Information Visualization

W04: Exercise - CG Programming

Graduation School of System Informatics

Department of Computational Science

Naohisa Sakamoto, Akira Kageyama

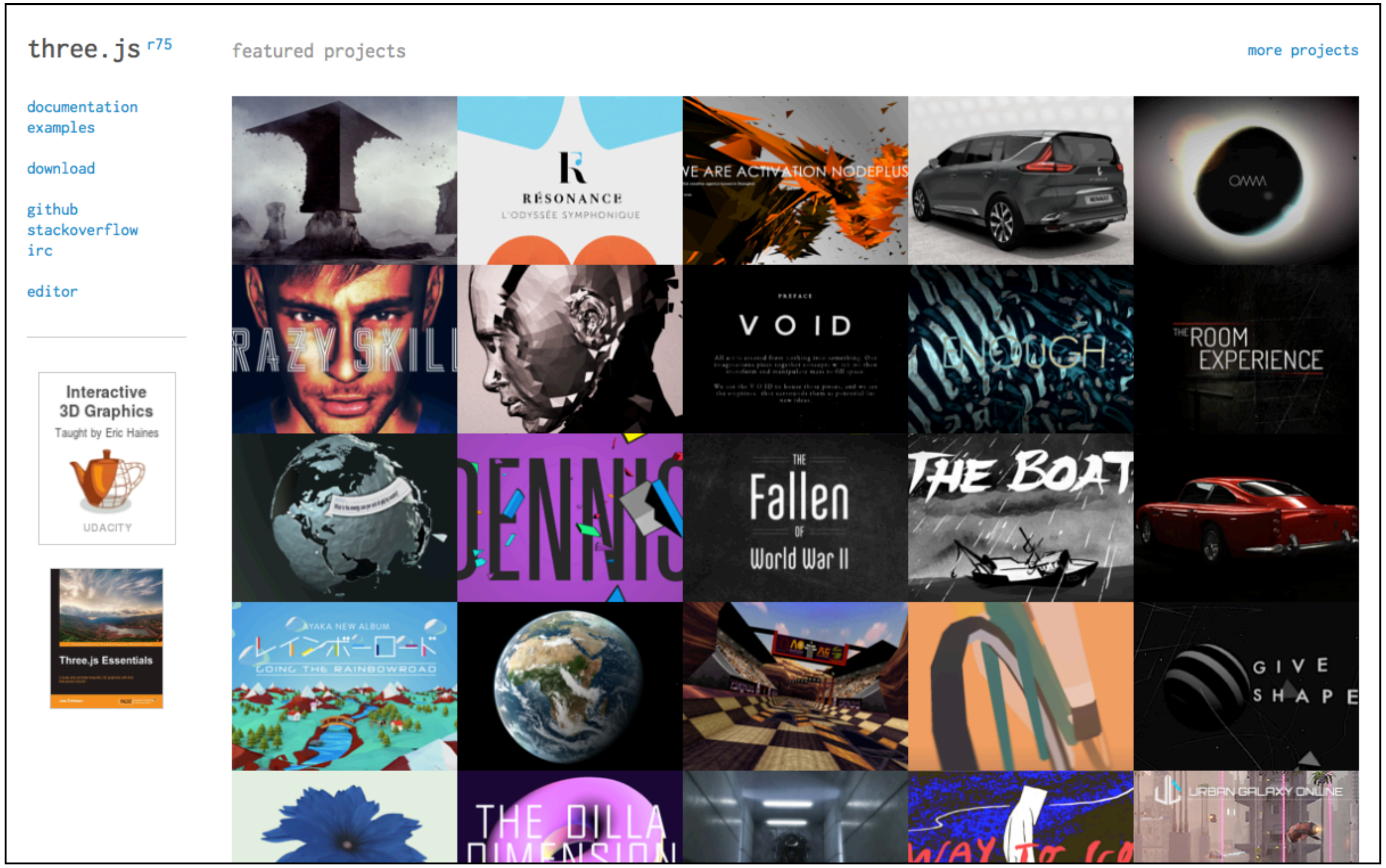
Apr. 19, 2017

Schedule

W01	4/11	Guidance
W02	4/12	Setup
W03	4/18	Introduction to Data Visualization
W04	4/19	CG Programming
W05	4/25	Rendering Pipeline
W06	4/26	Coordinate Systems and Transformations
W07	5/09	Shading
W08	5/10	Shader Programming
W09	5/16	Visualization Pipeline
W10	5/17	Data Model and Transfer Function
W11	5/23	Scalar Data Visualization 1 (Isosurface Extraction)
W12	5/24	Implementation of Isosurface Extraction
W13	5/30	Scalar Data Visualization 2 (Volume Rendering)
W14	5/31	Implementation of Volume Rendering
W15	6/06	Student Presentations

