Three.js Tutorial

http://threejs.org

https://github.com/mrdoob/three.js/

UNIVERSITY of NOTRE DAME OFFICE of the PROVOST

HesburghLibraries
UNIVERSITY OF NOTRE DAME

Website search »

Search & Find | Help & Guides | Libraries & Collections | About the Libraries | Ask a Librarian

OneSearch | Catalog | Articles | eJournals | Databases | eBooks

three.js SEARCH

Advanced search | eBook collections | Google Books | HathiTrust (FAQ) | About eBooks

1 2 3 4 5 6 7

REN-ALERTS Get Hesburgh Library Renovation Updates: email address

05/13/2016 - Elevator & East Entrance Closures

05/06/2016 - Elevator Closures: May 9th and 10th

Services
- My Library Accounts
- Library Reserves
- Circulation
- Interlibrary Loan
- Document Delivery
- Recent Acquisitions
- Electronic Resource Trials
- LINk (Library Information Network)
- ULRA (Undergraduate Library Research Award)
- CurateND
- One Button Studio

Resources & Tools
- LibGuides
- Subjects A-Z
- RefWorks
- Starting your research
- Pot of Gold tutorial
- Remix Digital Resource Portal
- CurateND

Popular Databases
- Academic Search Premier
- Google Scholar
- JSTOR
- Web of Science
- WorldCat

Contact
- Report a problem
- Request forms
- Workshop Registration

Chat with a Librarian

Building Hours

Monday - Friday	7:30AM to 11:00PM
Saturday	9:00AM to 11:00PM
Sunday	10:00AM to 11:00PM

Memorial Day
Effective on May 30th
Monday Closed

Hesburgh Service Desks

Au Bon Pain Cafe
Concourse opens early on weekends for Cafe.

24 Hour Access Procedures

- Online access available via ND library

Learning Objectives

- Students completing this lecture will be able to
  - Explain the need and benefits of using Three.js
  - Distinguish the similarities and differences between writing Three.js code and writing WebGL code
  - Write Three.js programs with reference to this set of slides and Three.js examples

Three.js

- A cross-browser JavaScript library/API used to create and display animated 3D computer graphics in a web browser
- Use WebGL and access to full GLSL capabilities
- First released by Ricardo Cabello in April 2010
- Allow us to create cool programs with a very low level of complexity (much less code to write than WebGL!)

An Example of Geometry Viewer

Getting Started

- Download Three.js package from http://threejs.org
- Open the folder and go to the build folder, copy *either* three.js *or* three.min.js into your local development directory
- You also need other js files such as OBJLoader.js and OrbitControls.js for our tutorial, find them and copy over
  - Examples > js > loaders > OBJLoader.js
  - Examples > js > controls > OrbitControls.js

Getting Started

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

- The full version of the Three.js library

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

- The minimalist version of the Three.js library

```
<script src="js/OBJLoader.js"></script>
<script src="js/OrbitControls.js"></script>
```

- Other utility library

```
var scene, camera, renderer, controls;
```

- Create some global variables

Create the Scene

```
var scene;

// Create the scene and set the scene size.
scene = new THREE.Scene();
var WIDTH = window.innerWidth;
var HEIGHT = window.innerHeight;
```

- Three.js uses the concept of a scene where you can place things like geometry, lights, cameras, and so on

Create the Renderer

```
var renderer;

// Create a renderer and add it to the DOM (document
object model).
renderer = new
THREE.WebGLRenderer({antialias:true});
renderer.setSize(WIDTH, HEIGHT);
document.body.appendChild(renderer.domElement);
```

- Create a WebGL renderer to take advantage of the GPU
- Apply it to the DOM via the body element, making three.js create a canvas inside the body element that will be used to render our scene

Create the Camera

```
var camera;

// Create a camera, zoom it out from the model a
bit, and add it to the scene.
camera = new THREE.PerspectiveCamera(45, WIDTH /
HEIGHT, 1, 2000);
camera.position.set(0,0,100);
scene.add(camera);
```

- Create a perspective camera with FOV, aspect ratio, near and far as parameters
- Set the position of the camera
- Add the camera to the scene

Add Lighting

```
// Set the background color of the scene.
renderer.setClearColor(0x333F47, 1);
// Create a point light.
var pointLight = new THREE.PointLight(0xffffff);
pointLight.position.set(0,10,0);
scene.add(pointLight);
// Create a directional light.
var directionalLight = new
    THREE.DirectionalLight(0xffeedd);
directionalLight.position.set(0,0,1);
scene.add(directionalLight);
```

