Web Based Visualization

Jonas Lukasczyk

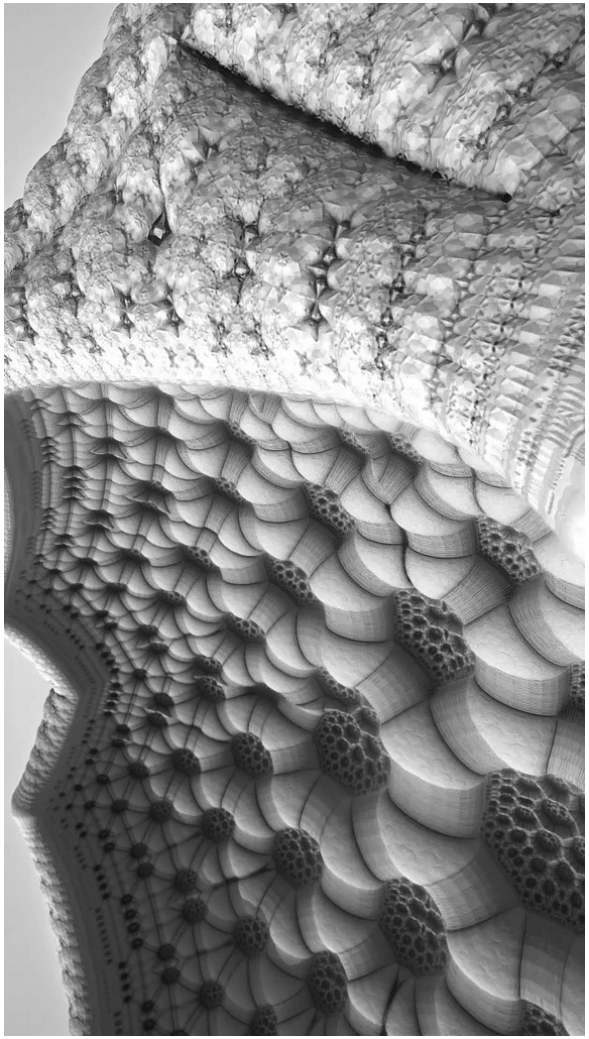
Technische Universität Kaiserslautern

Arizona State University

December 02, 2014

Web Based Visualization

Table of Contents



http://fractal.io/

Introduction

Demos

- WebGL Globe
  (`http://workshop.chromeexperiments.com/globe/`)
- Arms Globe
  (`http://workshop.chromeexperiments.com/projects/armsglobe/`)
- Fluid Simulation
  (`http://haxiomic.github.io/projects/webgl-fluid-and-particles/`)
- Zygote Body
  (`http://www.zygotebody.com/`)
- Artificial Neural Network
  (`http://nxxcxx.github.io/Neural-Network/`)

Advantages

- Accessibility
- Cross-Platform Compatibility

Disadvantages

- Unsuitable for large Projects
  - Development
  - Maintenance
  - Performance
- Limitations
  - No Tessellation and Geometry Shaders
  - No 3D Textures
  - ...

Development

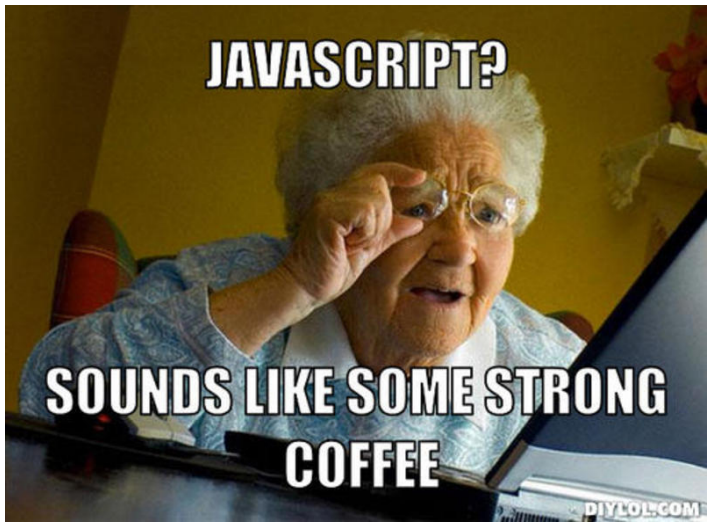
JavaScript

Advantages

- The only standardized language in all web browsers
- Modern JavaScript VMs perform fairly well

Disadvantages

- Dynamic Type System
- Garbage Collector
- Performance
- Development and Maintenance



```
1 var x;          // console.log(x); -> "undefined"
2 x=x+"X";        // console.log(x); -> "undefinedX"
```

Objects

```
1 var y={1:x};
2 // ...log(y["1"] + " " + y[1]); -> "undefinedX undefinedX"
3
4 y.z=4;          // console.log(y.z); -> 4
5 y["z"]="Bob";//...log(y);->Object{1:"undefinedX", z:"Bob"}
```

Method Scope

```
1 y.func = function(){return 2;} // ...log( y.func() ); -> 2
2 var q = y.func; // console.log( q() ); -> 2
```

Are there alternatives to JavaScript?

