DSCI 554 LECTURE 11

3D DATA VISUALIZATION TOOLS

Dr. Luciano Nocera

USC Viterbi
School of Engineering
Integrated Media Systems Center

OUTLINE

- Overview
- Canvas
- WebGL
- three.js
- Processing, Processing.js, P5.js
- Vtk.js
- Mapbox GL & Deck.gl

VISUALIZATION TOOLS

Ease-of-Use ↑

Chart Typologies
Excel, Google Sheets, Matplotlib, Seaborn

Visual Analysis Grammars
VizQL, Tableau, ggplot2, plotnine, Altair

Visualization Grammars
Protovis, D3, Vega, Vega-Lite

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics applications
Processing, P5.js, WebGL, three.js, OpenGL

Expressiveness ↓

already covered | covered today

Adapted from [Heer 2014]

OUTLINE

- Overview
- Canvas
- WebGL
- three.js
- Processing, Processing.js, P5.js
- Vtk.js
- Mapbox GL & Deck.gl

CANVAS

A raster (pixels) surface with two interfaces:

- **Canvas API** for drawing graphics via JavaScript and the HTML <canvas>

```
<canvas id="canvas" width="150" height="150"></canvas>

<script type="application/javascript">
  var canvas = document.getElementById("canvas");
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d');  //'2d' provides the Canvas API
    //use Canvas API to draw
  }
</script>
```

- **WebGL API** for rendering high-performance interactive 3D and 2D graphics

```
<canvas id="glCanvas" width="100" height="100"></canvas>
<script type="application/javascript">
  main();
  function main() {
    const canvas = document.querySelector("#glCanvas");
    const gl = canvas.getContext("webgl");  //initialize GL context
    //use WebGL API to draw
  }
</script>
```

CANVAS API EXAMPLE

```
<canvas style="background-color: orangered;" width="200" height="200"></canvas>
<canvas id="canvas" width="150" height="150"></canvas>
<script type="application/javascript">
  var canvas = document.getElementById("canvas");
  if (canvas.getContext) {
    var ctx = canvas.getContext('2d'); //initialize 2d context

    ctx.fillStyle = 'rgb(200, 0, 0)';
    ctx.fillRect(10, 10, 50, 50);

    ctx.fillStyle = 'rgba(0, 0, 200, 0.5)';
    ctx.fillRect(30, 30, 50, 50);
  }
</script>
```

See MDN Canvas API and Mike Bostock's World Tour observable with D3.

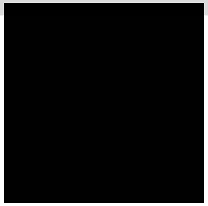
D3 WITH CANVAS API EXAMPLE

```
map = {
  const context = DOM.context2d(width, height);
  const path = d3.geoPath(projection, context);
  context.save();
  context.beginPath(), path(outline), context.clip(), context.fillStyle = "#fff", context.fillRect(0, 0, width, height);
  context.beginPath(), path(graticule), context.strokeStyle = "#ccc", context.stroke();
  context.beginPath(), path(land), context.fillStyle = "#000", context.fill();
  context.restore();
  context.beginPath(), path(outline), context.strokeStyle = "#000", context.stroke();
  return context.canvas;
}
```

Example from World Map (Canvas) Observable

WEBGL API EXAMPLE

```
<canvas id="glCanvas" width="100" height="100"></canvas>
<script type="application/javascript">
  main();
  function main() {
    const canvas = document.querySelector("#glCanvas");
    const gl = canvas.getContext("webgl");  //initialize GL context

    if (gl === null) {
      alert("Unable to initialize WebGL.");
      return;
    }

    gl.clearColor(0.0, 0.0, 0.0, 1.0);  //set clear color to black
    gl.clear(gl.COLOR_BUFFER_BIT);  //clear color buffer
  }
</script>
```



Documentation: MDN WebGL API, example presented from MDN webgl creation sample.

CANVAS VS. SVG

SVG (VECTOR)

- Simpler to use
- SVG redraw inefficient for large datasets
- SVG interactivity suffers with large datasets

CANVAS (RASTER) ADVANTAGES

- More complex to use
- Draw happens in immediate mode (DOM not involved)
- Uses GPU for rendering
- Stores data in graphic card memory for faster access

OUTLINE

- Overview
- Canvas
- WebGL
- three.js
- Processing, Processing.js, P5.js
- Vtk.js
- Mapbox GL & Deck.gl