- Create a point light and a directional light and add them to the scene

Update the Viewport on Resize

```
// Create an event listener that resizes the
renderer with the browser window.
window.addEventListener('resize', function() {
  var WIDTH = window.innerWidth;
  var HEIGHT = window.innerHeight;
  renderer.setSize(WIDTH, HEIGHT);
  camera.aspect = WIDTH / HEIGHT;
  camera.updateProjectionMatrix();
});
```

- Simply call `updateProjectionMatrix()` after setting the new size of our renderer and recalculating the aspect ratio of the camera

Load Texture Image

```
// texture
var texture = new THREE.Texture();
var imageLoader = new THREE.ImageLoader();
imageLoader.load( "models/kia.jpg", function (image)
{
  texture.image = image;
  texture.needsUpdate = true;
});
```

- Define an image loader that loads in a texture

Load Geometry

```
// Load in the mesh and add it to the scene.
var objLoader = new THREE.OBJLoader();
objLoader.load( "models/kia.obj", function (object)
{
  object.traverse(function (child) {
    if (child instanceof THREE.Mesh) {
      child.material.map = texture;
    }
  });
  scene.add(object);
});
```

- Define an object loader that loads in an obj file

Add Controls

```
 // Add OrbitControls so that we can pan around with
the mouse.
controls = new THREE.OrbitControls(camera,
renderer.domElement);
```

- OrbitControls allow us to
  - Drag the mouse across the mesh and orbit around it
  - Zoom in and out of the mesh with the mouse wheel

Finally, Render the Scene

```
// Renders the scene and updates the render as
needed.
function animate() {
  requestAnimationFrame(animate);
  // Render the scene.
  renderer.render(scene, camera);
  controls.update();
}
```

- OrbitControls allow us to
  - Drag the mouse across the mesh and orbit around it
  - Zoom in and out of the mesh with the mouse wheel

Questions

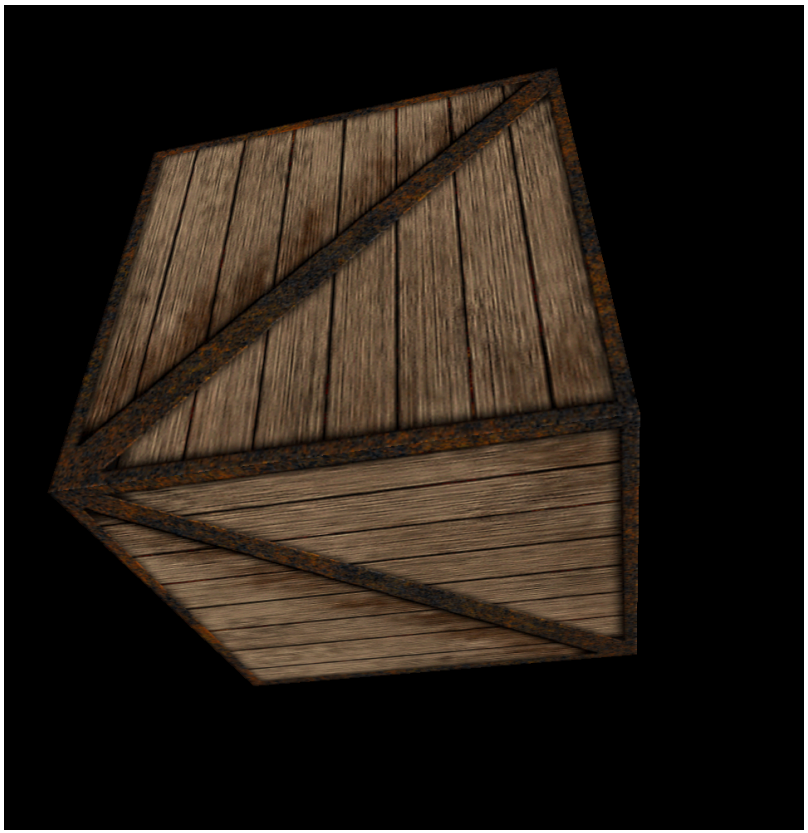
- What are the similarities between writing Three.js code and writing WebGL code?
- What are the differences between them?
- Well, you are using WebGL renderer in this example, but where is the GLSL shader code?

