCoord;
5   attribute vec4 vPosition;
6
7   varying vec2 v_texCoord;
8   varying vec3 normal;
9   varying vec3 light;
10  varying vec3 halfVector;
11
12  // USER_PARAMETERS
13
14  uniform float t;
15
16  vec4 function(float u, float v) {
17    vec4 result = vec4(USER_FUNCTION, 1.0);
18
19    /* Here are some transformations for coordinate systems.  To make
20     * this more transparent when it's used.  This way, any time you
21     * call ``function'' in this shader, you can be certain it has had
22     * the coordinate transformation performed on it.
23     */
24
25    /* CYLINDRICAL
26    result = vec4(result.y * cos(result.x), result.y * sin(result.x), result.z, 1.
27    //*/
28
29    /* SPHERICAL
30    result = vec4(result.x * sin(result.y) * cos(result.z), result.x * sin(result.
31    //*/
32
33    return result;
34  }
35
```

Line: 1 Column: 1 C Tab Size: 2

Parameters

- Programmer identifies additional parameters that will change often
  - `glot.set("alpha", 1);`
  - `value = glot.get("alpha");`
- Essentially an associative array, whose values get set in shaders
- Isovalues, frequencies, step lengths

User-Supplied Functions

- Users add primitives, passing in a string representation:
  - `glot.add(new surface("..."));`
- Other options are set here, like coordinate system
- String must correspond to algorithm
  - e.g. Three values for parametric

Advantages of Shader Generation

- Handles function parsing for us
  - Needs to be GLSL (C-style)
    - We do **some** extra string cleanup
- Change parameters quickly
  - Interactively explore

Getting Started

A VERY Quick Start

The Setup

demo.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello WebGLot World!</title>

  <!-- Just a little styling -->
  <style type="text">
  canvas {
    border: 1px solid black;
    width:100%;
    height:93%;
  }

  div {
    font-size: small;
  }
  </style>

  <!-- Include the WebGLot library -->
  <script src="grapher.js"></script>

  <script>
  var glot = null;
  var mySurface = null;

  function makeIsosurface() {
    // Get the user-supplied function
    f = document.forms["function"].elements.fn.value);
    if (mySurface) {
      glot.primitives.pop();
    }
    mySurface = new isosurface(f);
    glot.add(mySurface);
    glot.display();
    // So that it doesn't request a page refresh
    return false;
  }

  function start() {
    /* Instantiate a grapher */
    glot = new grapher();

    glot.setDomain(-2, 2, -2, 2);
    glot.add(new box());

    // You can add new primitives to the display with the add function.
    glot.set("isovalue", 0);

    /* Time-dependent primitives should be run */
    glot.run();
  }
  </script>
</head>

<body onload="start()">
  <!-- This is the canvas we'll display things in -->
  <canvas id="glot">It seems you do not have support for WebGL.</canvas>

  <div id="function_form">
    <form name="function" onsubmit="return makeIsosurface()">
      f(x, y, z, t) = <input size='100' name="fn" /> = 0
    </form>
  </div>
</body>
</html>
```

Line: 20 Column: 21 HTML Tab Size: 2

<!-- Include the WebGLot library -->
<script src="grapher.js"></script>

/* Instantiate a grapher */
glot = new grapher();

/* Redisplay regularly */
glot.run();

<!-- Display things in this canvas -->
<canvas id="glot">No WebGL!</canvas>

The Setup

demo.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello WebGlot World!</title>

  <!-- Just a little styling -->
  <style type="text">
  canvas {
    border: 1px solid black;
    width:100%;
    height:93%;
  }

  div {
    font-size: small;
  }
  </style>

  <!-- Include the WebGlot library -->
  <script src="grapher.js"></script>

  <script>
  var glot = null;
  var mySurface = null;

  function makeIsosurface() {
    // Get the user-supplied function
    f = document.forms["function"].elements.fn.value);
    if (mySurface) {
      glot.primitives.pop();
    }
    mySurface = new isosurface(f);
    glot.add(mySurface);
    glot.display();
    // So that it doesn't request a page refresh
    return false;
  }

  function start() {
    /* Instantiate a grapher */
    glot = new grapher();

    glot.setDomain(-2, 2, -2, 2);
    glot.add(new box());

    // You can add new primitives to the display with the add function.
    glot.set("isovalue", 0);

    /* Time-dependent primitives should be run */
    glot.run();
  }
  </script>
</head>

<body onload="start()">
  <!-- This is the canvas we'll display things in -->
  <canvas id="glot">It seems you do not have support for WebGL.</canvas>

  <div id="function_form">
    <form name="function" onsubmit="return makeIsosurface()">
      f(x, y, z, t) = <input size='100' name="fn" /> = 0
    </form>
  </div>
</body>
</html>
```

Line: 20 Column: 21 HTML Tab Size: 2

```
function makeIsosurface() {
    // Get user-supplied function
    form = document.forms["function"];
    f = form.elements.fn.value);
    if (mySurface) {
        glot.primitives.pop();
    }
    mySurface = new isosurface(f);
    glot.add(mySurface);
    glot.display();
    // JavaScript quirk
    return false;
}
```

The Setup

demo.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello WebGLot World!</title>

  <!-- Just a little styling -->
  <style type="text">
  canvas {
    border: 1px solid black;
    width:100%;
    height:93%;
  }

  div {
    font-size: small;
  }
  </style>

  <!-- Include the WebGLot library -->
  <script src="grapher.js"></script>

  <script>
  var glot = null;
  var mySurface = null;

  function makeIsosurface() {
    // Get the user-supplied function
    f = document.forms["function"].elements.fn.value);
    if (mySurface) {
      glot.primitives.pop();
    }
    mySurface = new isosurface(f);
    glot.add(mySurface);
    glot.display();
    // So that it doesn't request a page refresh
    return false;
  }

  function start() {
    /* Instantiate a grapher */
    glot = new grapher();

    glot.setDomain(-2, 2, -2, 2);
    glot.add(new box());

    // You can add new primitives to the display with the add function.
    glot.set("isovalue", 0);

    /* Time-dependent primitives should be run */
    glot.run();
  }
  </script>
</head>

<body onload="start()">
  <!-- This is the canvas we'll display things in -->
  <canvas id="glot">It seems you do not have support for WebGL.</canvas>

  <div id="function_form">
    <form name="function" onsubmit="return makeIsosurface()">
      f(x, y, z, t) = <input size='100' name="fn" /> = 0
    </form>
  </div>
</body>
</html>
```