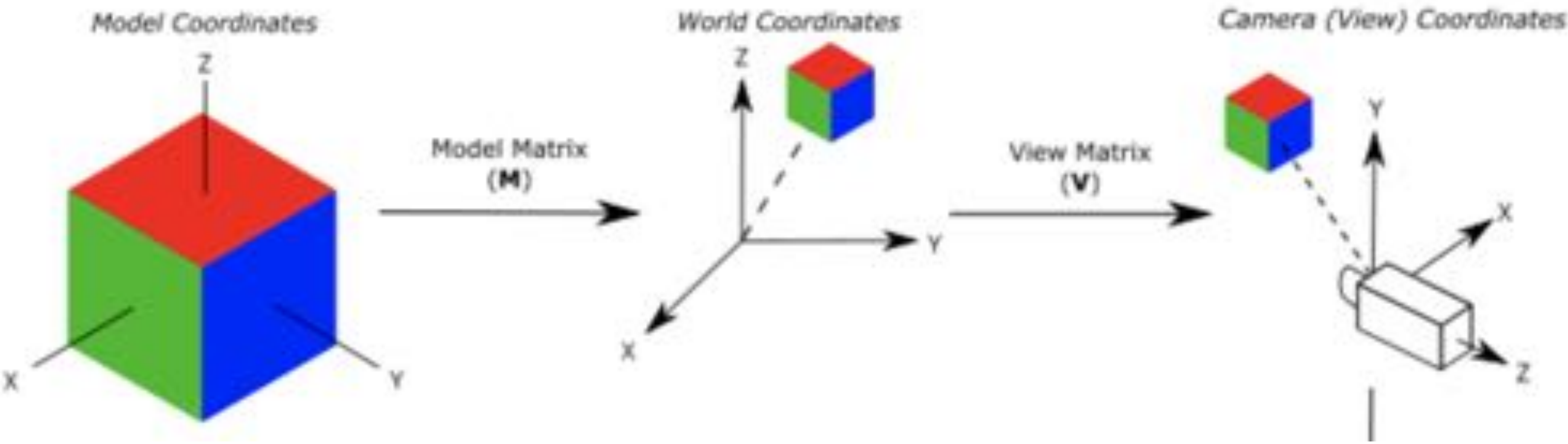
WEBGL

- Web Graphics Library, a W3 standard
- JS API for OpenGL ES 2.0 (OpenGL for mobile devices)
- 2D & 3D interactive rendering in HTML5 canvas
- GPU accelerated rendering
- Popular JS libraries based on WebGL:
  - three.js
  - PhiloGL
  - GLGE
  - P5.js

WEBGL VS. SVG

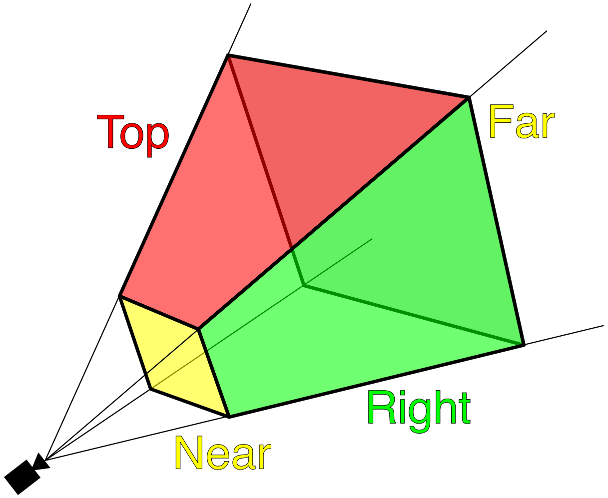
	SVG	WebGL
Supports 3D content		✓
DOM Interaction	✓	
Declarative scenegraph		✓
CSS Integration	✓	
Scripting access	✓	✓

SCENE GEOMETRY



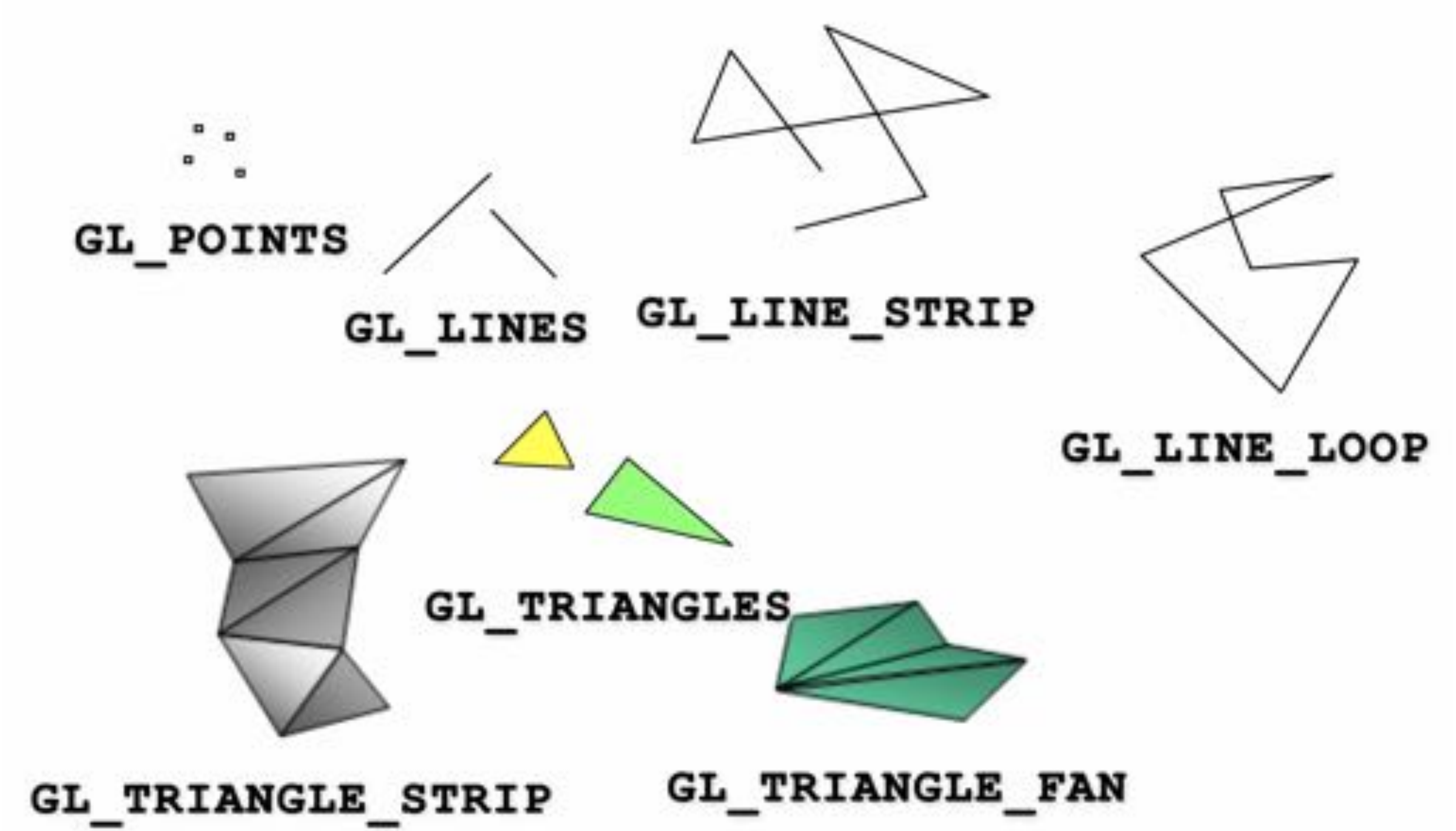
1. A **projection matrix** converts world space coordinates into clip space coordinates
2. A **model matrix**, takes model data and moves it around in 3D world space
3. A **view matrix** is used to move objects in the scene to simulate the position of the camera being changed