- Google Native Client (NaCl) and Portable Native Client (PNaCl), but still not standardized
- Idea: Compile C/C++ to JavaScript Code

$$\text{C/C++} \quad \Longrightarrow \quad \text{LLVM} \overset{emscripten}{\Longrightarrow} \text{JavaScript (ams.js)}$$

emscripten

- LLVM-to-JavaScript Compiler

asm.js

- Started as research project at Mozilla
- Low-Level Subset of JavaScript
- Optimized for Performance

C++ Code

```
1 int f(int i){
2     return i+1;
3 }
```

JavaScript Code (ams.js)

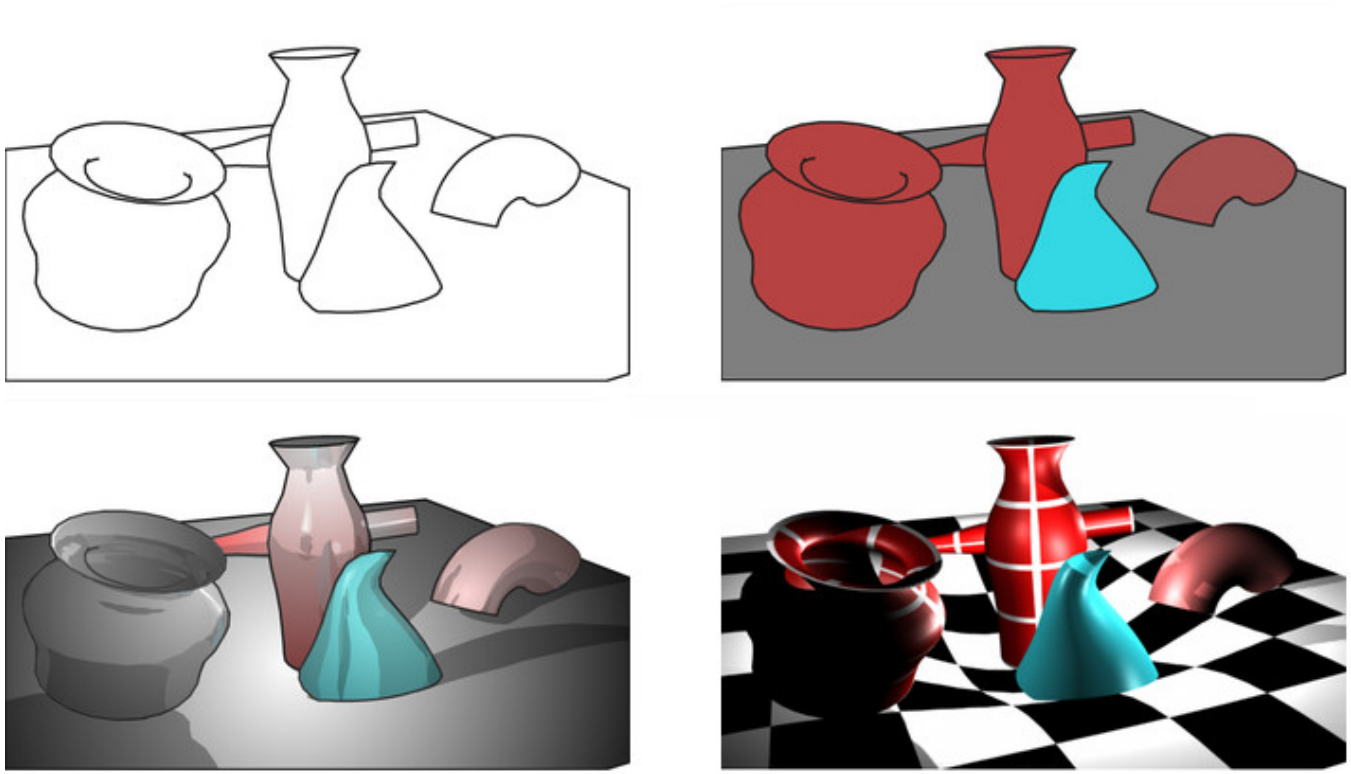
```
1 function f(i){
2     i = i|0;
3     return (i+1)|0;
4 }
```

Frameworks

- jQuery (HTML document manipulation)
  (`http://jquery.com/`)
- DataTables
  (`http://www.datatables.net/`)
- D3 (Graphics Library - Mostly HTML, SVG, and CSS based)
  (`http://d3js.org/`)
- THREE.js (WebGL Library)
  (`http://threejs.org/`)

Rendering with THREE.js

A 3D world can be visualized with a **Scene**, a **Camera**, and a **Renderer**.



