**Three.js**:- is a cross-browser JavaScript library/API used to create and display animated 3D computer graphics in a web browser

**objLoader.js**

In order to load object files (.obj), we used OBJLoader.js

**MTLLoader.js**

In short, this is an addition for the OBJ format. Because MTL is a file that describes materials of OBJ file.

All these javascript files we included in the index.html as shown below

<script type="text/javascript" src="three.min.js" ></script>

<script type="text/javascript" src="MTLLoader.js" ></script>

<script type="text/javascript" src="OBJLoader.js" ></script>

We created our own javascript file named Demo.js which is also included in the index.html file

<script type="text/javascript" src="demo.js" ></script>

----------------------------------------

In demo.js

**We created variables like**

scene, camera, renderer, mesh, meshFloor, ambientLight, light

**for textures we created variables**

crate, crateTexture, crateNormalMap, crateBumpMap;

**to rotate the objects we created player variable**

var player = { height:1.8, speed:0.2, turnSpeed:Math.PI\*0.02 };

for not to see the objects in wireframe mode we declared a variable USE\_WIREFRAME and the value we gave is false.

var USE\_WIREFRAME = false;

**now we initialize the scene with** THREE.Scene();

**the camera we created to view the scene is**

PerspectiveCamera(75, 1366/768, 0.1, 10000);

PerspectiveCamera( fov, aspect, near, far )

fov — Camera frustum vertical field of view.

aspect — Camera frustum aspect ratio.

near — Camera frustum near plane.

far — Camera frustum far plane.

Now we have to create a cube by coding

BoxGeometry

BoxGeometry is the quadrilateral primitive geometry class. It is typically used for creating a cube.

Having width,height,and depth .

To apply the materials we used MeshPhongMaterial

mesh = new THREE.Mesh(

new THREE.BoxGeometry(1,1,1),

new THREE.MeshPhongMaterial({color:0xff4444, wireframe:USE\_WIREFRAME})

);

Now we have to position the generated CUBE

From ground to up

mesh.position.y += 1;

ie

mesh.position.y = mesh.position.y +1;

**now the cube needs to generate shadow**

mesh.castShadow = true;

**and cube needs to receive the shadow**

mesh.receiveShadow = true;

**and add the cube to the scene**

scene.add(mesh);

---------------------------------------------------------

Now we have to add the floor

PlaneGeometry

A class for generating plane geometries

The syntax for plane is

PlaneGeometry(width, height, widthSegments, heightSegments)

width — Width along the X axis.

height — Height along the Y axis.

widthSegments — Optional. Default is 1.

heightSegments — Optional. Default is 1.

new THREE.PlaneGeometry(20,20, 10,10)

**To apply the materials we used MeshPhongMaterial**

**And the floor color is white**

**Here we set rotation for proper setting the floor to receive shadows**

meshFloor.rotation.x -= Math.PI / 2;

**to recive shadows we have to make**

meshFloor.receiveShadow = true;

**add the floor to the scene**

scene.add(meshFloor);

---------------------------------------------------

In this scene we are placing two lighting sources

1.ambient light

2.point light

1)AmbientLight

This light's color gets applied to all the objects in the scene globally.

Syntax

AmbientLight( color, intensity )

color — Numeric value of the RGB component of the color.

intensity -- Numeric value of the light's strength/intensity.

This creates an Ambientlight with a color and intensity.

If you give more intensity more lighting will come

We used the color white and the intensity is 0.2

ambientLight = new THREE.AmbientLight(0xffffff, 0.2);

we added the ambient light to the scene

scene.add(ambientLight);

2)point light

Creates a light at a specific position in the scene. The light shines in all directions (roughly similar to a light bulb.)

syntax

PointLight( color, intensity, distance, decay )

color — Numeric value of the RGB component of the color.

intensity — Numeric value of the light's strength/intensity.

distance -- The distance of the light where the intensity is 0. When distance is 0, then the distance is endless.

decay -- The amount the light dims along the distance of the light.

light = new THREE.PointLight(0xffffff, 0.8, 18);

we have to set the position of the point light

light.position.set(-3,6,-3);

to cast the shadow we have to enable it and the shadow near and far distance of the camera

i.e

light.castShadow = true;

light.shadow.camera.near = 0.1;

light.shadow.camera.far = 25;

scene.add(light);

----------------------------------------------------------------------------------------------

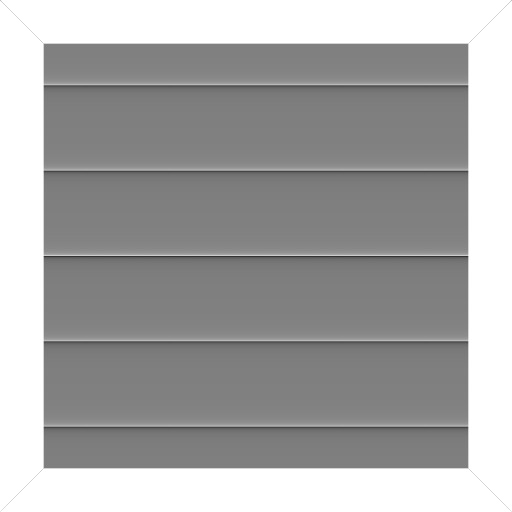
Second cube with textures

var textureLoader = new THREE.TextureLoader();