PROJECTION MATRIX



- The **projection matrix** is used to project the scene
- A **projection matrix** is defined by a view frustrum and clipping planes
- The **view frustum** defines the region whose contents are visible to the user
- **Clipping planes** further limit what is visible in the view frustrum
- Projection matrices are usually set to perspective or orthographic

OBJECTS ARE BUILT FROM PRIMITIVES



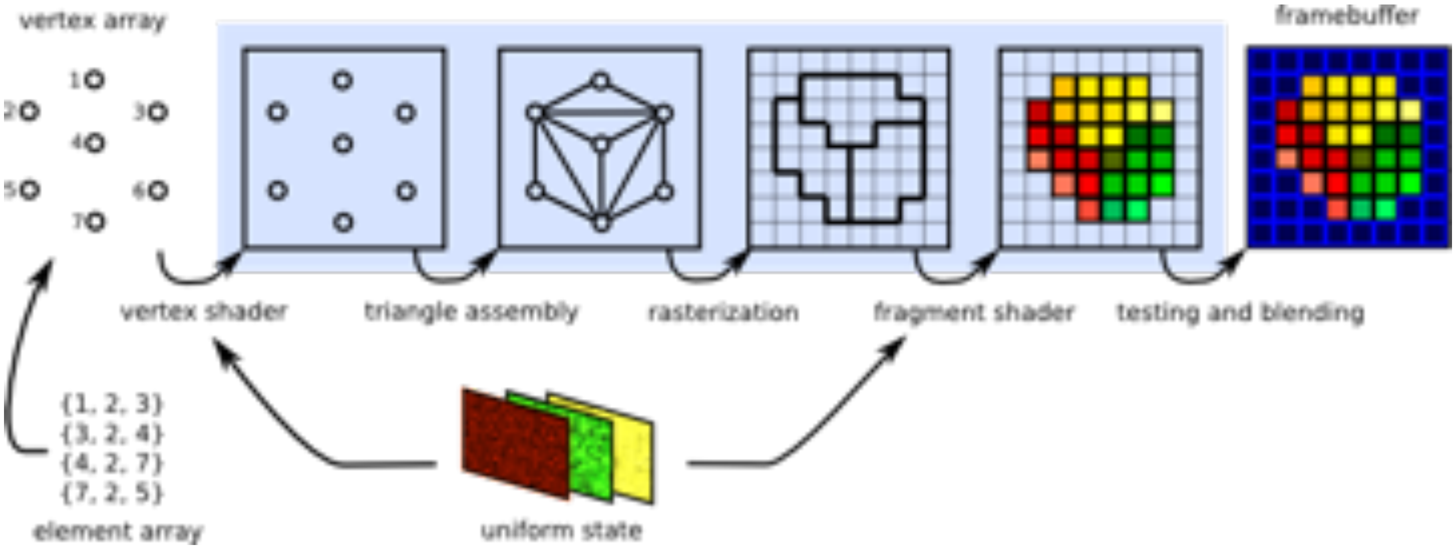
WebGL uses the same primitives than OpenGL