Exercise

- Update the example code tutorial-geometry-viewer.html to load in a JSON (JavaScript Object Notation) file
- JSON file is “models/logo.js”
- Use Lambert material for the mesh
- Check these three.js webpages for reference
  - http://threejs.org/docs/index.html#Reference/Loaders/JSONLoader
  - http://threejs.org/docs/index.html#Reference/Materials/MeshLambertMaterial

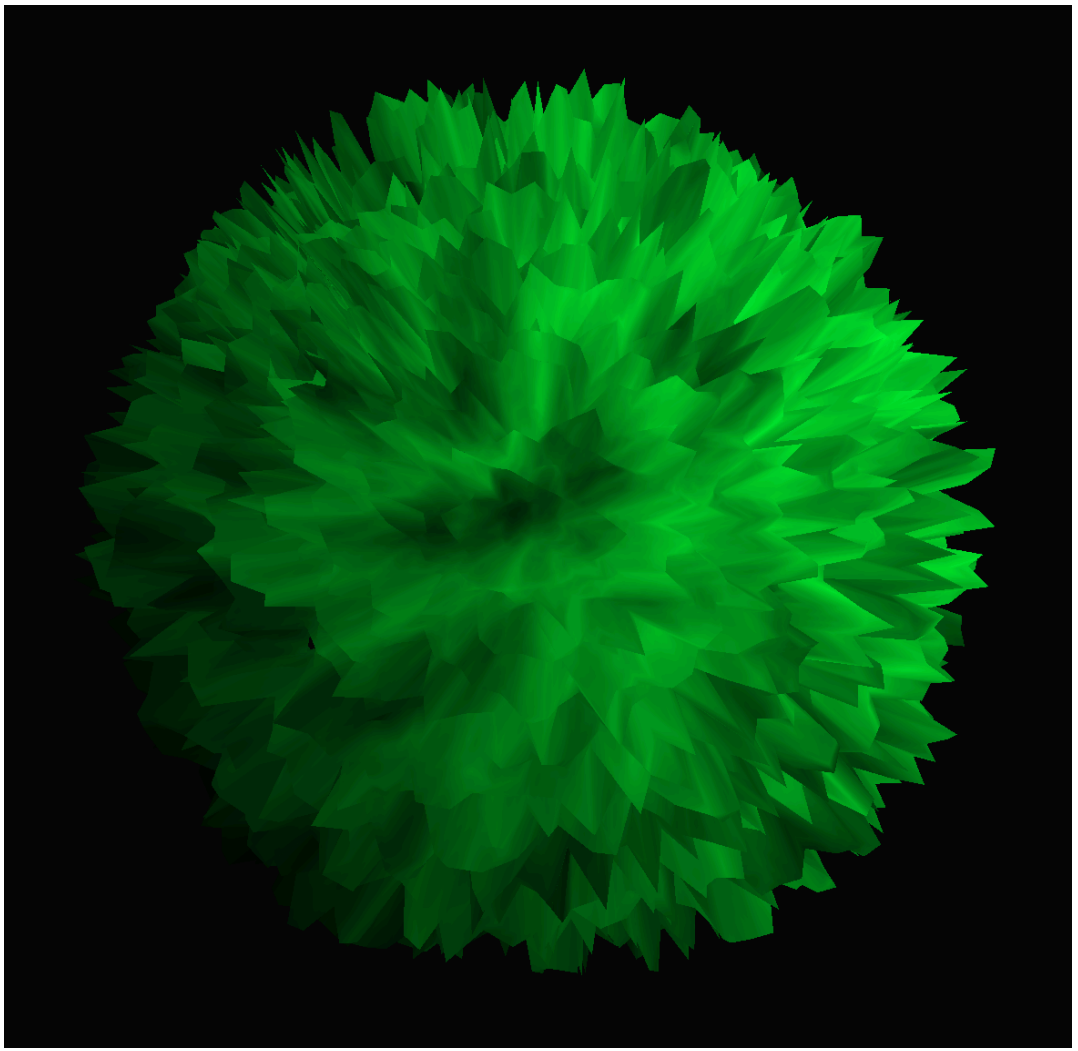
More Three.js Examples
(Let us do reverse engineering!)

webgl_geometry_cube



- You should have no difficulty in understanding this example.