Line: 20 Column: 21 HTML Tab Size: 2

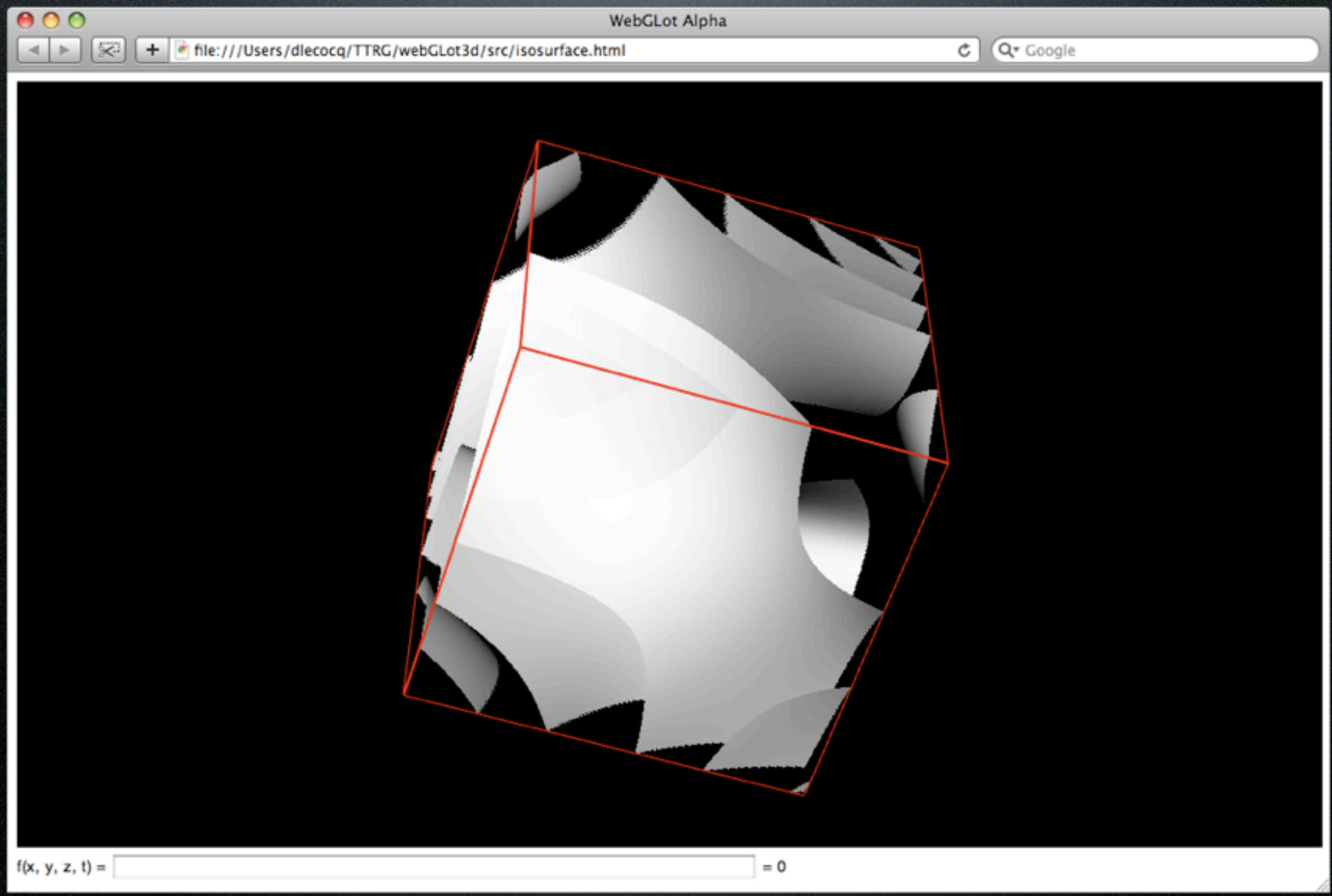
Some CSS for appropriately sizing the display area

```
<div id="function_form">
    <form onsubmit="makeIsosurface()">
        <input size='100' name="fn" /> = 0
    </form>
</div>
```

Live Demo

Fingers crossed!

In Action


WebGLot Alpha
file:///Users/dlecocq/TTRG/webGLot3d/src/isosurface.html
Google
f(x, y, z, t) =
= 0