OBJECTS DATA

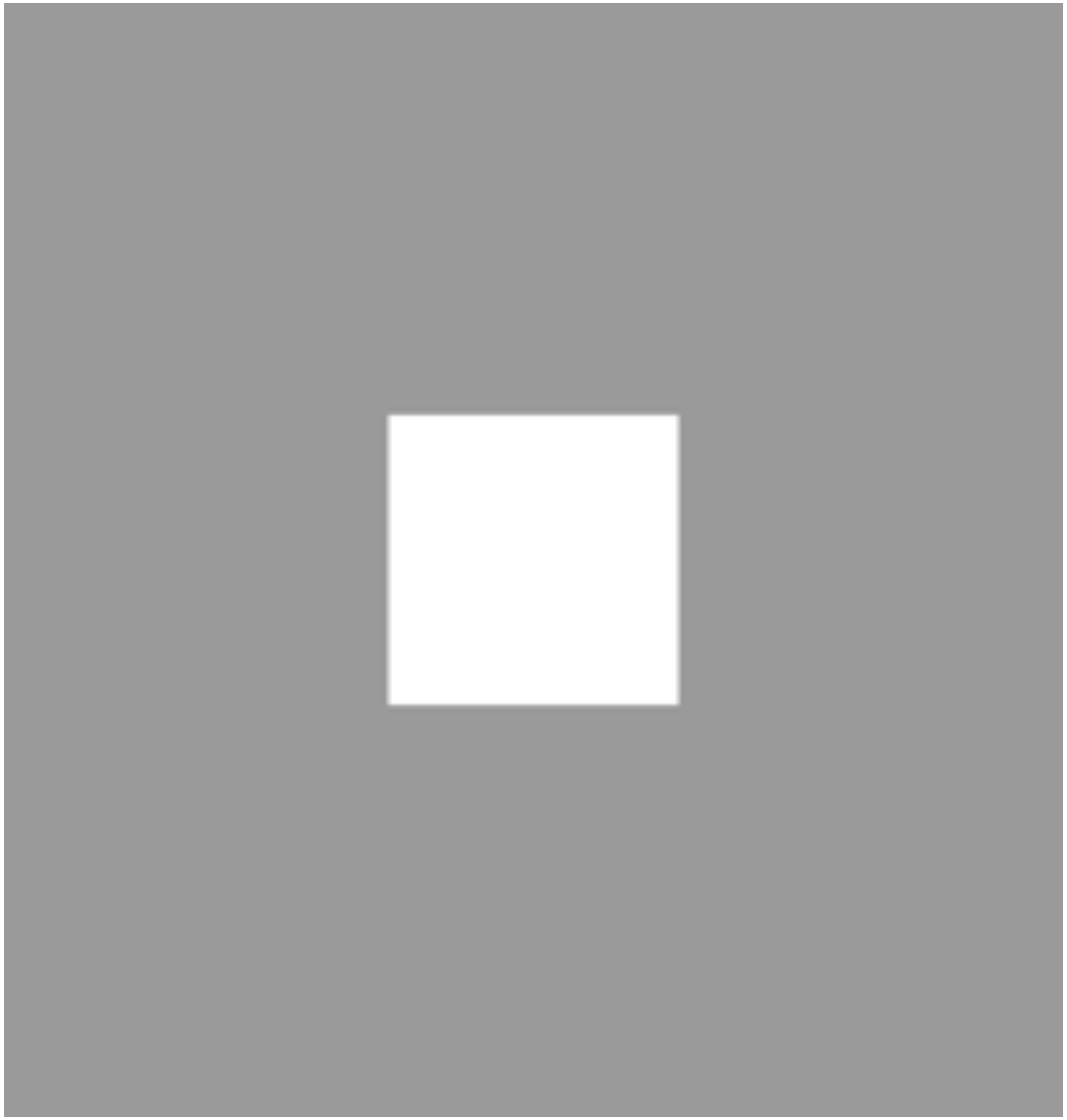
- Defined as **indexed arrays**
- Arrays are passed to GPU as **buffers**

```
var vertices = [-1, -1, -1, 1, -1, -1, 1, 1, -1, -1, 1, -1,
  -1, -1, 1, 1, -1, 1, 1, 1, 1, -1, 1, 1,
  -1, -1, -1, -1, 1, -1, -1, 1, 1, -1, -1, 1,
  1, -1, -1, 1, 1, -1, 1, 1, 1, 1, -1, 1,
  -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, -1, -1,
  -1, 1, -1, -1, 1, 1, 1, 1, 1, 1, 1, -1];

var colors = [5, 3, 7, 5, 3, 7, 5, 3, 7, 5, 3, 7,
  1, 1, 3, 1, 1, 3, 1, 1, 3, 1, 1, 3,
  0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
  1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
  1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
  0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0];

var indices = [0, 1, 2, 0, 2, 3, 4, 5, 6, 4, 6, 7,
  8, 9, 10, 8, 10, 11, 12, 13, 14, 12, 14, 15,
  16, 17, 18, 16, 18, 19, 20, 21, 22, 20, 22, 23];

var vertex_buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertex_buffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertices), gl.STATIC_DRAW);

var color_buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, color_buffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colors), gl.STATIC_DRAW);

var index_buffer = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, index_buffer);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint16Array(indices), gl.STATIC_DRAW);
```