webgl_custom_attributes



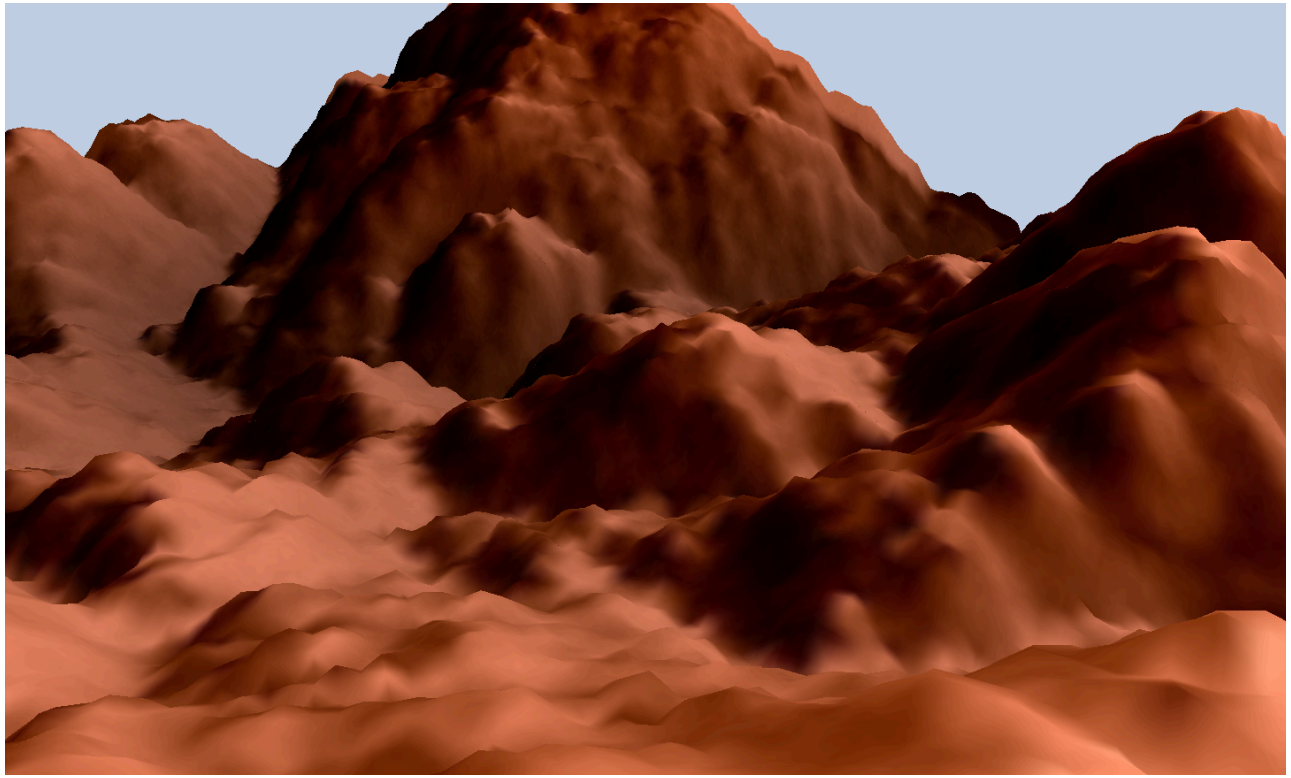
Exercise

- Run the program
- Go over the code and try to understand it the best you could
- Note that this example writes custom shaders
- Check out the following references
  - http://threejs.org/docs/#Reference/Materials/ShaderMaterial
  - http://threejs.org/docs/#Reference/Renderers.WebGL/WebGLProgram
- There would be some questions for you in the next slide

Questions

- What is the role of “Detector”?
- What are the roles of “container” and “stat”?
- What are the roles of “displacement” and “noise”?
- How do you make connection between values specified in the program and values specified in the shader?
- What are the built-in uniforms and attributes in shaders?

