Einführung in WebGL

3D Grafik im Web Browser

Einführung in WebGL

- **Mit HTML5 kam ein neues Element <canvas> hinzu.**
- **Innerhalb eines Canvas können mit Zeichenbefehlen 2D Graphiken dynamisch erstellt werden**
- **Ähnlich zu Canvas Elementen in GUI Libraries z. B. von Java**

Einführung in WebGL

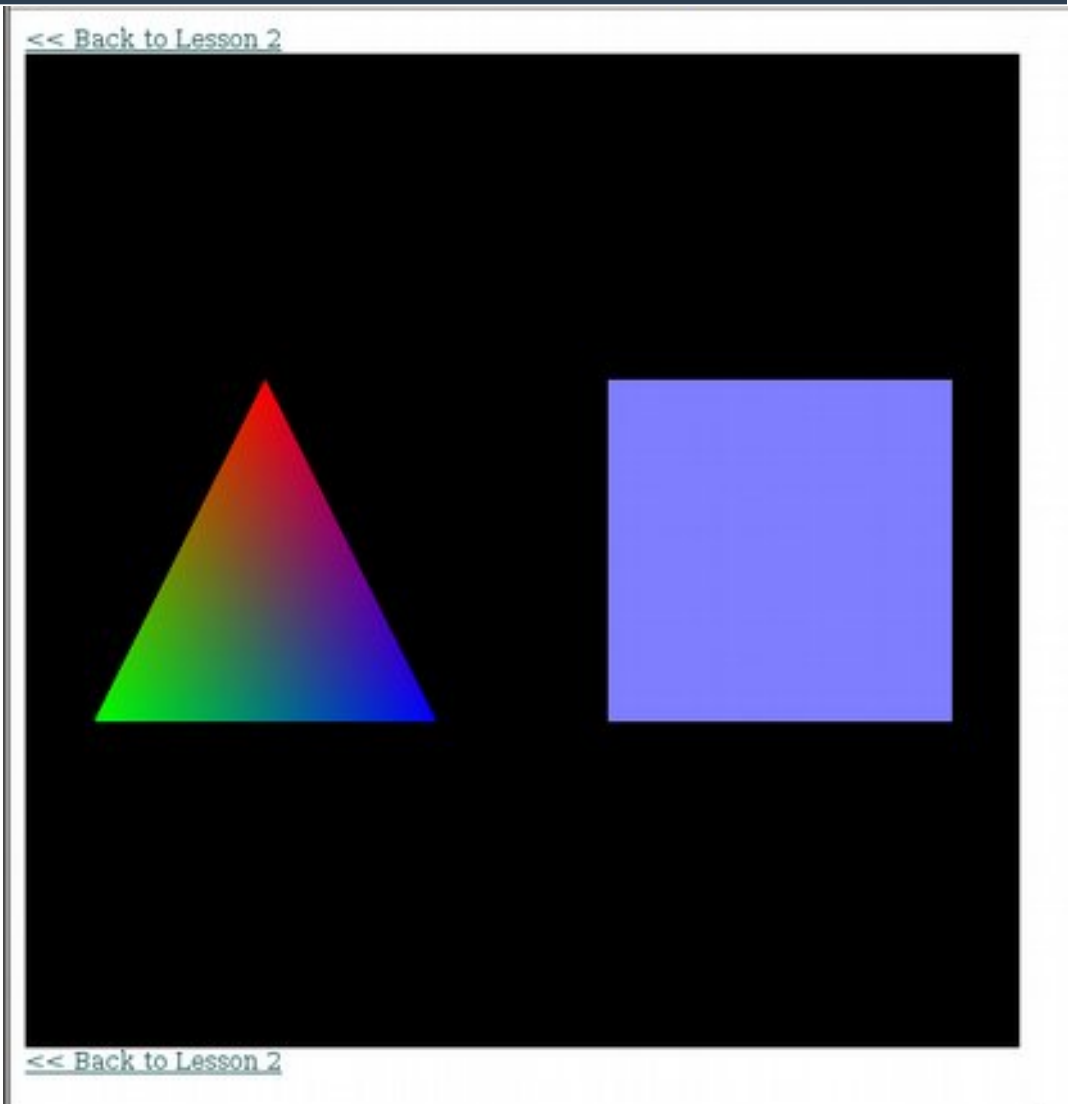
- **WebGL erweitert diesen 2D Canvas zu einem 3D OpenGL Zeichenbereich**
- **Erlaubt das rendern gemäss**
- **OpenGL ES 2.0 API und**
- **OpenGL ES Shading Language, 1.00**
- **Rendern mit OpenGL ES 2.0 erzwingt den gebrauch von Shadern. Keine Fixed-Function-Pipeline mehr!**