Getting Started with Three.js

- threejs.org



Three.js code

- Template

```
<html>
    <head>

    </head>

    <body>
        <script src="three.min.js"></script>

        <script>
            JavaScript code ...
        </script>
    </body>
</html>
```

Three.js code

- Template
  - Download the minified library (three.min.js)

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

  - Refer to the minified library on the Web

```
<script
src="http://mrdoob.github.io/three.js/build/three.min.js">
</script>
```

Basics

- The basic components to create a 3D world with Three.js
  - Scene
  - Camera
  - Renderer
  - Object

Scene

- THREE.Scene
  - Scenes allow you to set up what and where is to be rendered by three.js. This is where you place objects, lights and cameras.

Example

```
var scene = new THREE.Scene();
```

http://threejs.org/docs/index.html#Reference/Scenes/Scene

Camera

- THREE.Camera
  - Abstract base class for cameras. This class should always be inherited when you build a new camera.
    - Orthographic Camera
    - Perspective Camera

http://threejs.org/docs/index.html#Reference/Cameras/Camera

Orthographic Camera

- THREE.OrthographicCamera
  - Camera with orthographic projection.

Example

```
var camera = new THREE.OrthographicCamera(
    left,  // Camera frustum left plane
    right, // Camera frustum right plane
    top,   // Camera frustum top plane
    bottom,// Camera frustum bottom plane
    near,  // Camera frustum near plane
    far    // Camera frustum far plane
);

camera.position.set( x, y, z );

scene.add( camera );
```

http://threejs.org/docs/index.html#Reference/Cameras/OrthographicCamera

Perspective Camera

- THREE.PerspectiveCamera
  - Camera with perspective projection.

Example

```
var camera = new THREE.PerspectiveCamera(
    fov,    // Camera frustum vertical field of view
    aspect,// Camera frustum aspect ratio
    near,   // Camera frustum near plane
    far     // Camera frustum far plane
);

camera.position.set( x, y, z );

scene.add( camera );
```

http://threejs.org/docs/index.html#Reference/Cameras/PerspectiveCamera

Renderer

- THREE.WebGLRenderer
  - The WebGL renderer displays your beautifully crafted scenes using WebGL, if your device supports it.

Example

```
var renderer = new THREE.WebGLRenderer( params );
renderer->setSize( width, height );
document.body.appendChild( renderer.domElement );
```

`params` (optional object)

- `context` : The RenderingContext context to use.
- `precision` : Shader precision.
- `antialias` : Boolean, default is false
- `depth` : Boolean, default is true
- …

http://threejs.org/docs/index.html#Reference/Renderers/WebGLRenderer

Renderer

- THREE.WebGLRenderer
  - The WebGL renderer displays your beautifully crafted scenes using WebGL, if your device supports it.

Example

```
var params = { antialias: true, depth: true };
var renderer = new THREE.WebGLRenderer( params );
```

Example

```
var renderer = new THREE.WebGLRenderer({
    antialias: true,
    depth: true
});
```

http://threejs.org/docs/index.html#Reference/Renderers/WebGLRenderer

Object

- Object = Geometry + Material
  - An object is composed of a geometry and a material in Three.js.
- Geometry
  - A geometry holds all data necessary to describe a 3D model.
- Material
  - A material describes the surface appearance of a 3D model.

http://threejs.org/docs/#Reference/Core/Geometry
http://threejs.org/docs/#Reference/Materials/Material

Creating the cube object

- THREE.BoxGeometry
  - BoxGeometry is the quadrilateral primitive geometry class.

Example

```
var geometry = new THREE.BoxGeometry(
    width,  // Width of the sides on the X axis
    height, // Height of the sides on the Y axis
    depth   // Depth of the sides on the Z axis
);
```

http://threejs.org/docs/#Reference/Extras.Geometries/BoxGeometry

Creating the cube object

- THREE.MeshBasicMaterial
  - A material for drawing geometries in a simple shaded (flat or wireframe) way.

