WebGL

web JavaScript WebGL api , 2 d 3 d ,\

- WebGL Three.js Babylon.js

WebGL Three.js Babylon.js

> WebGL Three.js Babylon.js

w`JavaScript WebGL api`,`2D` `3D` `WebGL` ,

, api WebGL , , , , , WebGL

Three.js Babylon.js `web` WebGL

`Three.js` 2009`JavaScript` `ActionScript` `WebGL` `three.js` `WebGL` `SVG` `HTML5`

`Babylon.js`, ,`Babylon.js` `WebGL API` `Internet Explorer 11`

Three.js and Babylon.js WebGL

HTML JavaScript :Babylon.js , hand.js

Three.js:

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

Babylon.js:

```
<script src="babylon.js"></script>
<script src="hand.js"></script>
```

`Three.js` `web` `GPU` `3d` `web` `Three.js` threejs.org)

`Three.js` `WebGL` `Babylon.js` `Silverlight` `Babylon.js` `web`

, 3d ,

3 d

Three.js:

```
<div style="height: 250px; width: 250px;" id="three"></div>
```

```
var div = document.getElementById('three');
```

Babylon.js:

```
var canvas = document.getElementById('babylonCanvas');
```

With Three.js we simply create an empty div as our container for the animation. Babylon.js on the other hand makes use of an explicitly defined HTML5 canvas to hold its 3D graphics.

Three.js div Babylon.js HTML5 canvas 3D

,

Three.js:

```
var renderer = new THREE.WebGLRenderer();
renderer.setSize(width, height);
div.appendChild(renderer.domElement);
```

Babylon.js:

```
var engine = new BABYLON.Engine(canvas, true);
```

,

cube

Three.js:

```
var sceneT = new THREE.Scene();
var camera = new THREE.PerspectiveCamera(70, width / height, 1, 1000);
camera.position.z = 400;
```

Babylon.js:

```
var sceneB = new BABYLON.Scene(engine);
var camera = new BABYLON.ArcRotateCamera("camera", 1, 0.8, 10, new BABYLON.Vector3(0, 0, 0),
sceneB);
sceneB.activeCamera.attachControl(canvas);
var light = new BABYLON.DirectionalLight("light", new BABYLON.Vector3(0, -1, 0), sceneB);
light.diffuse = new BABYLON.Color3(1, 0, 0);
light.specular = new BABYLON.Color3(1, 1, 1);
```

, ( ),

Babylon.js

,

Three.js:

```
var cube = new THREE.CubeGeometry(100, 100, 100);

var texture = THREE.ImageUtils.loadTexture('texture.gif');
texture.anisotropy = renderer.getMaxAnisotropy();
var material = new THREE.MeshBasicMaterial({ map: texture });

var mesh = new THREE.Mesh(cube, material);
sceneT.add(mesh);
```

Babylon.js:

```
var box = BABYLON.Mesh.CreateBox("box", 3.0, sceneB);
var material = new BABYLON.StandardMaterial("texture", sceneB);
box.material = material;
material.diffuseTexture = new BABYLON.Texture("texture.gif", sceneB);
```

First, we create our cube objects of the specified size and then create our material/mesh (think texture) that will be painted onto the cubes. Any image file will work for the texture and both frameworks support mesh exports from 3D modeling tools like Blender.

, , / texture, 3D (Blender)

, ,

Three.js:

```
function animate() {
     requestAnimationFrame(animate);
     mesh.rotation.x += 0.005;
     mesh.rotation.y += 0.01;
     renderer.render(sceneT, camera);
}
```

Babylon.js:

```
```

```
engine.runRenderLoop(function () {
      box.rotation.x += 0.005;
      box.rotation.y += 0.01;
      sceneB.render();
      });
```

`Three.js` `Babylon.js` `Three.js` `Babylon.js`

`WebGL` `Web` , ,

`web` `3D WebGL` `3D web`