1. Beispiel: Kontext kreieren

```
<!DOCTYPE html>
<html><body>
  <canvas id=”c”></canvas>
  <script type=”text/javascript”>
    var canvas = document.getElementById(“c”);
    var gl = canvas.getContext(“webgl”);
    gl.clearColor(1.0, 0.0, 0.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);
  </script>
</body></html>
```

2. Beispiel: Nur WebGL inkl. Shader

- **Datei: webGL-only/learningwebgl-lesson02.html**
- **Beispiel nur mit WebGL eigenen Mitteln ohne Zuhilfenahme von Libraries**



What is Three.js?

Let's try to describe it briefly:

Three.js is a library that makes WebGL - 3D in the browser - easy to use.

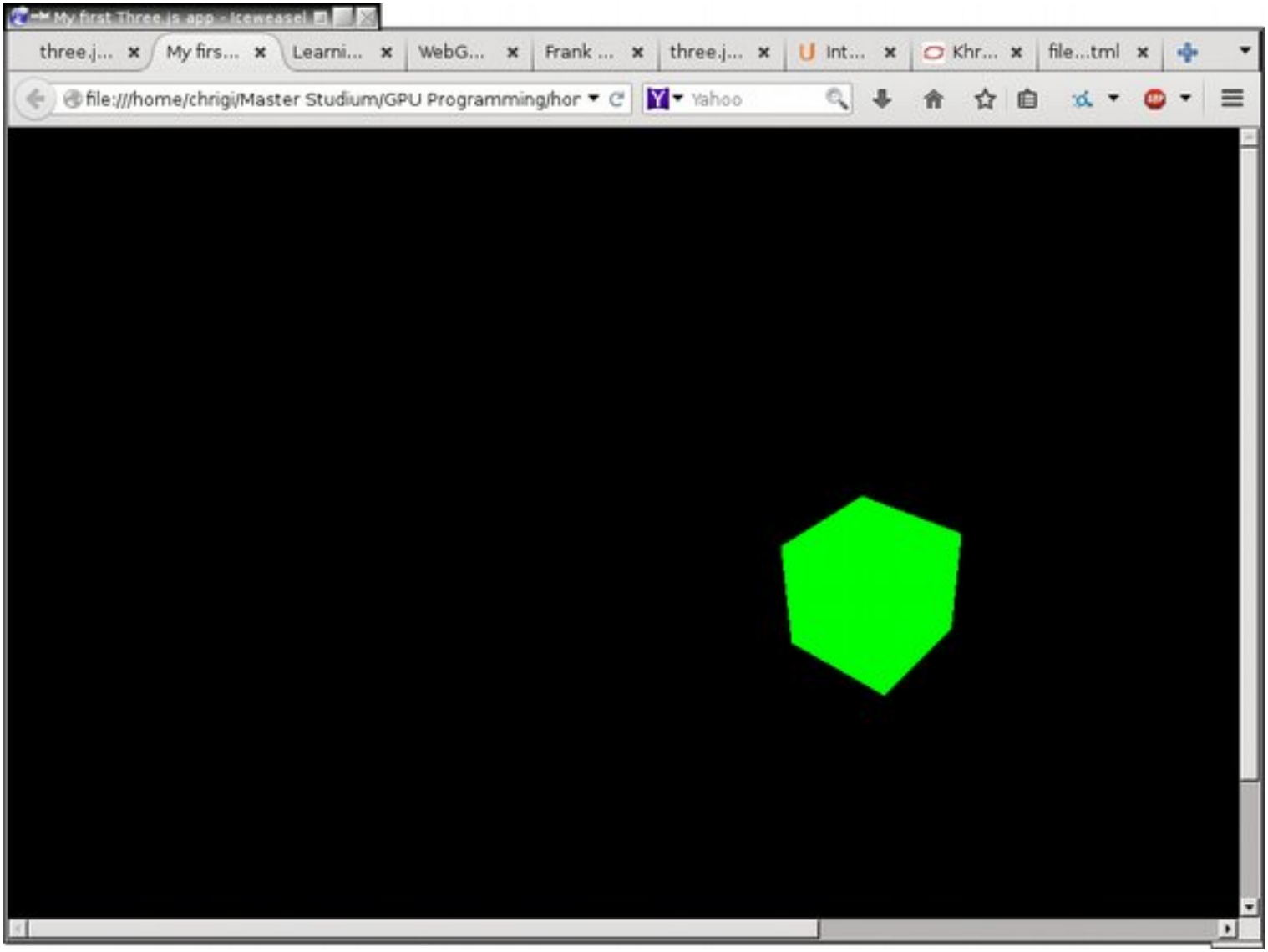
While a simple cube in raw WebGL would turn out hundreds of lines of Javascript and shader code, a Three.js equivalent is only a fraction of that.