The Scene

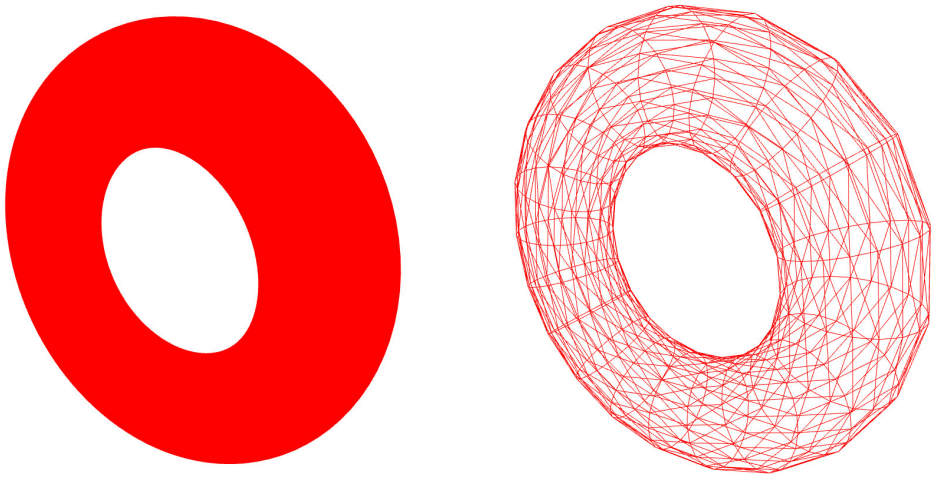
```
1 var scene = new THREE.Scene();
2
3 var tGeometry= new THREE.TorusGeometry(18, 6, 100, 100);
4 var tMaterial= new THREE.MeshBasicMaterial({color:'red'});
5 var torus = new THREE.Mesh(tGeometry, tMaterial);
6
7 scene.add(torus);
```

The Camera

```
1 var camera = new THREE.PerspectiveCamera(60, 600/800, 1,
      1000);
2 camera.position.set(-40, 40, -50);
```

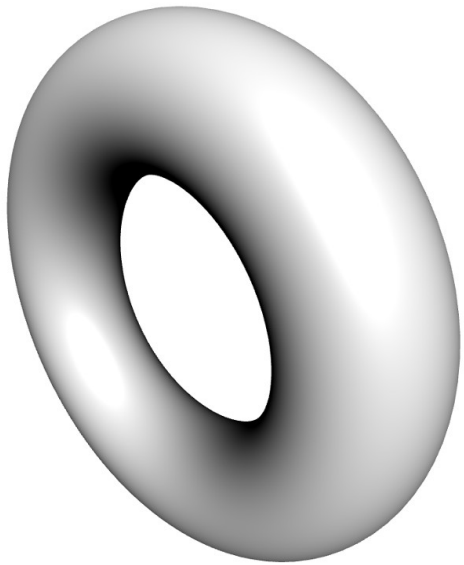
The Renderer

```
1 var renderer = new THREE.WebGLRenderer({antialias:true});
2 renderer.setSize( 800, 600 );
3 renderer.setClearColor( 0xFFFFFF, 1 );
4 var container = document.getElementById('container');
5 container.appendChild( renderer.domElement );
6 renderer.render( scene, camera );
```