webgl_geometry_terrain



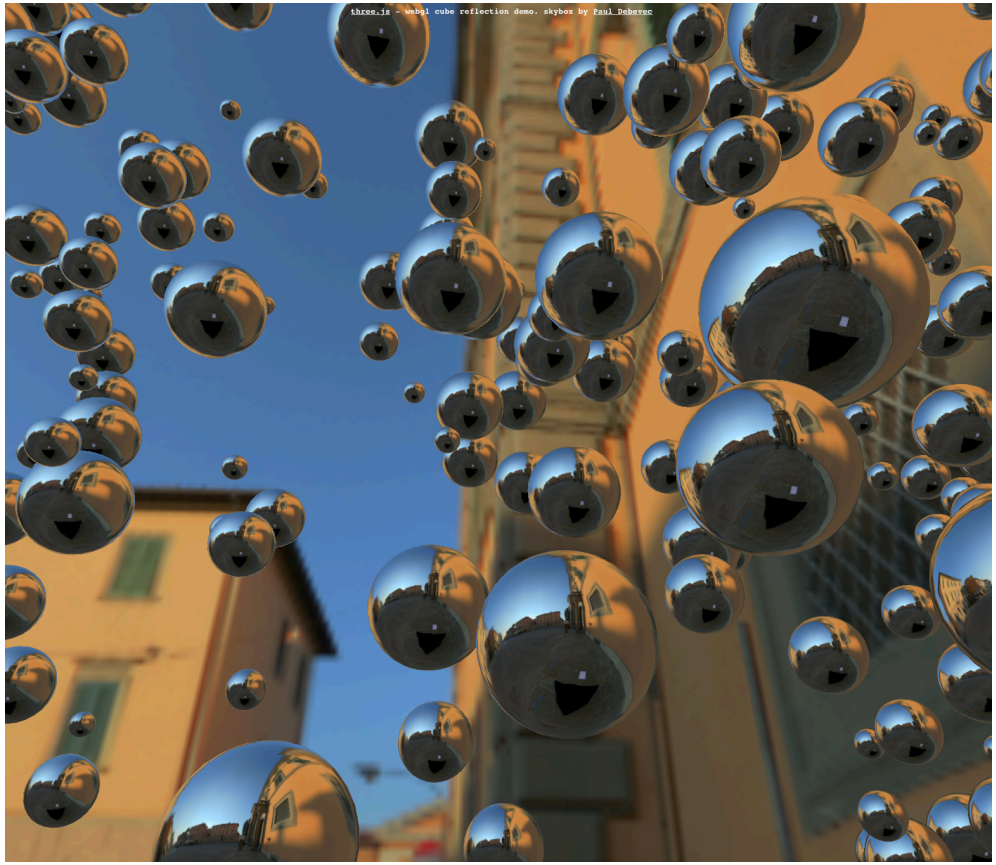
Questions

- What is the first person controls?
- How the terrain is generated?
  - What does this loop do?

```
for ( var j = 0; j < 4; j ++ ) { ... }
```

- What the role of texture used in this program?