Einfaches three.js Beispiel

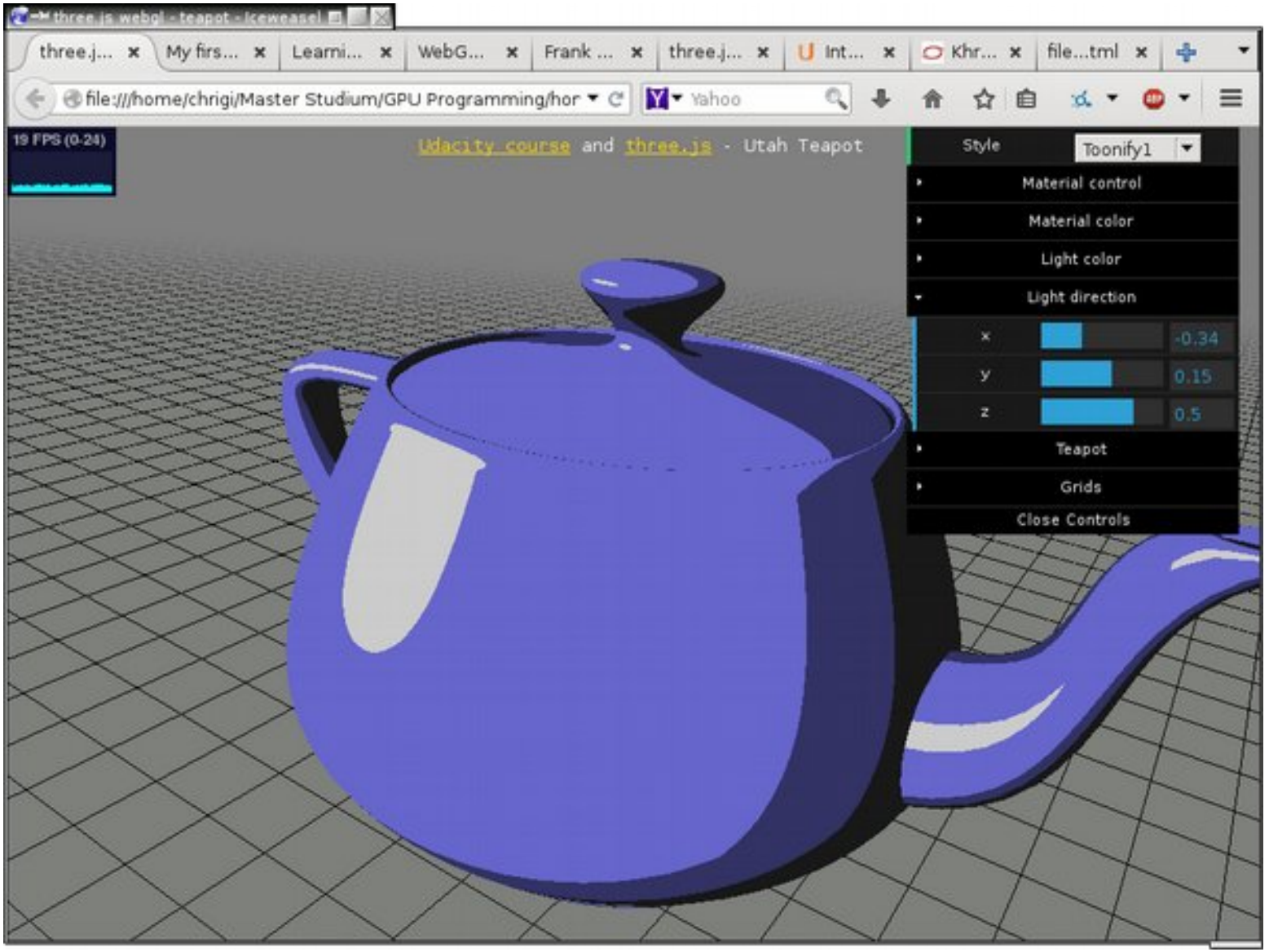
```
<html>
  <head>
    <title>My first Three.js app</title>
    <style>
      body { margin: 0; }
      canvas { width: 100%; height: 100% }
    </style>
  </head>
  <body>
    <script src="three.min.js"></script>
    <script>
      var scene = new THREE.Scene();
      var camera = new THREE.PerspectiveCamera( 75, window.innerWidth/window.innerHeight, 0.1, 1000 );
      var renderer = new THREE.WebGLRenderer(); renderer.setSize( window.innerWidth, window.innerHeight );
      document.body.appendChild( renderer.domElement );
      var geometry = new THREE.BoxGeometry( 1, 1, 1 );
      var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
      var cube = new THREE.Mesh( geometry, material ); scene.add( cube ); camera.position.z = 5;

      var render = function () {
        requestAnimationFrame( render );
        cube.rotation.x += 0.03; cube.rotation.y += 0.03;
        renderer.render(scene, camera);
      };
      render();
    </script>
  </body>
</html>
```

