Intro to THREE.js

Dr. Mihail

November 2, 2015

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects
- Implements the concept of “scene” to which you can add/remove objects during runtime

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects
- Implements the concept of “scene” to which you can add/remove objects during runtime
- Cameras

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects
- Implements the concept of “scene” to which you can add/remove objects during runtime
- Cameras
- Animation (skinning, forward kinematics, inverse kinematics, morph, keyframe, etc.)

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects
- Implements the concept of "scene" to which you can add/remove objects during runtime
- Cameras
- Animation (skinning, forward kinematics, inverse kinematics, morph, keyframe, etc.)
- Lights

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects
- Implements the concept of “scene” to which you can add/remove objects during runtime
- Cameras
- Animation (skinning, forward kinematics, inverse kinematics, morph, keyframe, etc.)
- Lights
- Materials

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects
- Implements the concept of “scene” to which you can add/remove objects during runtime
- Cameras
- Animation (skinning, forward kinematics, inverse kinematics, morph, keyframe, etc.)
- Lights
- Materials
- Shaders (access to full GLSL)

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects
- Implements the concept of “scene” to which you can add/remove objects during runtime
- Cameras
- Animation (skinning, forward kinematics, inverse kinematics, morph, keyframe, etc.)
- Lights
- Materials
- Shaders (access to full GLSL)
- Geometry (plane, cube, sphere, torus, 3D text)

What is THREE

- High-level library built on top of WebGL. THREE makes it possible to author complex 3D graphics with minimal effort.

Features:

- Highly object oriented
- Many built in effects
- Implements the concept of “scene” to which you can add/remove objects during runtime
- Cameras
- Animation (skinning, forward kinematics, inverse kinematics, morph, keyframe, etc.)
- Lights
- Materials
- Shaders (access to full GLSL)
- Geometry (plane, cube, sphere, torus, 3D text)
- Data loaders (textuers and models)

THREE

Basic idea

- Scene
- Camera
- Lights
- Action

Download

Download three.js

`threejs.org`

The download will contain all the source code, including examples, etc.

You need the ./build folder.

HTML

Basics: index.html

```html

<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="./style.css">
    <script src="./three.js"></script>
</head>
<body>
    <script src="./main.js"></script>
</body>
</html>
```

HTML

Basics: index.html

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="./style.css">
    <script src="./three.js"></script>
</head>
<body>
    <script src="./main.js"></script>
</body>
</html>
```

HTML

Basics: style.css

```
canvas {
    position: fixed;
    top: 0;
    left: 0;
}
```

HTML

Basics: main.js

```

// initialize WebGL and THREE renderer

var width = window.innerWidth;
var height = window.innerHeight;


var renderer = new THREE.WebGLRenderer({
   antialias: true });
renderer.setSize(width, height);
document.body.appendChild(renderer.domElement);
```

HTML

Basics: main.js

```

// initialize WebGL and THREE renderer

var width = window.innerWidth;
var height = window.innerHeight;


var renderer = new THREE.WebGLRenderer({
   antialias: true });
renderer.setSize(width, height);
document.body.appendChild(renderer.domElement);
```

Scene

Basics: main.js

```

// create scene object
var scene = new THREE.Scene;

// create simple geometry and add to scene
var cubeGeometry = new THREE.CubeGeometry(15,15,
    15);
var cubeMaterial = new THREE.MeshLambertMaterial
    ({ color: 0xaaff44 });
var cube = new THREE.Mesh(cubeGeometry,
    cubeMaterial);
scene.add(cube);
```

Camera

Basics: main.js

```
// create perspective camera
var camera = new THREE.PerspectiveCamera(45,
   width / height, 0.1, 10000);
camera.position.y = 16;
camera.position.z = 40;
// add to scene and renderer
scene.add(camera);
renderer.render(scene, camera);
// create the view matrix (lookAt)
camera.lookAt(cube.position);
```

Lights

Basics: main.js

```
// add lighting and add to scene
var pointLight = new THREE.PointLight(0xaabbcc);
pointLight.position.set(0, 16, 16);
scene.add(pointLight);
```

Action

Basics: main.js

```
renderer.render(scene, camera);
function render() {
    renderer.render(scene, camera);
    requestAnimationFrame(render);
    cube.rotation.y+=0.01; // animate
}
render();
```

Textures

Basics: main.js

```
  var cubeMaterial = new THREE.
    MeshLambertMaterial({ map: THREE.ImageUtils.
    loadTexture('crate.jpg')});
```

Models

3D Models

Asynchronously:

- Load the model's texture maps
- Load the model
- Add to scene

Textures

Basics: main.js

```
var texture = new THREE.Texture();
var loader = new THREE.ImageLoader( manager );
loader.load( 'UV_Grid_Sm.jpg', function ( image )
    {
  texture.image = image;
  texture.needsUpdate = true;
} );
```

Model

Housekeeping

```
var manager = new THREE.LoadingManager();
manager.onProgress = function ( item, loaded,
    total ) {
    console.log( item, loaded, total );
};
var onProgress = function ( xhr ) {
  if ( xhr.lengthComputable ) {
    var percentComplete = xhr.loaded / xhr.total
        * 100;
    console.log( Math.round(percentComplete, 2) +
        '% downloaded' );
  }
};
var onError = function ( xhr ) { };
```

Model

Loading the model

```
var loader = new THREE.OBJLoader( manager );
loader.load( 'male02.obj', function ( object ) {
  object.scale.set(0.5, 0.5, 0.5);
  object.position.y = -50;
  object.traverse( function ( child ) {
  if ( child instanceof THREE.Mesh ) {
    child.material.map = texture;
  }
} );
scene.add( object );
}, onProgress, onError );
```