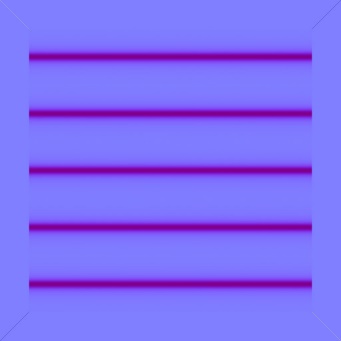
Class for loading a texture.



crateTexture = textureLoader.load("crate0/crate0\_diffuse.jpg");



crateBumpMap = textureLoader.load("crate0/crate0\_bump.jpg");



crateNormalMap = textureLoader.load("crate0/crate0\_normal.jpg");

with these three images we added to the BoxGeometry and the size of the geometry is 3 and the below code is the complete cube with the textures applied to it, added to the scene ,position is also setted with casting shadows and received shadows

crate = new THREE.Mesh(

new THREE.BoxGeometry(3,3,3),

new THREE.MeshPhongMaterial({

color:0xffffff,

map:crateTexture,

bumpMap:crateBumpMap,

normalMap:crateNormalMap

})

);

scene.add(crate);

crate.position.set(2.5, 3/2, 2.5);

crate.receiveShadow = true;

crate.castShadow = true;

-------------------------------------------------------------------------------------------------

Now we created 3 planes left right and back planes similar to the floor plane only the positions are and rotation changed

var geometry = new THREE.PlaneGeometry( 15, 20, 32 );

var material=new THREE.MeshPhongMaterial({color:0xffffff, side: THREE.DoubleSide})

var plane = new THREE.Mesh( geometry, material );

scene.add( plane );

plane.position.set(-3,0,3);

plane.receiveShadow = true;

var geometry = new THREE.PlaneGeometry( 23, 20, 32 );

var material=new THREE.MeshPhongMaterial({color:0xffffff, side: THREE.DoubleSide})

var plane = new THREE.Mesh( geometry, material );

scene.add( plane );

plane.position.set(8,4,1);

plane.receiveShadow = true;

plane.rotation.y = -Math.PI/2;

var geometry = new THREE.PlaneGeometry( 23, 20, 32 );

var material=new THREE.MeshPhongMaterial({color:0xffffff, side: THREE.DoubleSide})

var plane = new THREE.Mesh( geometry, material );

scene.add( plane );

plane.position.set(-7,4,1);

plane.receiveShadow = true;

plane.rotation.y = -Math.PI/2;

---------------------------------------------------------------------------------------------

**Loading of material and 3d obj model**

Here in this example after exporting 3d model form AUTODESK MAYA software in the format of OBJ ,we got 2 files

* male112.mtl
* male112.obj

**syntax of MTLLoader is**

MTLLoader( baseUrl, options, crossOrigin )

baseUrl — The base url from which to find subsequent resources.

options — Options passed to the created material (side, wrap, normalizeRGB, ignoreZeroRGBs).

crossOrigin — The crossOrigin string to implement CORS for loading the url from a different domain that allows CORS.

Creates a new MTLLoader.

.load ( url, onLoad, onProgress, onError )

url — required

onLoad — Will be called when load completes. The argument will be the loaded MTLLoader.MaterialCreator instance.

onProgress — Will be called while load progresses. The argument will be the XmlHttpRequest instance, that contain .total and .loaded bytes.

onError — Will be called when load errors.

Begin loading from url and return the loaded material.

var mtlLoader = new THREE.MTLLoader();

mtlLoader.load("models/male112.mtl", function(materials){

materials.preload();

after this we have to load the obj model with OBJLoader

var objLoader = new THREE.OBJLoader();

objLoader.setMaterials(materials);

objLoader.load("models/male112.obj", function(mesh){

mesh.traverse(function(node){

if( node instanceof THREE.Mesh ){

node.castShadow = true;

node.receiveShadow = true;

node.position.set(-3,0,-3);

}

});

In the above code we enabled the receive shadow and cast shadow and set the position also;

------------------------------------------------------------------------------------------------------------------

Here we are setting the perspective camera position and what it needs to project

camera.position.set(0, player.height, -5);

camera.lookAt(new THREE.Vector3(0,player.height,0));

now for rendering we are using WebGLRenderer

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

The size of the out put is 1280X720

renderer = new THREE.WebGLRenderer();

renderer.setSize(1280, 720);

renderer.shadowMap.enabled = true;

renderer.shadowMap.type = THREE.BasicShadowMap;

document.body.appendChild(renderer.domElement);

animate();

}

We enabled all shadowmaps for the rendering.

In the above code we raised one function animate()

Now we are working on animate function

W key, S key,A key D key funtions are added for zoom in zoom out move left move right

Left arrow rotation and right arrow for right rotation alike.

We added all this code to the page by writing

window.onload = init;