Ergebnis: Rotierender Würfel



Teapot Spielwiese des Udacity Kurs

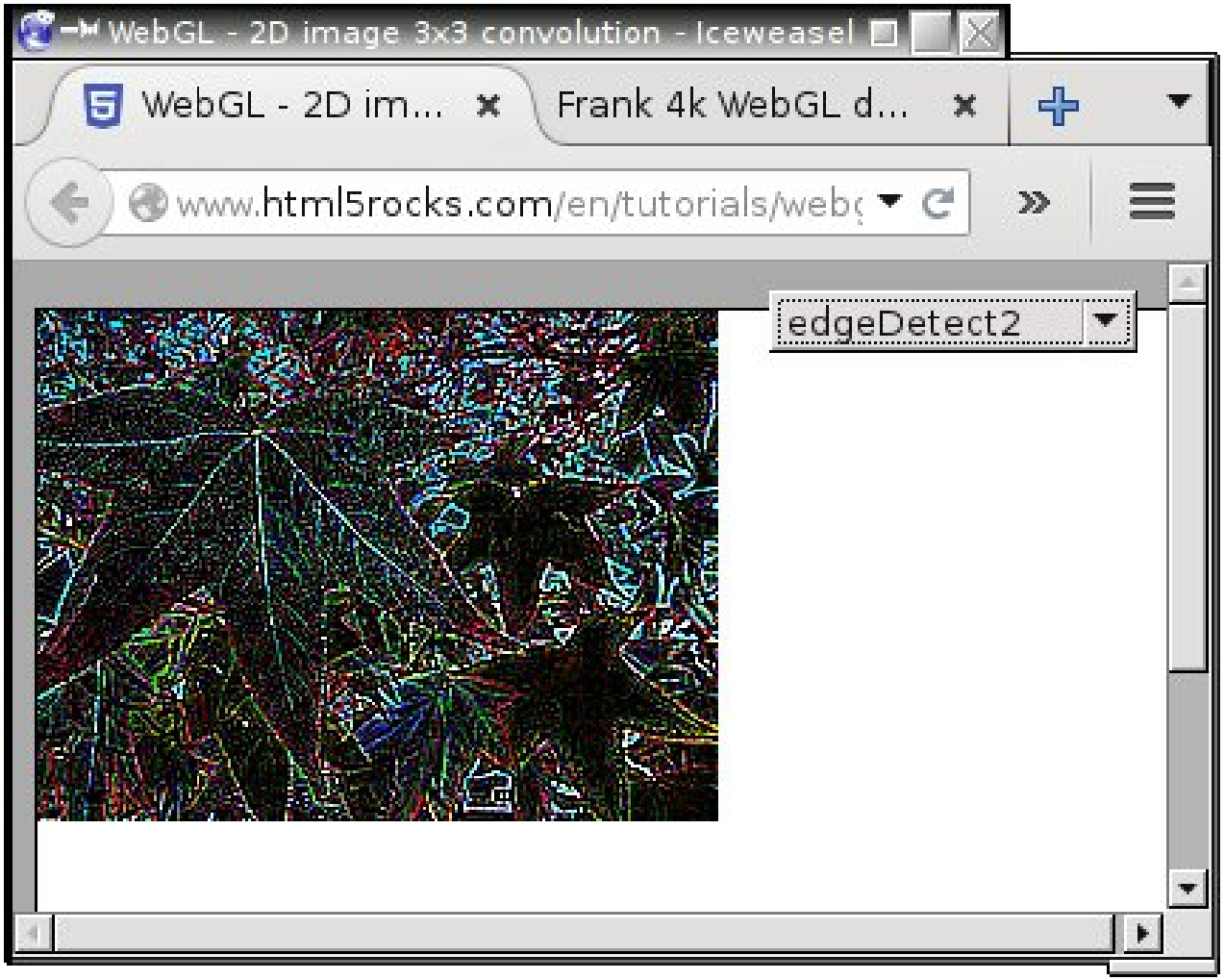


Wie weiter?