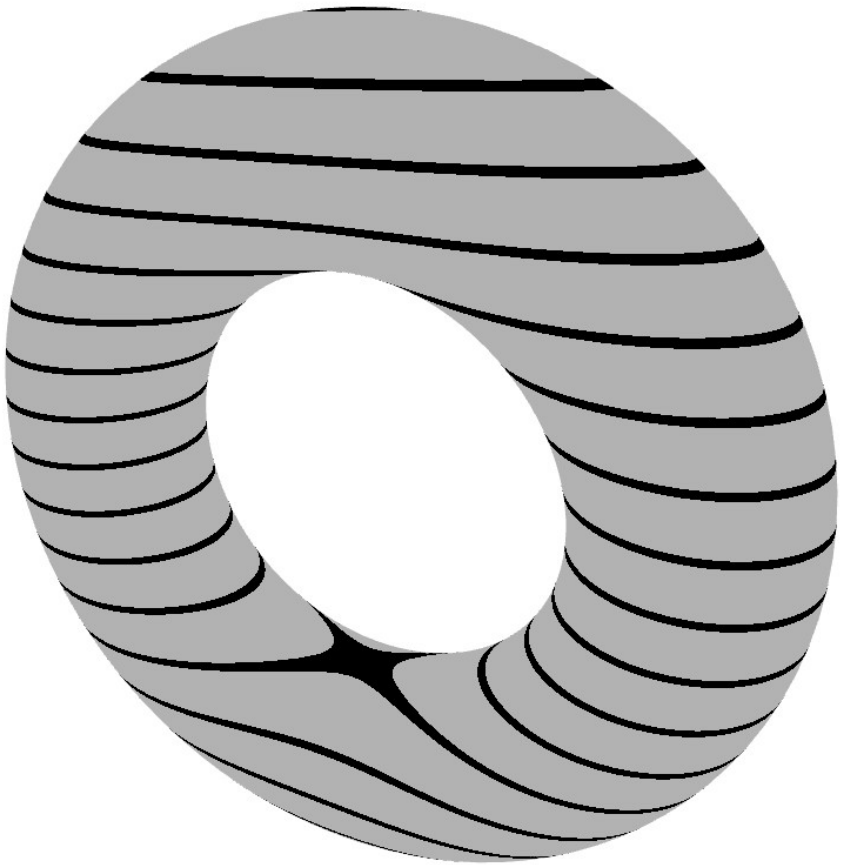


Advanced Shading

```
1 var tMaterial = new THREE.MeshPhongMaterial();
2
3 var dLight = new THREE.DirectionalLight(0xffffff);
4 dLight.position.set(1, 1, 1).normalize();
5 scene.add(dLight);
```



Custom Shading (Example: Iso-Lines of Surface)



Step 1: Set uv-coordinates

```
1  tGeometry.faceVertexUvs[0]=[];
2  for (i = 0; i < tGeometry.faces.length ; i++) {
3      var v1 = tGeometry.vertices[ tGeometry.faces[i].a ];
4      var v2 = tGeometry.vertices[ tGeometry.faces[i].b ];
5      var v3 = tGeometry.vertices[ tGeometry.faces[i].c ];
6      tGeometry.faceVertexUvs[0].push([
7          new THREE.Vector2(v1.y, 0),
8          new THREE.Vector2(v2.y, 0),
9          new THREE.Vector2(v3.y, 0)
10     ]);
11 }
12 tGeometry.uvsNeedUpdate = true;
```

Step 2: Write Vertex-Shader (NOT JavaScript!)

```
1 <script type="x-shader/x-vertex" id="vertexShader">
2     varying vec2 vUv;
3
4     void main(){
5         vUv = uv;
6         gl_Position = projectionMatrix * modelViewMatrix *
              vec4(position, 1.0);
7     }
8 </script>
```

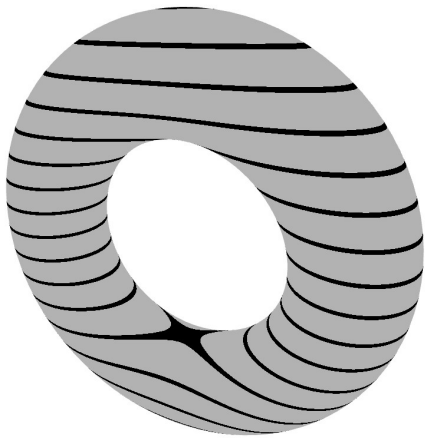
Step 3: Write Fragment-Shader

```
<script type="x-shader/x-fragment" id="fragmentShader">
    varying vec2 vUv;

    void main(){
        float gp = floor(vUv[0]/3.+0.5);
        if( abs( vUv[0] - gp*3. ) < 0.2 ){
            gl FragColor = vec4(0., 0., 0., 1.);
        } else {
            gl FragColor = vec4(0.7, 0.7, 0.7, 1.);
        }
    }
</script>
```

Step 4: Create Shader Material

```
1 var torusMaterial = new THREE.ShaderMaterial({
2   vertexShader:
        document.getElementById('vertexShader').textContent,
3   fragmentShader:
        document.getElementById('fragmentShader').textContent
4 });
```



Mouse Interaction

```
1  var controls = new THREE.OrbitControls(
2      camera,
3      renderer.domElement
4  );
5
6  controls.addEventListener(
7      'change',
8      function(){
9          renderer.render( scene, camera );
10     }
11 );
12
13 controls.update();
```

Animation

```
1  var clock = new THREE.Clock();
2
3  function animate(){
4      requestAnimationFrame( animate );
5      renderer.render( scene, camera );
6      update();
7  }
8
9  function update(){
10     var time = clock.getElapsedTime();
11     torus.position.x = Math.sin( time ) * 10;
12     torus.position.y = Math.cos( time ) * 10;
13 }
```

Particle Systems I (Example: Snow on CPU)

Step 1: Setup Particle System

```
1  var psM = new THREE.ParticleBasicMaterial(color:0xFFFFFF);
2
3  var psG = new THREE.Geometry();
4  for( var i = 0; i < 100; i++ ){
5      var vertex = new THREE.Vector3(
6          Math.random()*100-50,
7          Math.random()*100-50,
8          Math.random()*100-50
9      );
10     psG.vertices.push( vertex );
11 }
12
13 var particleSystem = new THREE.ParticleSystem( psG, psM );
14 scene.add( particleSystem );
```