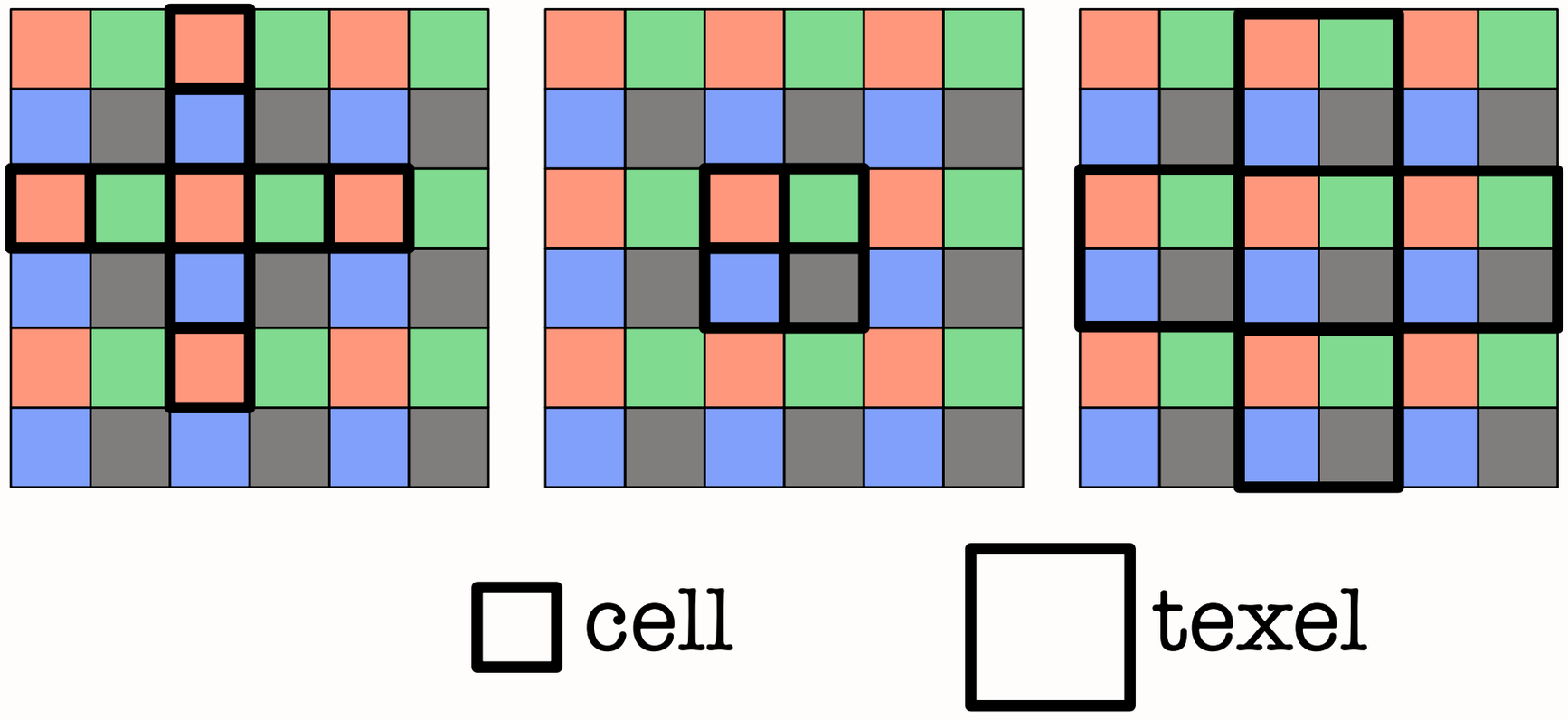
GPGPU

Solving the Poisson Equation

PDE Solver

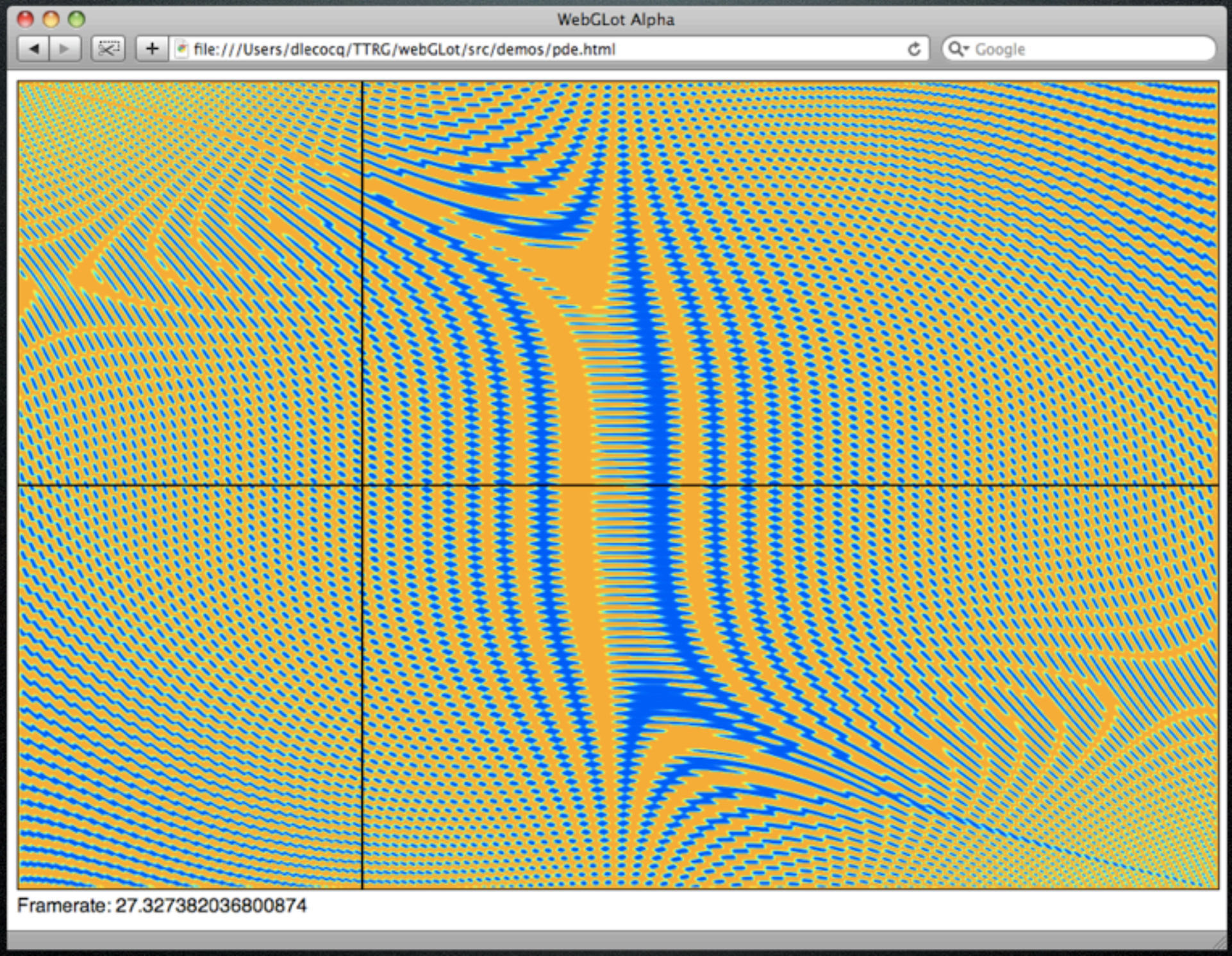
- Ping-pong rendering
  - 9-point Jacobi kernel Poisson eqn.
- 24 Gflops (45% efficiency)
  - MacBook NVIDIA GeForce 9400M
  - 2400 x 1300 domain