GRAPHIC PIPELINE



A fragment shader computes a color for each pixel of the primitive being drawn, operations that may include texture mapping and lighting



```
var canvas = document.getElementById('canvas01');
gl = canvas.getContext('webgl');

var vertices = [-1, -1, -1, 1, -1, -1, 1, 1, -1, -1, 1, -1,
  -1, -1, 1, 1, -1, 1, 1, 1, 1, -1, 1, 1,
  -1, -1, -1, -1, 1, -1, -1, 1, 1, -1, -1, 1,
  1, -1, -1, 1, 1, -1, 1, 1, 1, 1, -1, 1,
  -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, -1, -1,
  -1, 1, -1, -1, 1, 1, 1, 1, 1, 1, 1, -1];

var colors = [5, 3, 7, 5, 3, 7, 5, 3, 7, 5, 3, 7,
  1, 1, 3, 1, 1, 3, 1, 1, 3, 1, 1, 3,
  0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
  1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,
  1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
  0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0];

var indices = [0, 1, 2, 0, 2, 3, 4, 5, 6, 4, 6, 7,
  8, 9, 10, 8, 10, 11, 12, 13, 14, 12, 14, 15,
  16, 17, 18, 16, 18, 19, 20, 21, 22, 20, 22, 23];

var vertex_buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertex_buffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertices), gl.STATIC_DRAW);

var color_buffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, color_buffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colors), gl.STATIC_DRAW);

var index_buffer = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, index_buffer);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint16Array(indices), gl.STATIC_DRAW);

var vertCode = 'attribute vec3 position;' +
  'uniform mat4 Pmatrix;' +
  'uniform mat4 Vmatrix;' +
  'uniform mat4 Mmatrix;' +
  'attribute vec3 color;' +
  'varying vec3 vColor;' +
  'void main(void) { ' +
  'gl_Position = Pmatrix*Vmatrix*Mmatrix*vec4(position, 1.);' +
  'vColor = color;' +
  '}';
```

WEBGL EXAMPLES

- Stardust:
  - Force-directed Graph with d3 and Stardust
  - Glyph-based visualization
  - Index chart with d3 and Stardust (same index chart with d3)
  - Bar charts with isotype
  - Daily activities of creative people
  - Bar charts with isotype
- SandDance
- NYT 3D yield curve 101
- CandyGraph
- Mapbox GL JS

OUTLINE

- Overview
- Canvas
- WebGL
- three.js
- Processing, Processing.js, P5.js
- Vtk.js
- Mapbox GL & Deck.gl

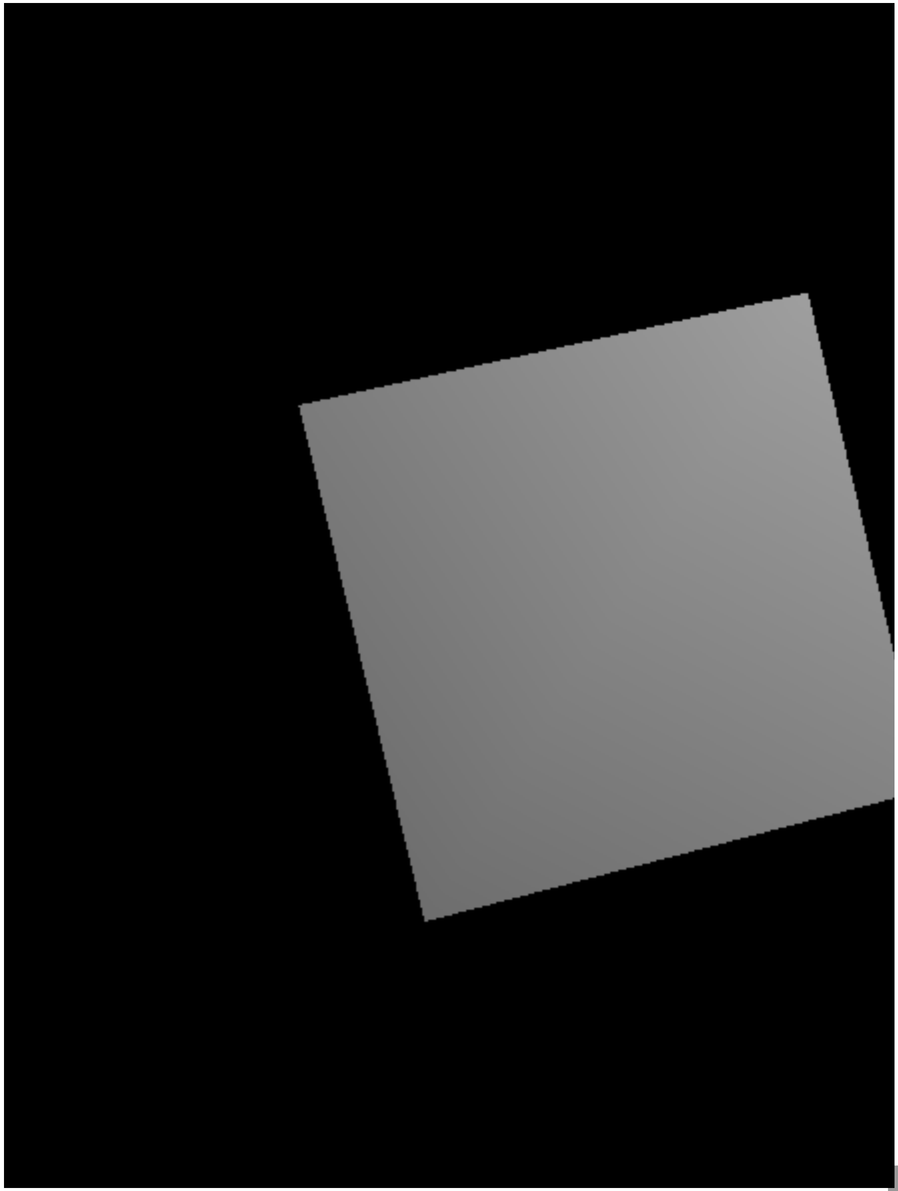
THREE.JS

High-level access to WebGL and graphical utilities:

- Scene
- Camera
- Geometry
- 3D Model Loaders
- Lights
- Materials
- Shaders
- Particles
- Animation
- Math Utilities

BASIC THREE.JS EXAMPLE

```
<canvas id="canvas" style="width: 600px; height: 600px;">

<script type="application/javascript">
  var scene = new THREE.Scene();
  var camera = new THREE.PerspectiveCamera(75, 1, 0.1, 1000);
  camera.position.set(0, 0, 2);  //camera.position.z = 2;
  //camera.lookAt(0, 0, 0);

  var renderer = new THREE.WebGLRenderer({ canvas: canvas });
  renderer.setSize(600, 600);

  var geometry = new THREE.BoxGeometry(1, 1, 1);
  var material = new THREE.MeshLambertMaterial({color: 0xFFFFFF});
  var cube = new THREE.Mesh(geometry, material);
  scene.add(cube);

  var light = new THREE.PointLight(0xFFFFFF);
  light.position.set(2, 2, 2);
  scene.add(light);

  var anim1 = () => {
    requestAnimationFrame(anim1)
    renderer.render(scene, camera)

    cube.rotation.x += Math.PI / 180
    cube.rotation.y += Math.PI / 180
    cube.rotation.z += Math.PI / 180
  }

  anim1();
</script>
```



THREE.JS EXAMPLES

- three.js Kinect skeleton player demo (dsci-554-kinect-tutorial)
- three.js gallery
- PhiloGL projects & examples

OUTLINE

- Overview
- Canvas
- WebGL
- three.js
- Processing, Processing.js, P5.js
- Vtk.js
- Mapbox GL & Deck.gl

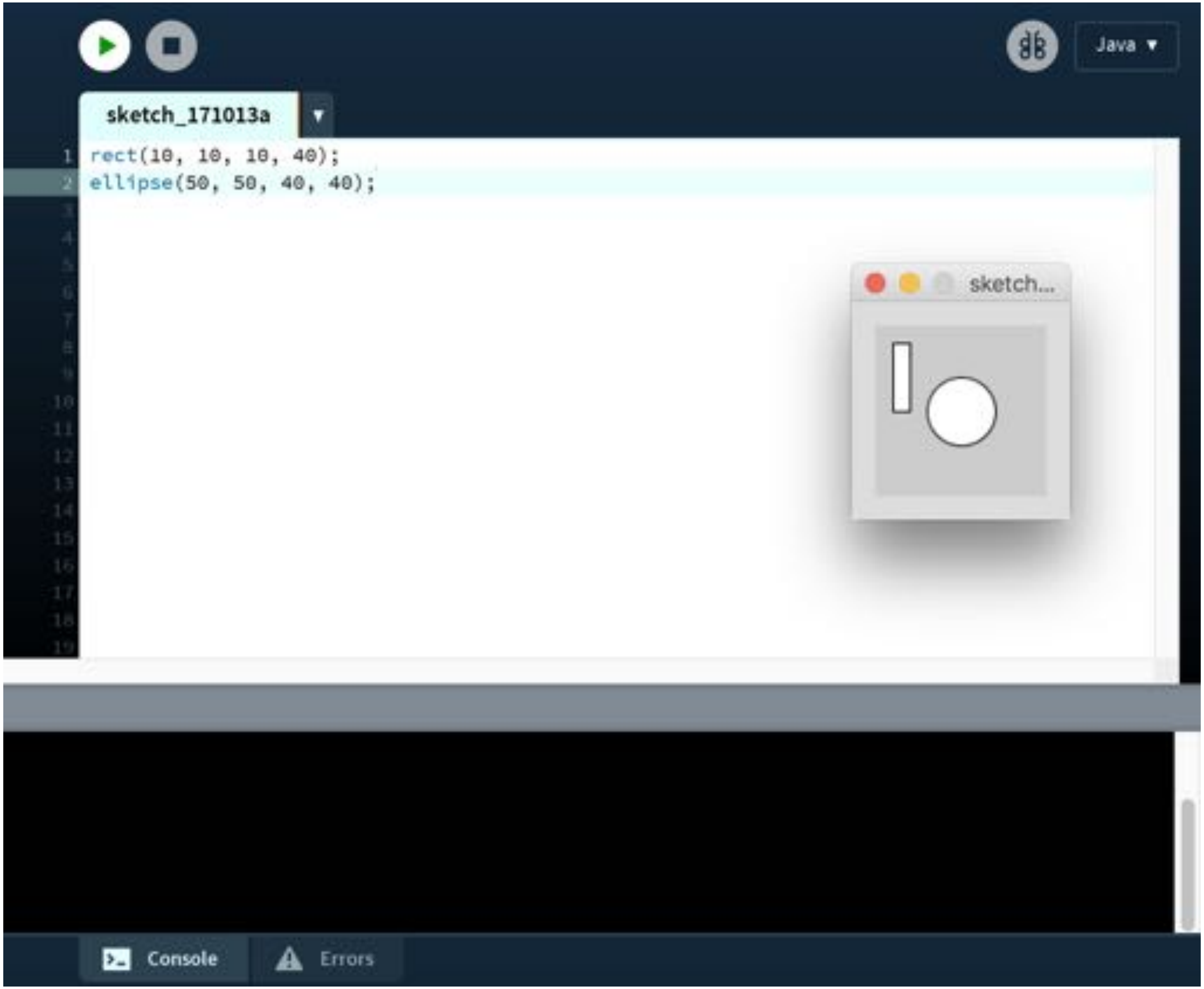
PROCESSING IMPLEMENTATIONS

- Sketchbook and language for learning to code targeted at visual arts
- All processing implementations support 2D and 3D <canvas> contexts