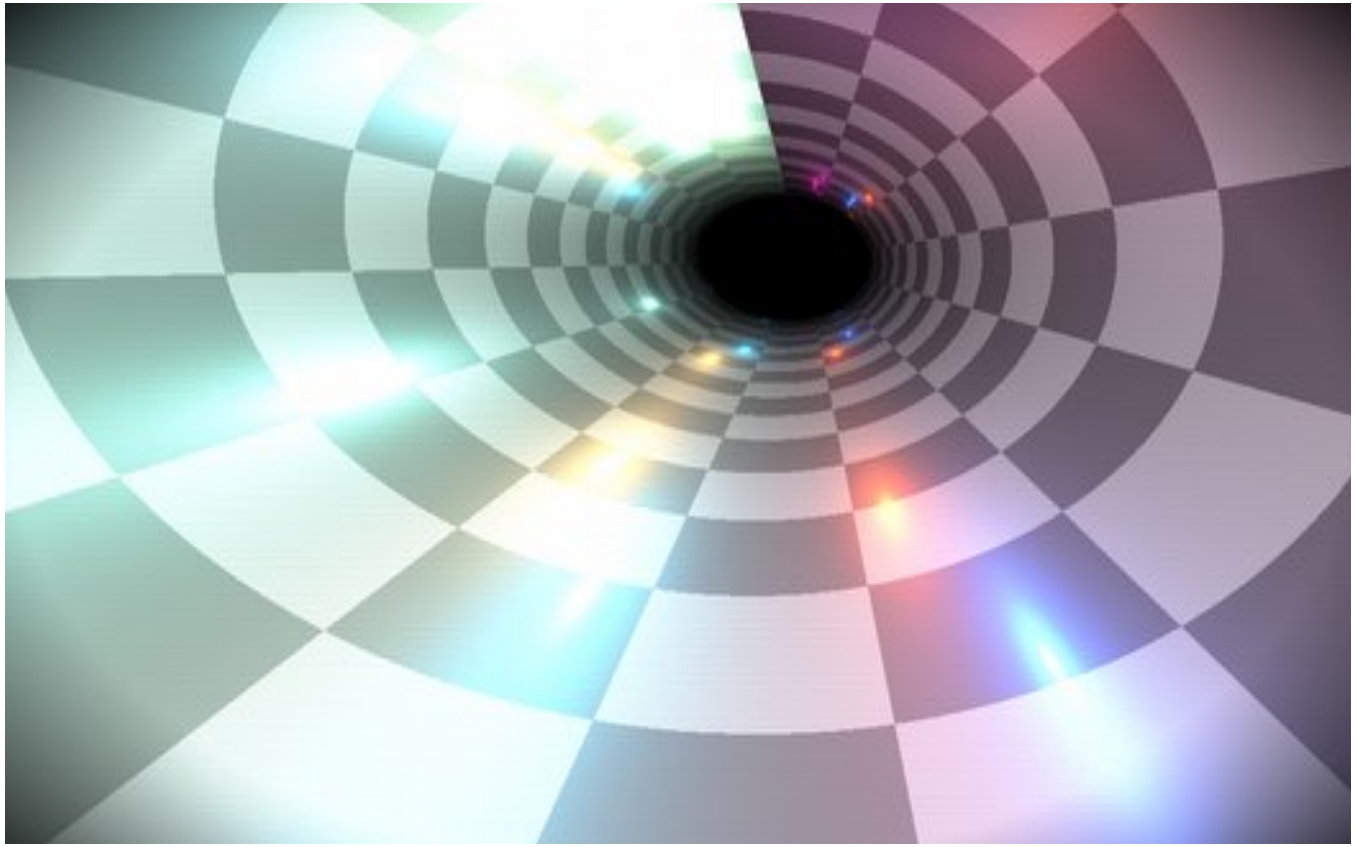
- **WebGL Specification https://www.khronos.org/registry/webgl/specs/1.0.3/**
- **Learning WebGL (Kurs und Blog) http://learningwebgl.com/blog/**
- **Einführung in three.js http://threejs.org/docs/index.html#Manual/Introduction/Creating_a_scene**
- **Online Kurs über die Grundprinzipien der 3D Computergrafik (2 Monate): meshes, transforms, cameras, materials, lighting, and animation https://www.udacity.com/course/interactive-3d-graphics--cs291**

Beispiel für 2D Bildverarbeitung mit Shadern

- **http://www.html5rocks.com/en/tutorials/webgl/webgl_fundamentals/webgl/webgl-2d-image-3x3-convolution.html**



Grössenlimitiertes Demo (4 kByte)



- **http://www.bitsnbites.eu/?p=112**