Kernel Specifics



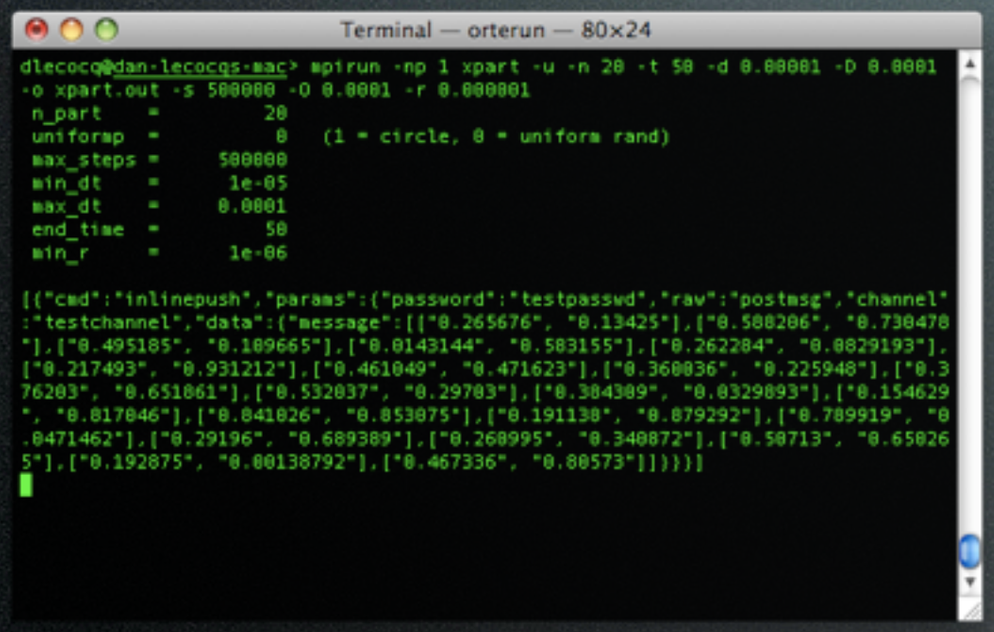
- 4 stores, only 5 fetches
- Improves calculation/fetch ratio

PDE Solver


WebGLot Alpha
file:///Users/dlecocq/TTRG/webGLot/src/demos/pde.html
Google
Framerate: 27.327382036800874