webgl_materials_cubemap_balls_reflection



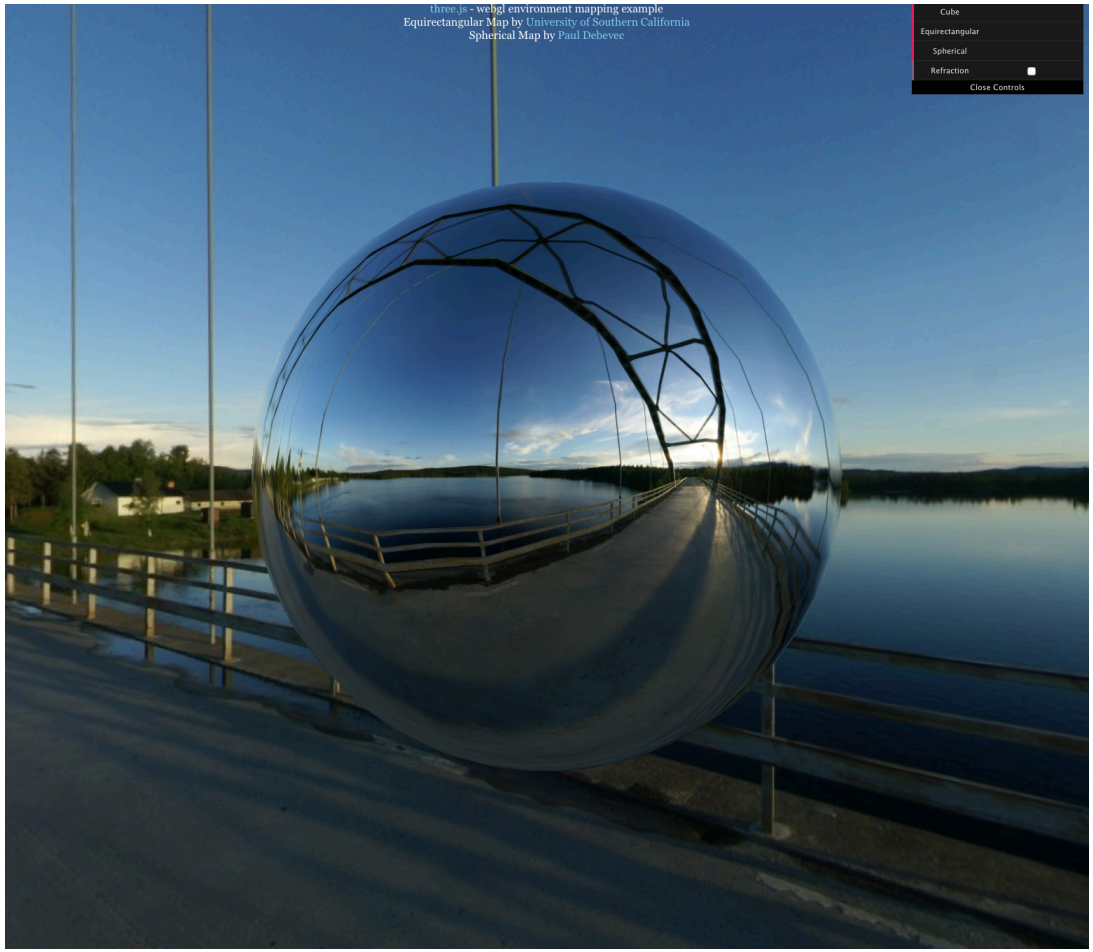
Questions

- What is the role of the cube map?
- How the cube map is set up?
- Change camera.position.z from 3200 to 0 or -3200 and see the difference