---

IMPLEMENTATIONS

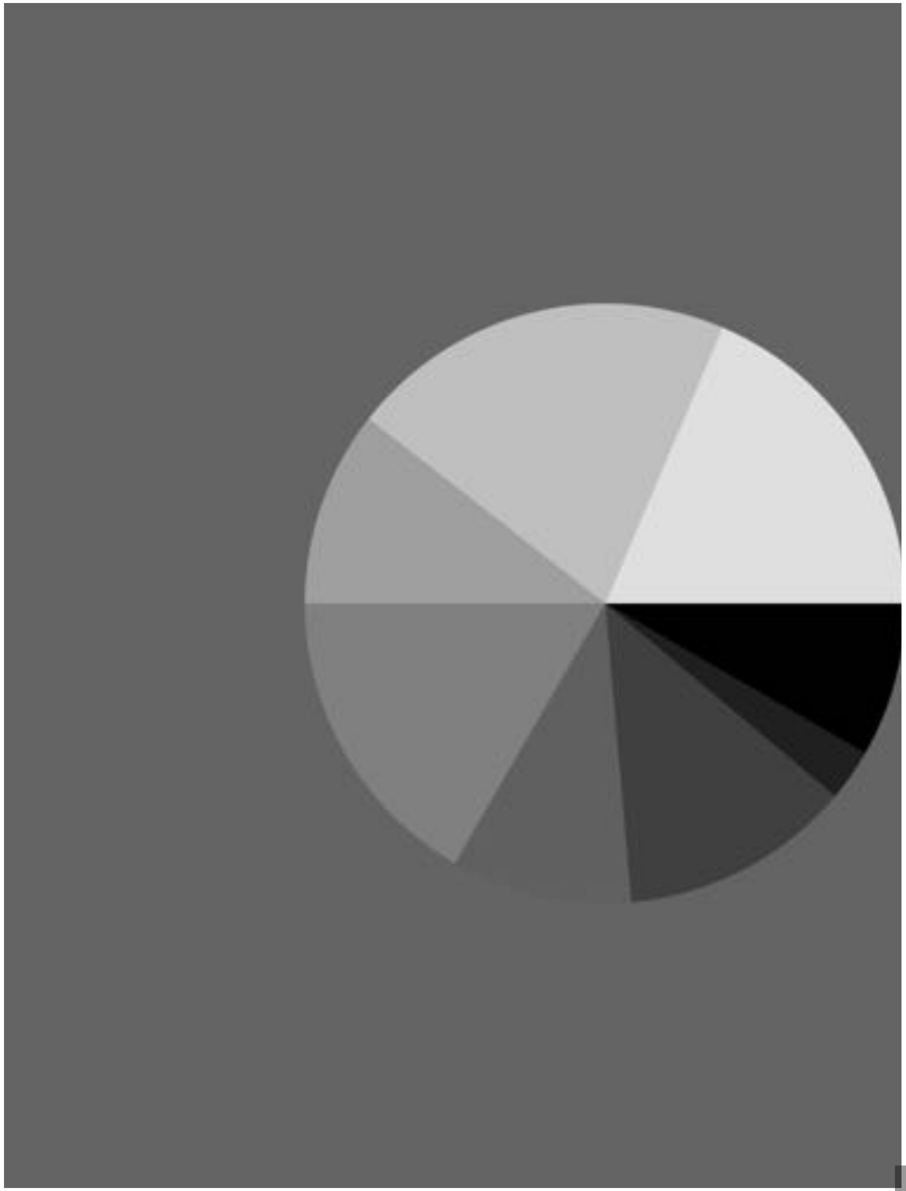
- Processing Simplified Java API for drawing and graphics [Fry & Reas 2001]
- Processing.js JS API to use Processing code [Resig 2008]
- P5.js HTML5 processing implementation Gallery [McCarthy 2015]



Processing sketch

BASIC P5.JS EXAMPLE

```
<script src="p5.min.js"></script>

<script>
int[] angles = { 30, 10, 45, 35, 60, 38, 75, 67 };

void setup() {
  size(640, 360);
  noStroke();
  noLoop();  // Run once and stop
}

void draw() {
  background(100);
  pieChart(300, angles);
}

void pieChart(float diameter, int[] data) {
  float lastAngle = 0;
  for (int i = 0; i < data.length; i++) {
  float gray = map(i, 0, data.length, 0, 255);
  fill(gray);
  arc(width/2,
    height/2,
    diameter,
    diameter,
    lastAngle,
    lastAngle+radians(data[i]));
  lastAngle += radians(data[i]);
  }
}
</script>
```



PROCESSING EXAMPLES

- Canvas, P5.js and circle packing (collision and cluster force) on a map on Observable
- P5.js examples

OUTLINE

- Overview
- Canvas
- WebGL
- three.js
- Processing, Processing.js, P5.js
- Vtk.js
- Mapbox GL & Deck.gl

VTK.JS

- VTK from Kitware is an example of a component architecture
- Adds a rendering abstraction layer over OpenGL
- Adds a rendering abstraction layer over OpenGL
- Targeted at medical imaging and engineering work
- vtk.js is a re-implementation of VTK/C++ in JavaScript