Step 2: Animate Particle System

```
1  function update(){
2      var geometry = particleSystem.geometry,
3      for(var i = 0; i < geometry.vertices.length; i++){
4          var v = geometry.vertices[i];
5          if( v.y > -psHeight/2 )
6              v.y -= 1;
7          else
8              v.y = psHeight/2;
9      }
10     particleSystem.rotation.y += 0.008;
11     geometry.verticesNeedUpdate = true;
12 }
```

Particle Systems II (Example: Snow on GPU)

Step 1: Define Material for Particle System

```
1  var psM = new THREE.ShaderMaterial({
2      uniforms: {
3          height: { type: 'f', value: psHeight },
4          elapsedTime: { type: 'f', value: 0 }
5      },
6      vertexShader:
7       document.getElementById('vertexShader').textContent,
8      fragmentShader:
9       document.getElementById('fragmentShader').textContent
10 });
```

Step 2: Write Vertex-Shader

```
1  <script type="x-shader/x-vertex" id="vertexShader">
2  uniform float height;
3  uniform float elapsedTime;
4
5  void main(){
6      vec3 pos = position;
7      pos.y = mod(pos.y - elapsedTime, height);
8      gl_Position = projectionMatrix * modelViewMatrix *
           vec4(pos, 1.);
9  }
10 </script>
```

Step 3: Write Fragment-Shader

```
<script type="x-shader/x-fragment" id="fragmentShader">
void main(){
    gl_FragColor = vec4(1., 1., 1., 1.);
}
</script>
```

Step 4: Define Update Function

```
function update(){
    var t = clock.getElapsedTime();
    particleSystem.material.uniforms.elapsedTime.value =
        t*10;
}
```

Resources

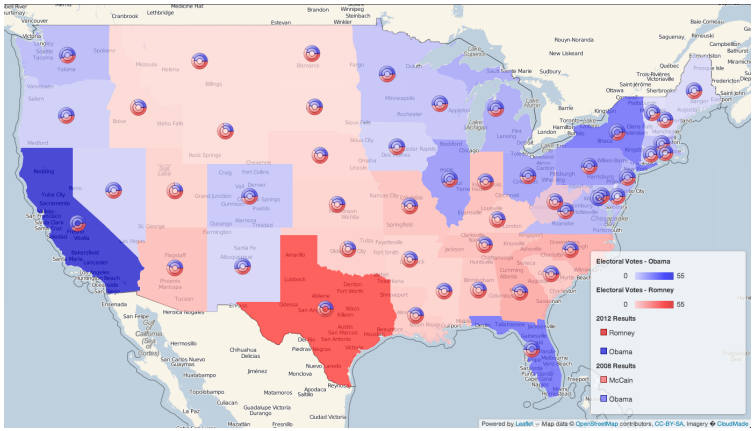
- http://threejs.org/examples/
- http://aerotwist.com/tutorials/getting-started-with-three-js/
- https://stemkoski.github.io/Three.js/
- http://www.smartjava.org/content/all-109-examples-my-book-threejs-threejs-version-r63

Geovisualization

Geovisualization

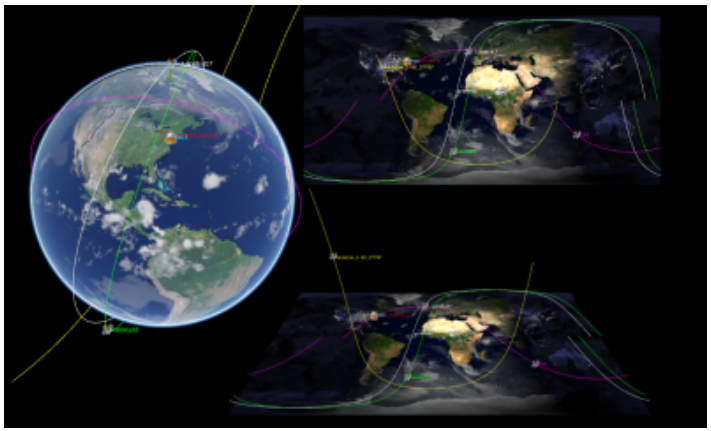
Frameworks

(a) Google Maps



(b) Leaflet

(a) Google Earth



(b) Cesium

Geovisualization

Frameworks

a. Google Maps
   - \+ King of 2D Geovisualizations
   - – 3D capability only via Flash and Android API
   - – Feels Outdated

b. Leaflet
   - \+ Lighter than Google-Maps
   - \+ Better API
   - – No 3D capability
   - – Maps not customizable as with Google-Maps

c. Google Earth
   - \+ 3D capability
   - – Requires additional PlugIn

d. Cesium (`http://cesiumjs.org/`)
   - \+ King of 3D Geovisualizations
   - – Requires time-consuming modifications
   - – Performance