Example

```
var material = new THREE.MeshBasicMaterial(
    params
);
```

`params` (optional object)

- `color` : Geometry color in hexadecimal. Default is `0xffffff`.
- `shading` : Define shading type. Default is `THREE.SmoothShading`.
- `wireframe` : Render geometry as wireframe. Default is `false`.
- …

http://threejs.org/docs/#Reference/Materials/MeshBasicMaterial

Creating the cube object

- THREE.Mesh
  - Base class for Mesh objects.

Example

```
var geometry = new THREE.BoxGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial();

var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
```

http://threejs.org/docs/#Reference/Objects/Mesh

Rendering the scene

- To draw the scene, we need a render loop as follows:

Example

```
loop();

function loop()
{
    requestAnimationFrame( loop );
    renderer.render( scene, camera );
}
```

Rotating the cube

- To rotate the cube, we add the following code (red part) in the loop function.

Example

```
loop();

function loop()
{
    requestAnimationFrame( loop );
    cube.rotation.x += 0.001;
    cube.rotation.y += 0.001;
    renderer.render( scene, camera );
}
```

Result

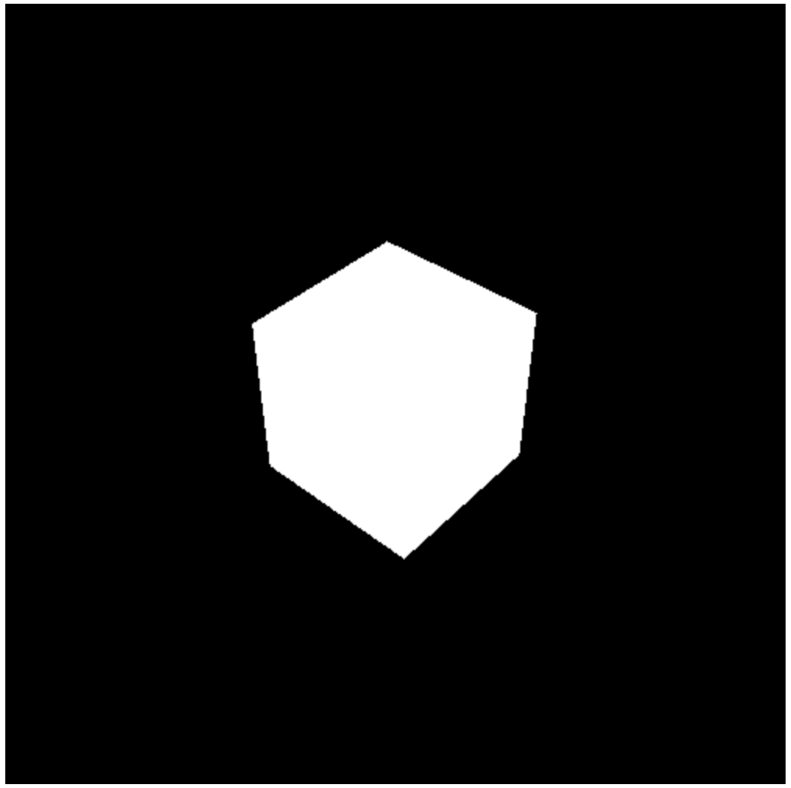
- Example code of a rotating cube.

w04_ex01.html

```
<html>
    <head>
        <title>W04: Example 01</title>
    </head>
    <body>
        <script src="three.min.js"></script>
        <script src="main.js"></script>
        <script>
            main();
        </script>
    </body>
</html>
```

Task 1

- Download a file named as main.js.
- Open w04_ex01.html with your web browser.



Task 2

- Add a point light to the scene
  - cf. `THREE.DirectionalLight`

Example

```
var light = new THREE.PointLight( 0xffffff );
light.position.set( 1, 1, 1 );
scene.add( light );
```

- Change the material to a Lambert material
  - cf. `THREE.MeshPhongMaterial`

Example

```
var material = new THREE.MeshLambertMaterial({
    color: 0xffffff
});
```

Polling

- Take the poll
  - Student ID Number
  - Name
  - URL to Task 1
  - URL to Task 2