VTK.JS EXAMPLES

- ParaView Glance
- VTK.js: Scientific Visualization on the Web

OUTLINE

- Overview
- Canvas
- WebGL
- three.js
- Processing, Processing.js, P5.js
- Vtk.js
- Mapbox GL & Deck.gl

MAPBOX GL & DECK.GL

MAPBOX GL

- Open-source libraries for embedding customizable and responsive client-side maps in web
- You can use Mapbox GL JS to display Mapbox maps in a web browser

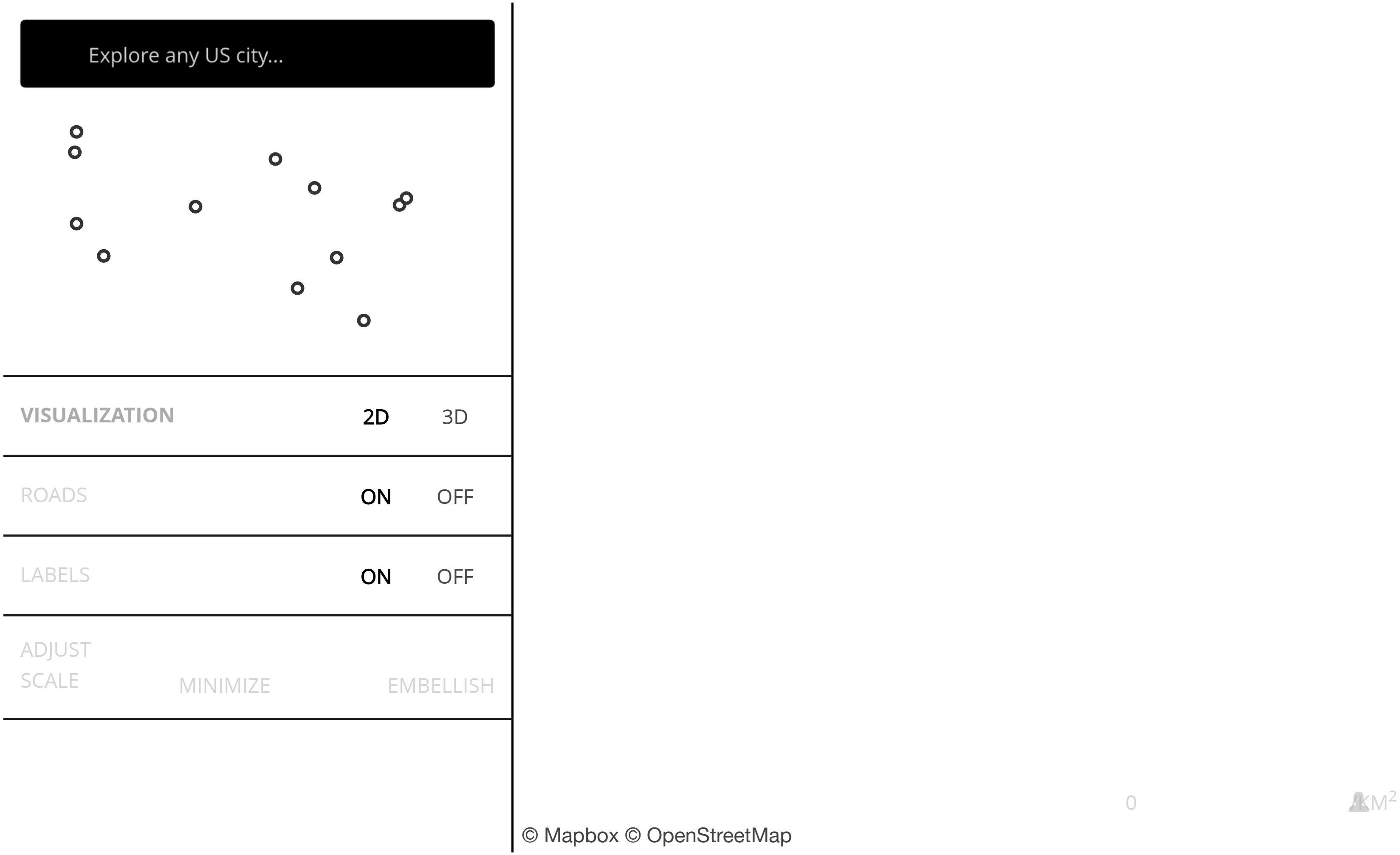
DECK.GL

- Simplify high-performance, WebGL-based visualization of large data sets
- deck.gl maps data (usually an array of JSON objects) into a stack of visual layers
- Cartographic projections and integration with major basemap providers including Mapbox, Google Maps and ESRI
- Part of the vis.gl framework suite

COMMONALITIES BETWEEN MAPBOX GL AND DECK.GL

- "GL" comes from OpenGL
- Interactive event handling such as picking, highlighting and filtering
- Renders at a high frame rate

MAPBOX GL JS EXAMPLE



Mapbox GL JS dive into large datasets with 3D shapes in Mapbox GL blog and full-screen demo

MAPBOX GL & DECK.GL EXAMPLES

- deck.gl + Mapbox GL in Vue app demo (mapbox-deckgl-example)
- deck.gl Examples
- deck.gl Showcase
- deck.gl Observable Getting Started

Questions?