WebGL-based Geodata Visualization for Policy Support and Decision Making

Motivating Project: Energy Consumption Data in the Area of Phoenix

Data Set: Scalar Values for Travel Analysis Zones (TAZ):

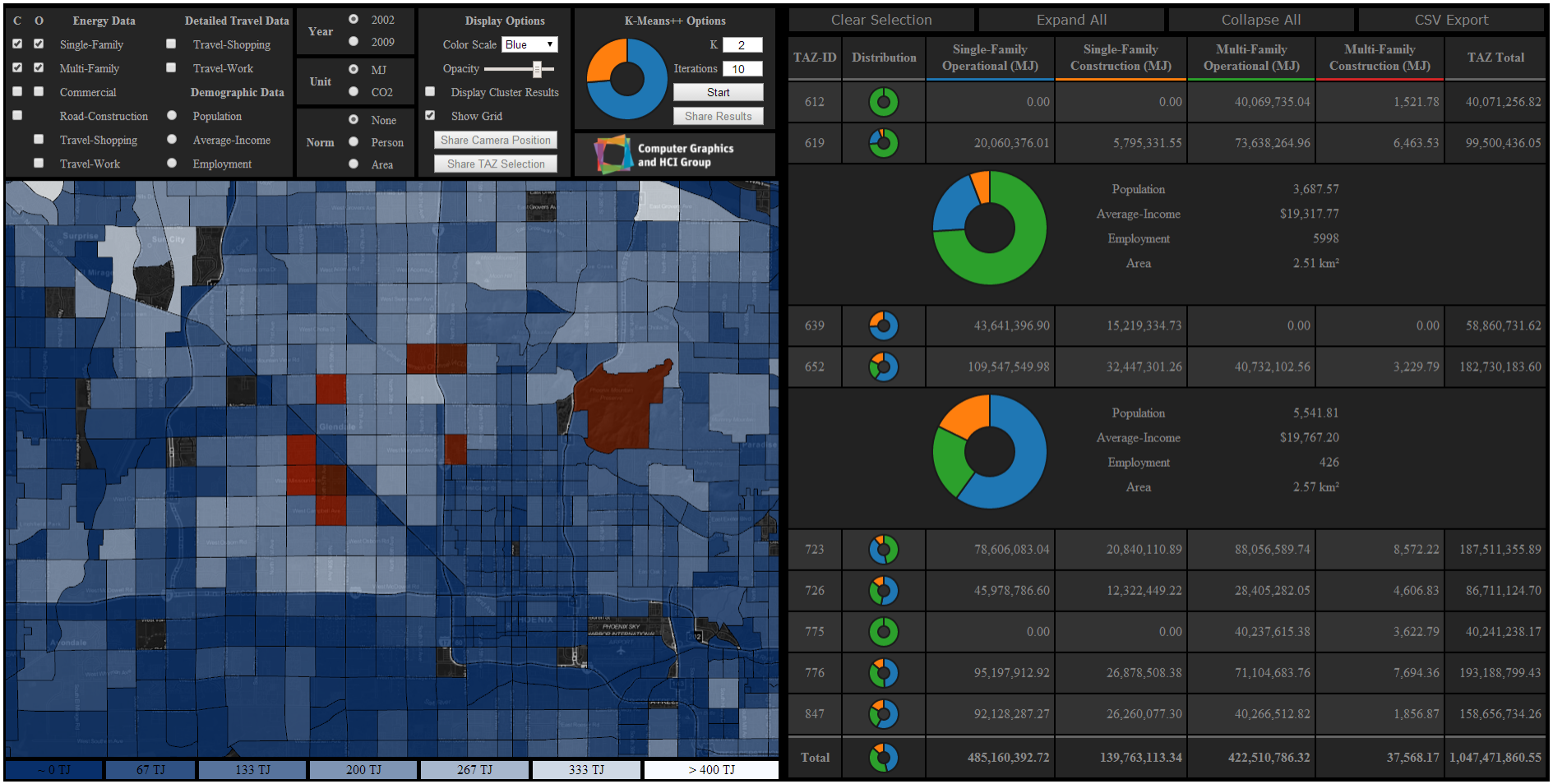
- Operational and Constructional Energy for
  - Singe-Families
  - Multi-Families
  - Commercial Buildings
- Road-Construction
- Travel Energy