Streaming Visualization

Streaming Viz

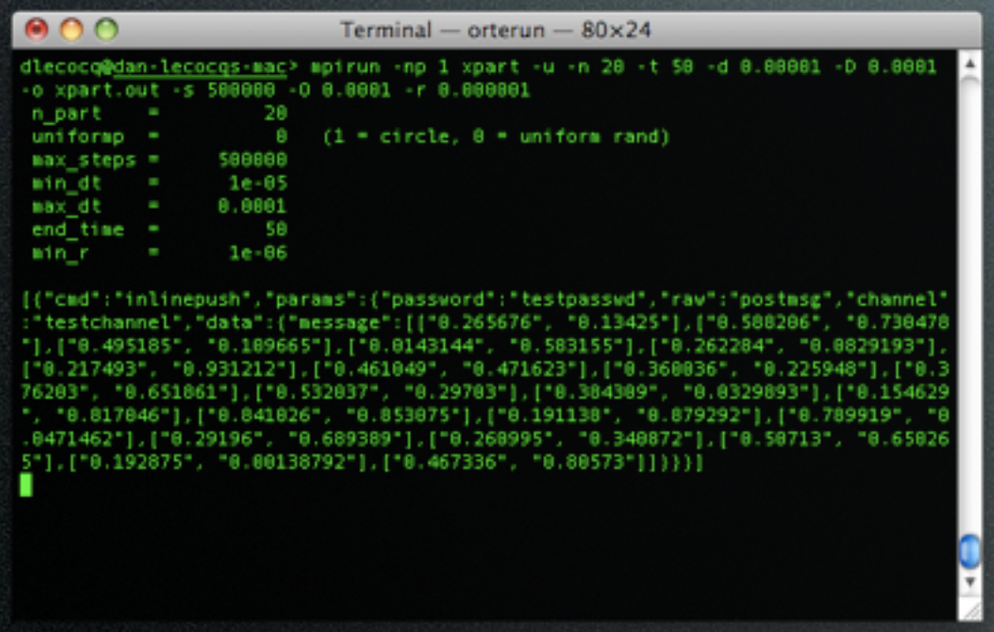


APE Server

APE

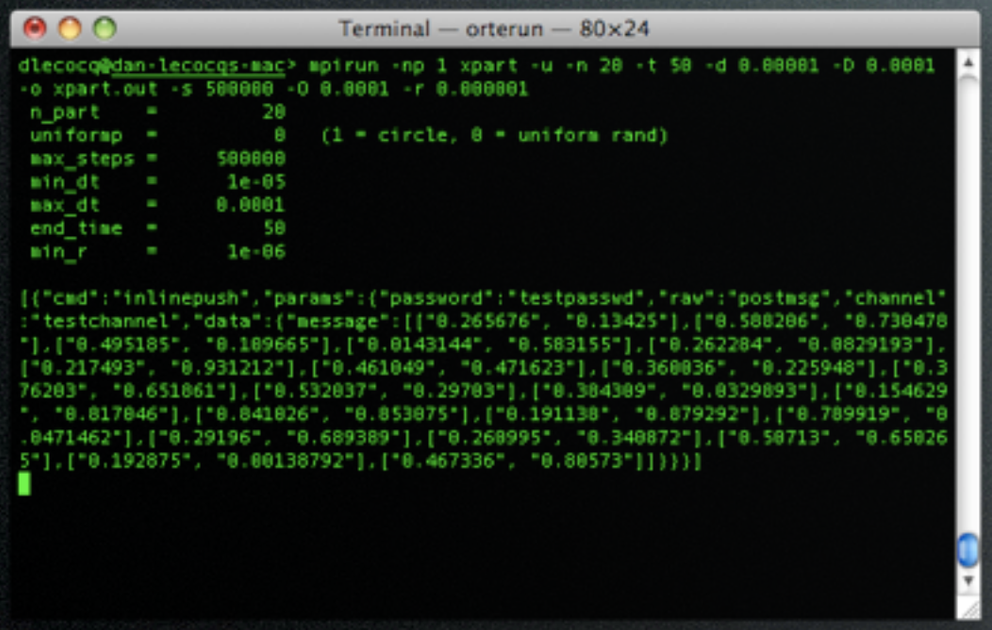
- AJAX Push Engine
  - Comet - server push
  - AJAX - user-initiated
- http://www.ape-project.org/ajax-push.html
- Scalable broadcast