Cubemap images

	x	y	z
-			
+			

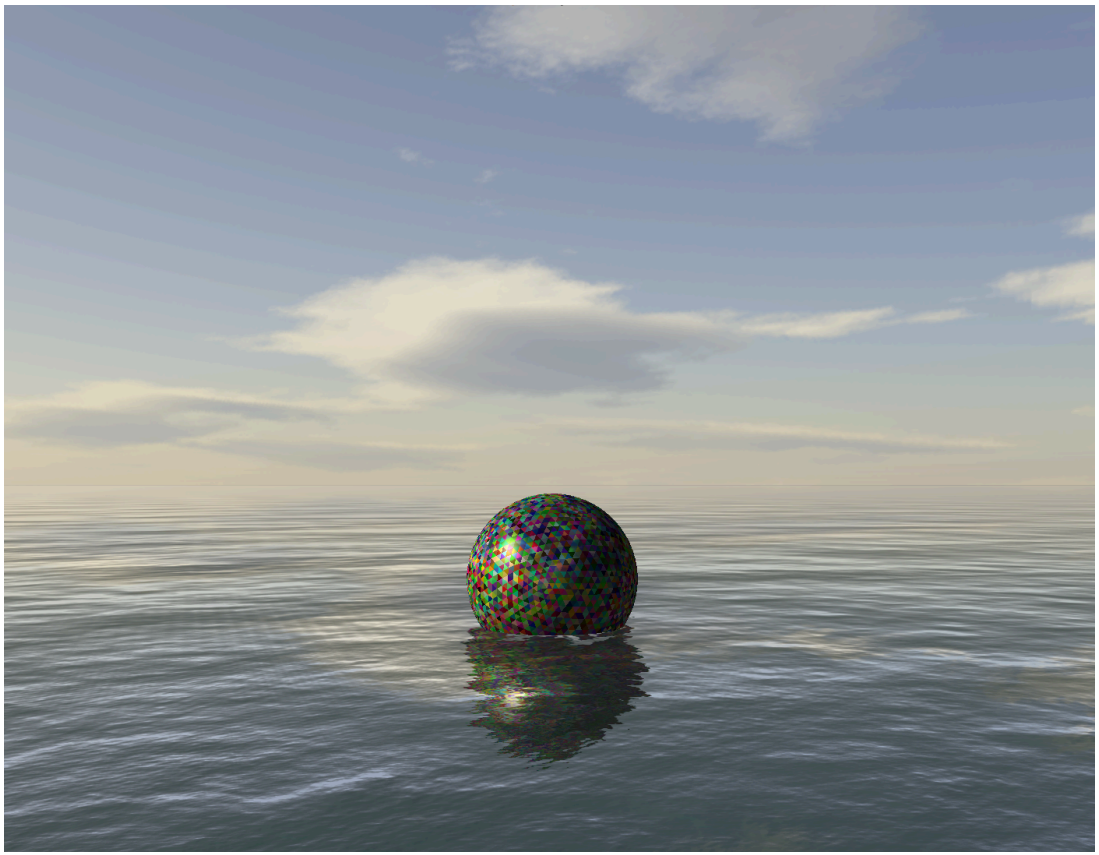
webgl_materials_envmaps



Questions

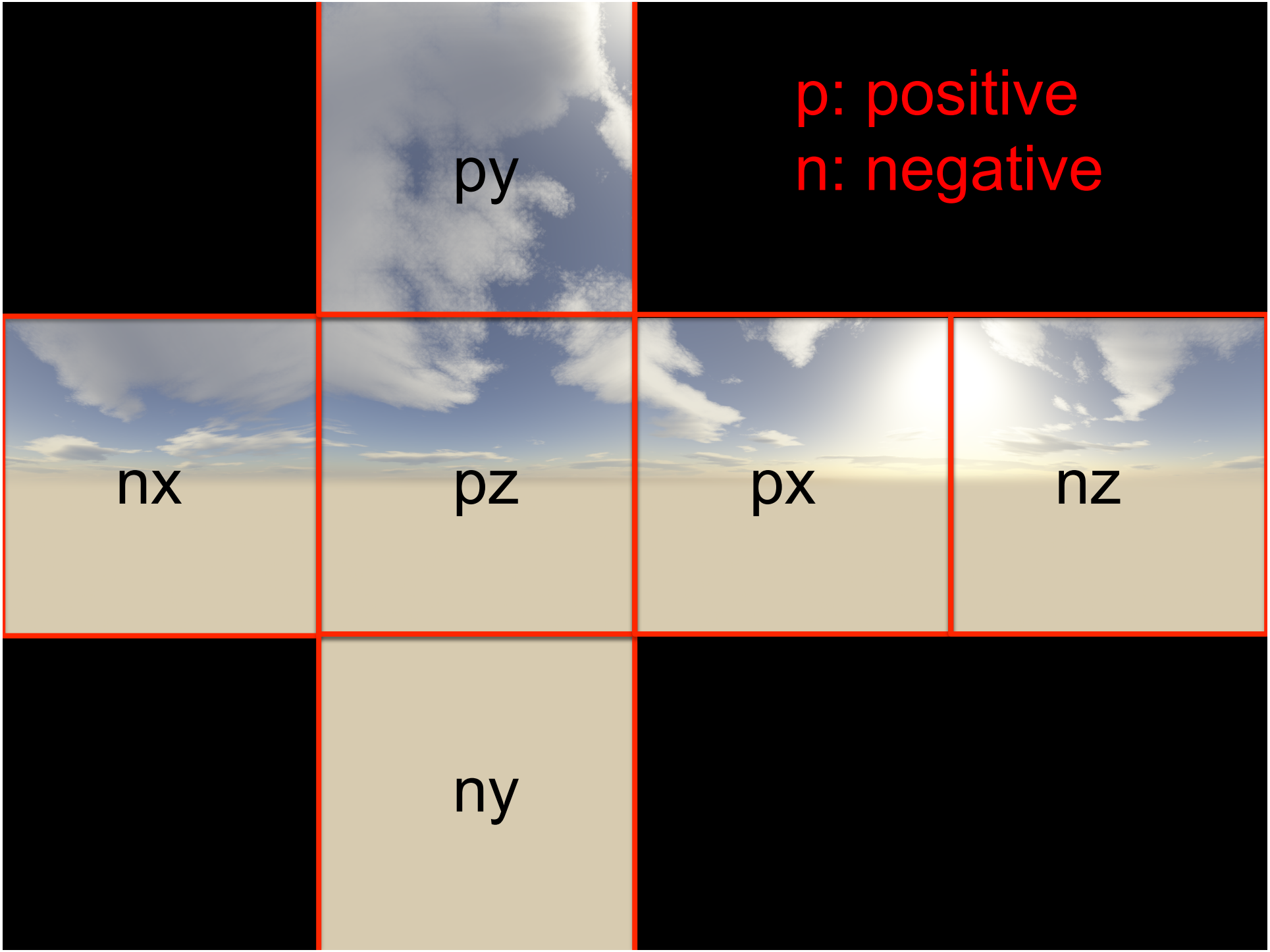
- What are the differences among
  - Cube mapping
  - Equirectangular (360 spherical panoramas) mapping
  - Spherical environment mapping?
- How to set up the menu-like interface?
- What is the difference between reflection and refraction?
- Why we need two scenes and two cameras (scene, sceneCube, camera, cameraCube)?