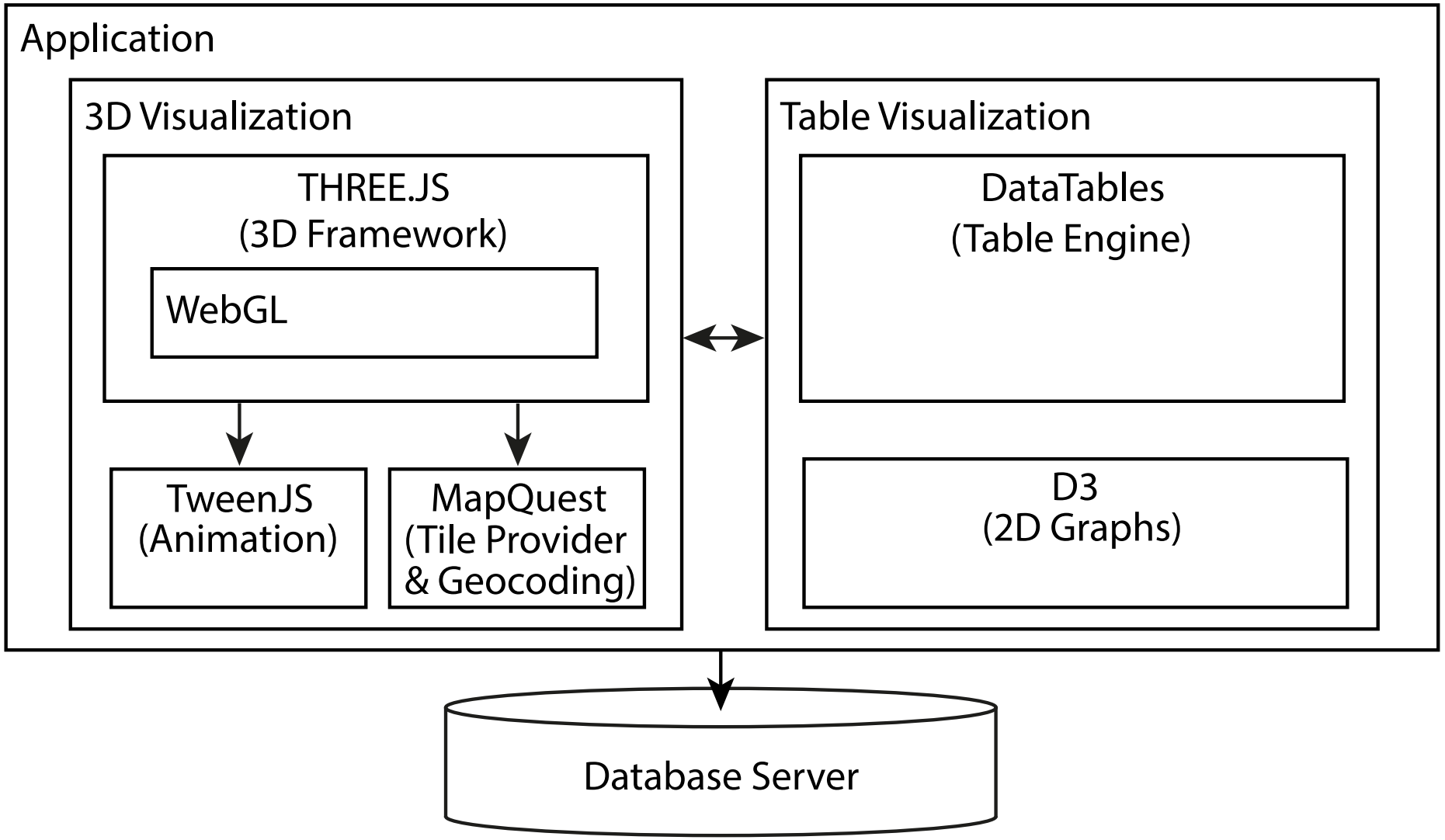
EnergyVis 1.0

(www.jluk.de/projects/WebGL/EnergyVis/release.html)

EnergyVis 2.0

(www.jluk.de/projects/WebGL/EnergyVis2/release.html)




Application
3D Visualization
THREE.JS
(3D Framework)
WebGL
TweenJS
(Animation)
MapQuest
(Tile Provider
& Geocoding)
Table Visualization
DataTables
(Table Engine)
D3
(2D Graphs)
Database Server

Decision Theater

Satellite Center

Multi-User Interaction

Node.js (Server)

- based on Chrome's JavaScript runtime
- platform for fast, scalable network applications
- very easy to setup

Socket.io (Communication)

- features real-time bidirectional event-based communication
- works on almost every platform, browser and device

Excellent Tutorial: `http://socket.io/get-started/chat/`

Server Code (index.js)

```
1  var app = require('express')();
2  var http = require('http').Server(app);
3  var io = require('socket.io')(http);
4
5  app.get('/', function(req, res){
6    res.sendfile('index.html');
7  });
8
9  io.on('connection', function(socket){
10   socket.on('chat message', function(msg){
11     io.emit('chat message', msg);
12   });
13 });
14
15 http.listen(1337, function(){
16   console.log('listening on *:1337');
17 });
```