Streaming Viz

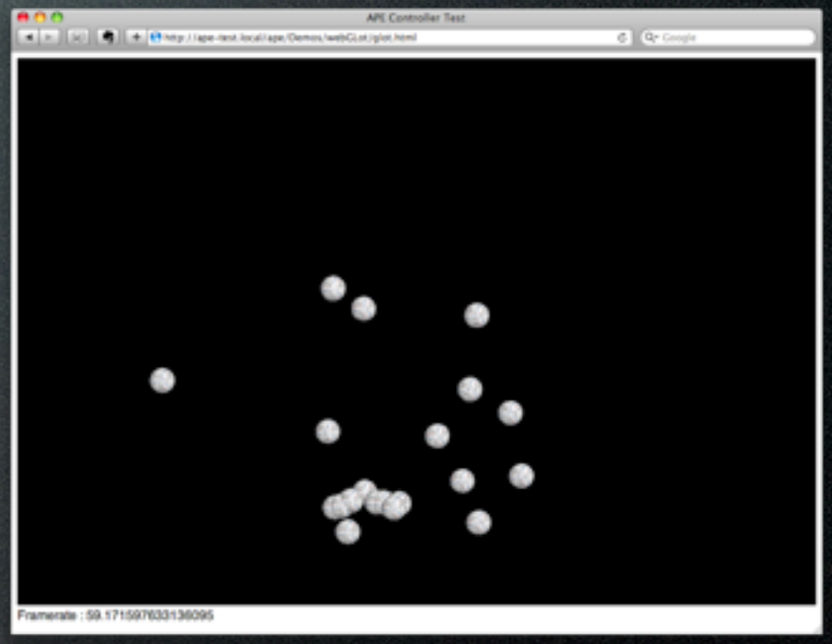


APE Server

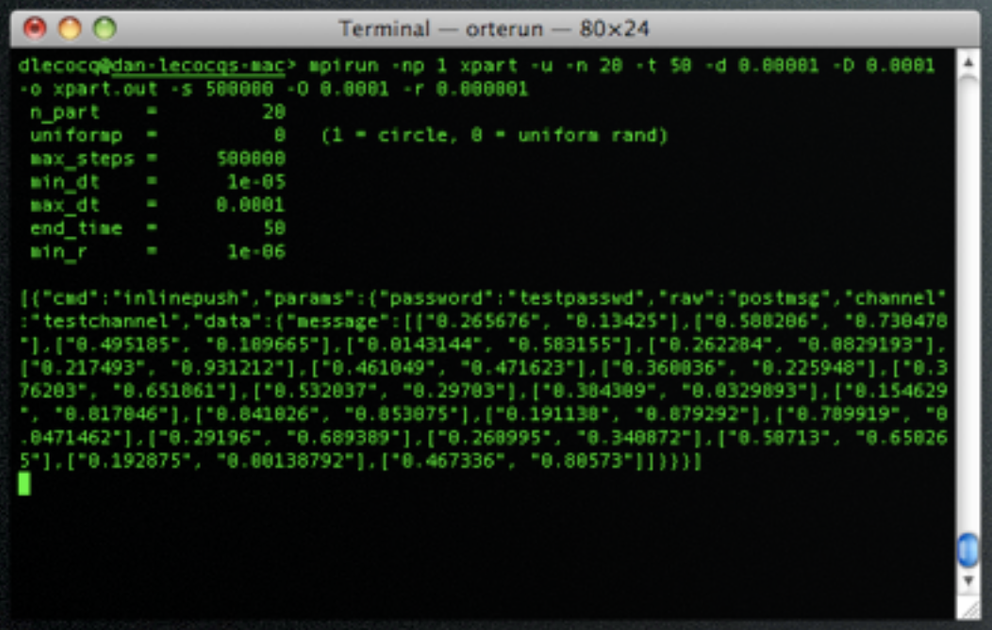
Steered Computation



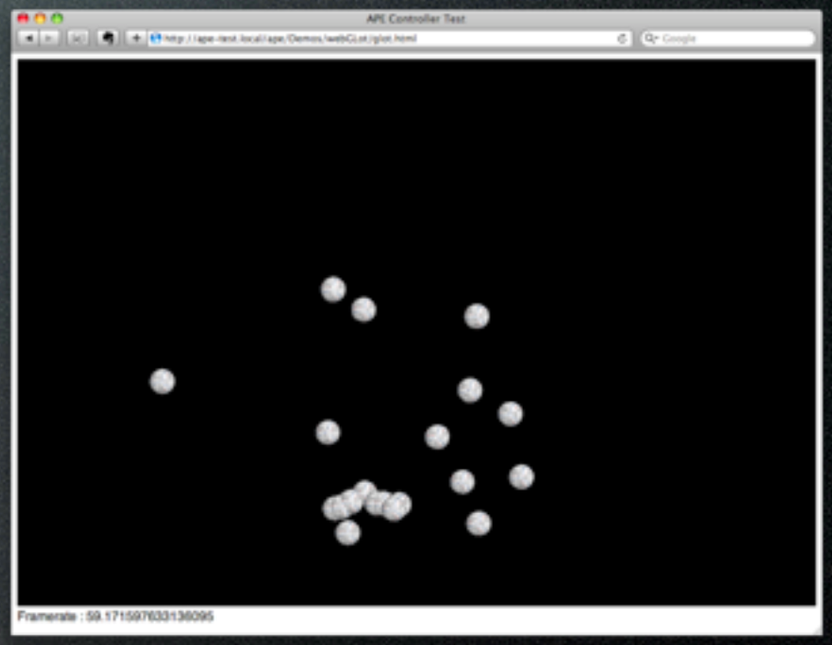
WebSockets



Steered Computation



WebSockets



HTML5

Challenges

- Working with code under development
  - Nightly builds => not all features
    - Floating-point texture extension
- OpenGL ES 2.0
  - No 3D textures
  - No geometry shaders (isocontours)

No 3D Textures

- Tile slices into 2D texture
  - This is the engine block volume:

Future Directions

Future Directions

- Solidified Streaming Visualization
- API, utilities, bindings

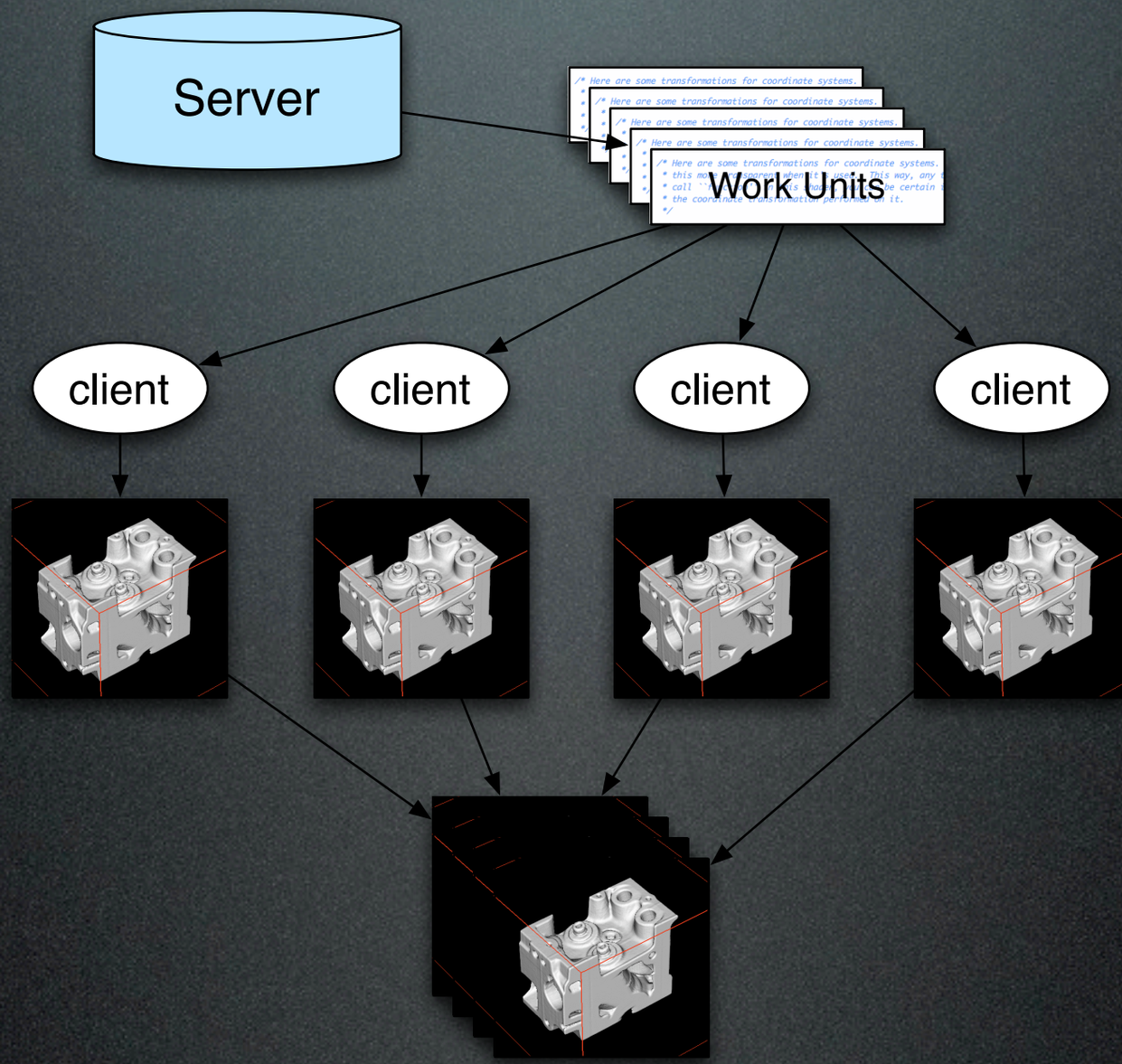
Future Directions

- Solidified Streaming Visualization
  - API, utilities, bindings
- Computational Steering