webgl_shader_ocean

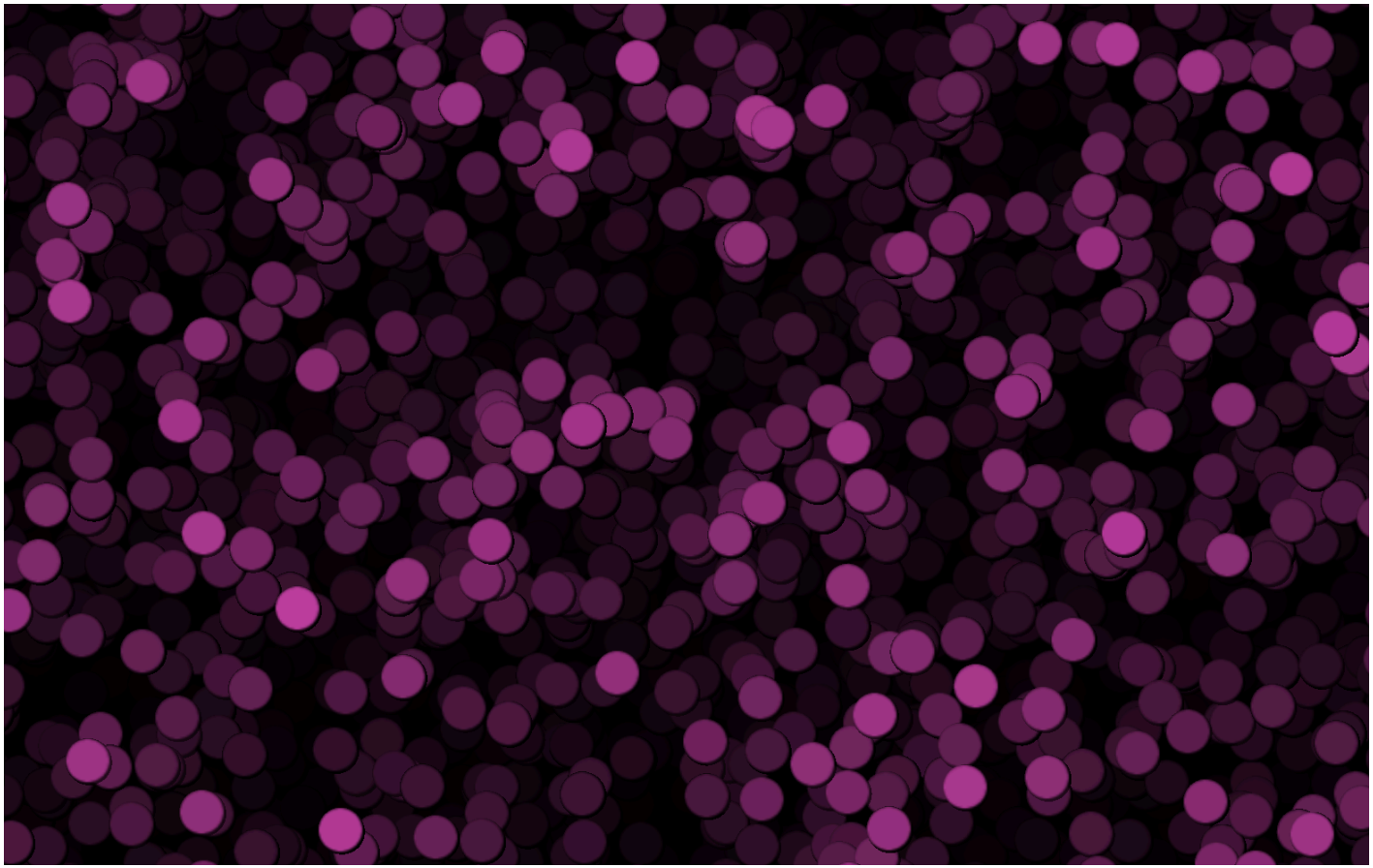


Questions

- What is the role of normal map for water?
- How the cube map is set up (any difference of the cube map texture from the previous examples)?
- What is the role of the environment map on the sphere?
- What setting makes the sphere to cast shadow on the water?


py
p: positive
n: negative
nx
pz
px
nz
ny

webgl_points_billboards



Questions

- What are sprites?
  - 2D bitmap that is integrated into a larger scene
- What is billboard?
  - Sprites that are texture mapped 3D facets that always have their surface normal facing into the camera
- What are the geometries?
- What is the use of texture?
- What makes this program interactable on touch screen?

webgl_morphnormals

Questions

- What information is stored in the animated flamingo model (flamingo.js)?
  - Vertices, morph vertices, normals, morph normals?
- What is the role of animation mixer?
- What is the different between flat shading and smooth shading?