Launch Server

```
1 $ node index.js
```

Client Code (index.html)

```
1  ...
2  <ul id="messages"></ul>
3  <form action="">
4      <input id="m" /><button>Send</button>
5  </form>
6  ...
7  <script>
8      var socket = io();
9      $('form').submit(function(){
10         socket.emit('chat message', $('#m').val());
11         $('#m').val('');
12         return false;
13     });
14     socket.on('chat message', function(msg){
15         $('#messages').append($('<li>').text(msg));
16     });
17 </script>
```

Model-Viewer

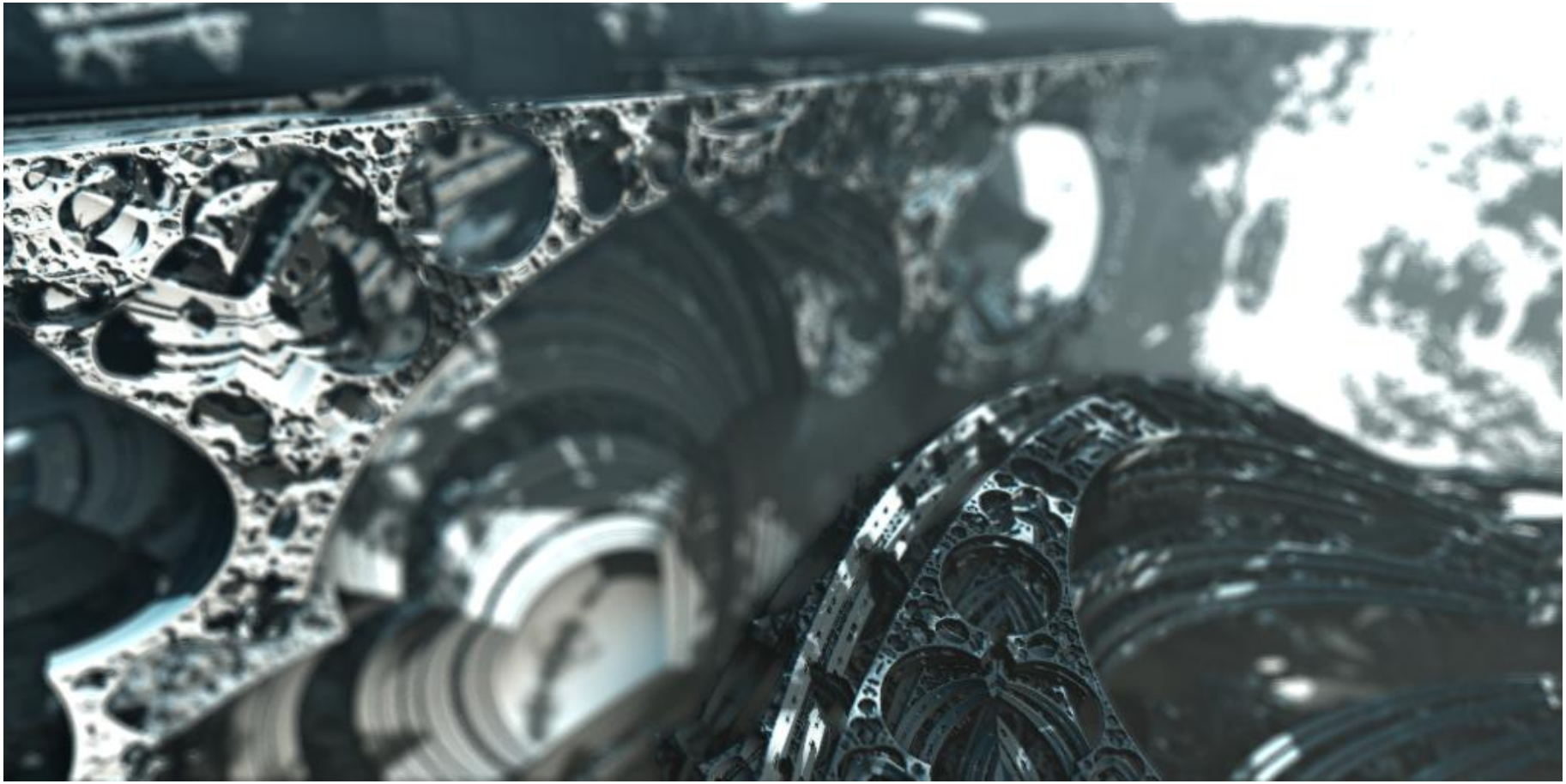
(`www.jluk.de/projects/WebGL/ModelViewer/demo.html`)

Conclusion

Cornerstones

- Development
  - Pros and Cons of JavaScript
  - C++ $\rightarrow$ JavaScript
- Frameworks
  - Three.js
  - D3
  - Google-Maps (and alike)
- Multi-User Interaction
  - Node.js
  - Socket.io

Thank You



http://projekte.sinnpirat.de/fraktal/