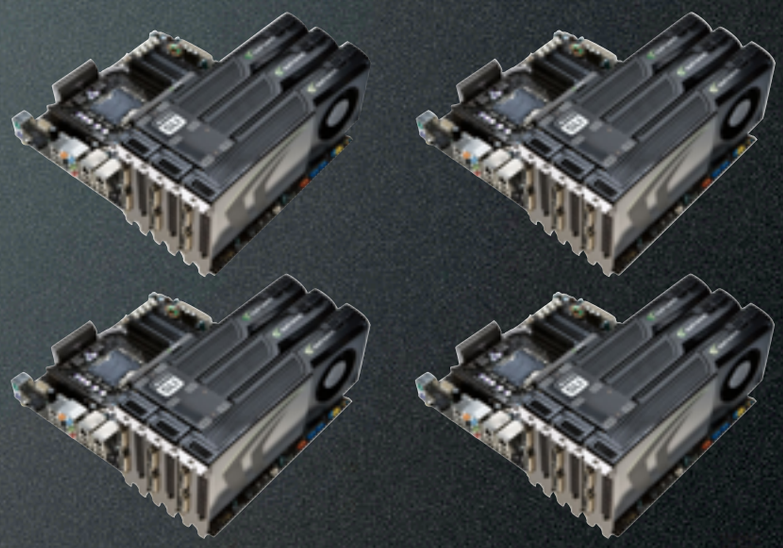
Future Directions

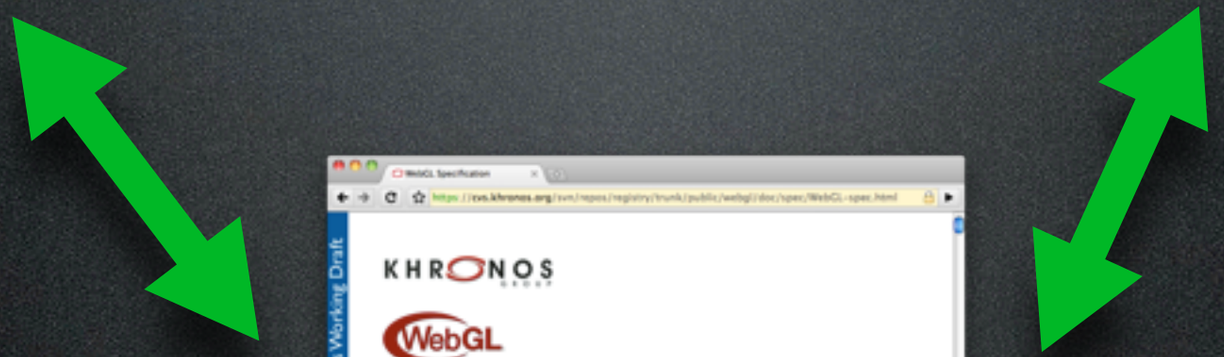
- Solidified Streaming Visualization
  - API, utilities, bindings
- Computational Steering
- Distributed GPGPU

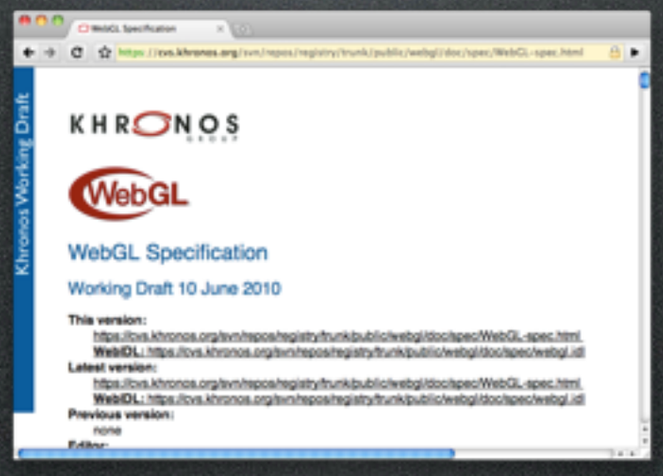
Distributed Rendering



Steering and Streaming



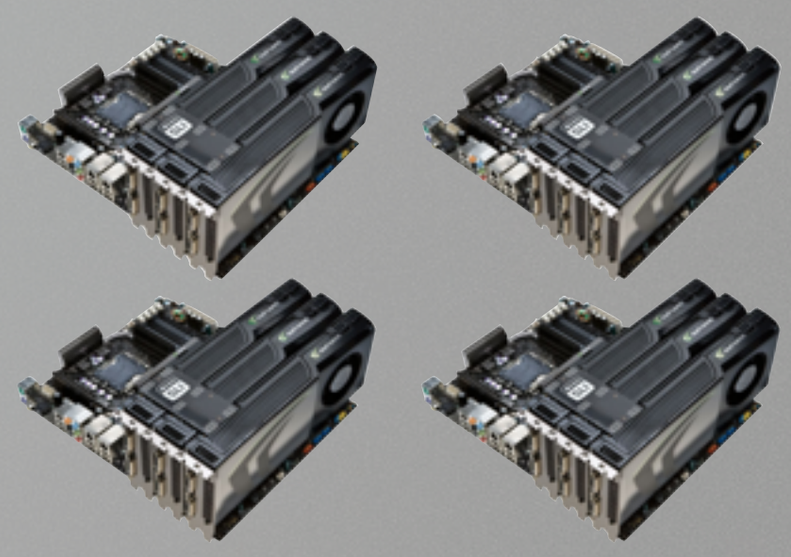


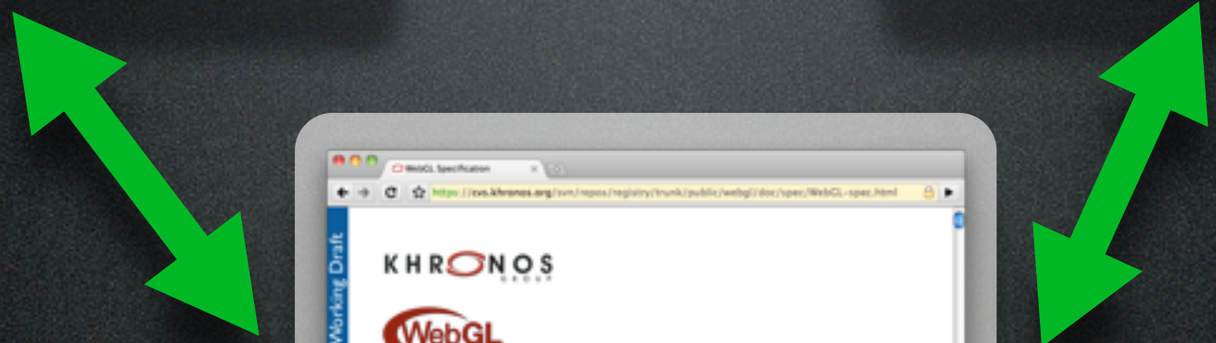


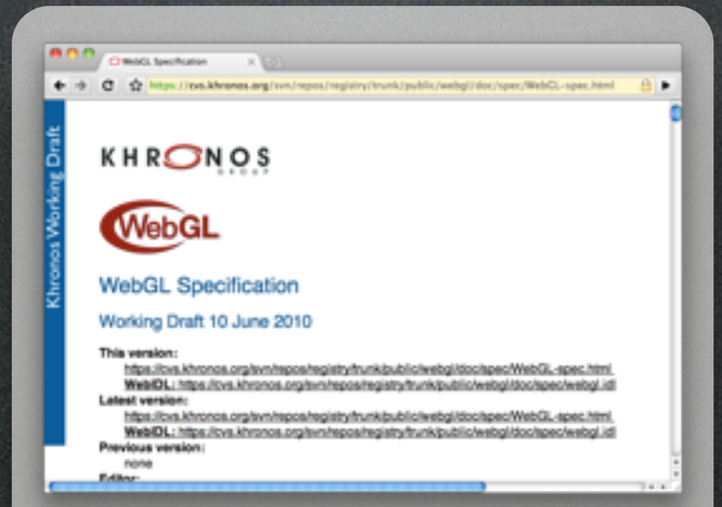
Steering and Streaming

CPU

GPU







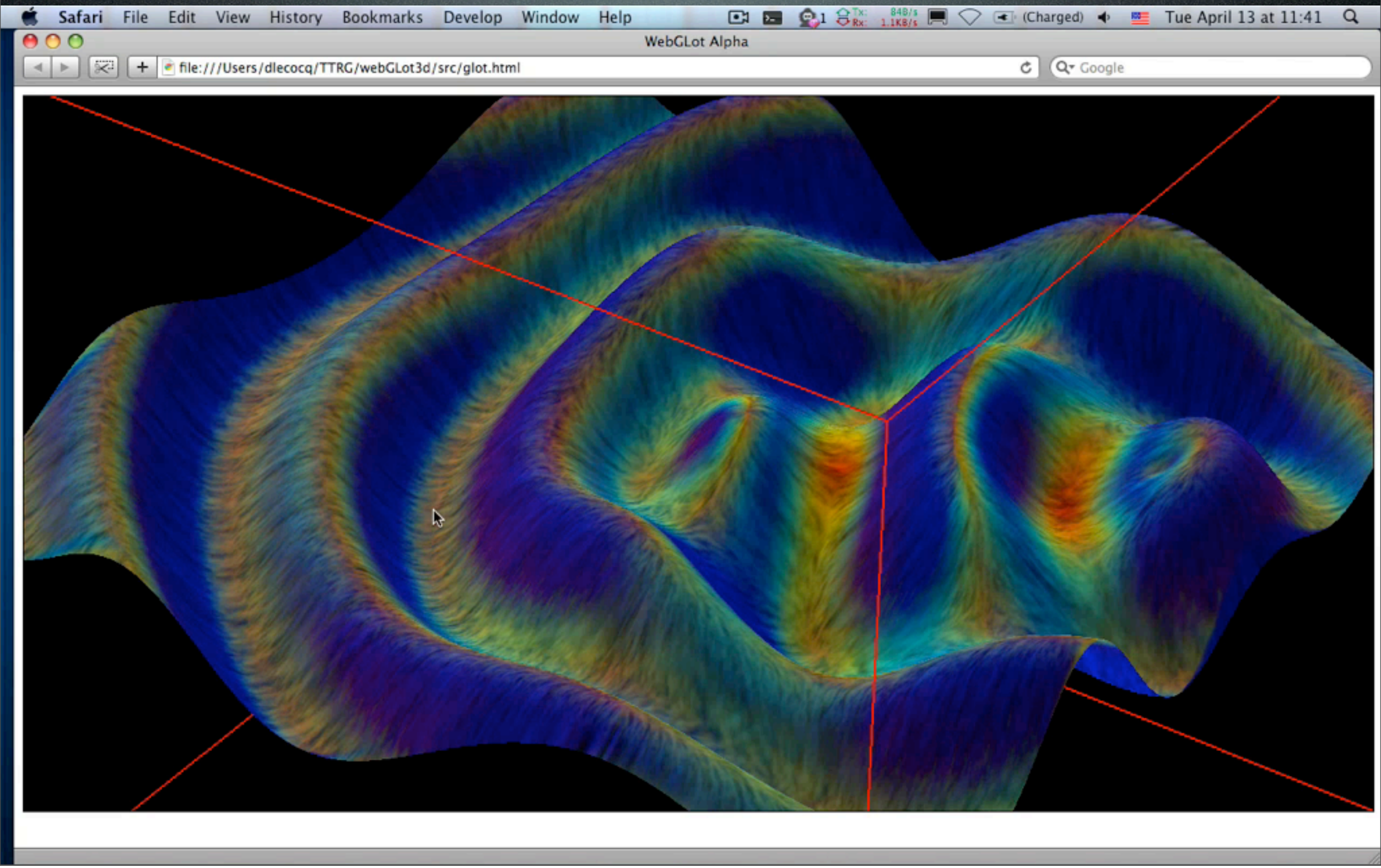
UI

Goals

- **Portable**
  - Run in browsers on any platform
- **Lightweight**
  - Code about 200kB
- **Powerful**
  - Enable science

Questions?

Questions?


Safari File Edit View History Bookmarks Develop Window Help
Tue April 13 at 11:41
WebGLot Alpha
file:///Users/dlecocq/TTRG/webGLot3d/src/glot.html
Google