



**VIBLO**  
**WE NEED YOU**

Drop us a message at [contact@viblo.asia](mailto:contact@viblo.asia)

- BACKEND DEVELOPERS WITH SKILLS IN [LARAVEL](#) [NODEJS](#)
- FRONTEND DEVELOPERS WITH SKILLS IN [VUEJS](#) [SASS](#)
- SMART PHONE DEVELOPERS WITH SKILLS IN [REACT NATIVE](#)
- DEVOPS ENGINEER WITH SKILLS IN [CI&CD](#) [LINUX](#) [DOCKER](#)



**Tan Nguyen**

[Follow](#)

Published Jun 24th, 2016 9:06 AM

# Tạo 1 cảnh động 3D đơn giản trên web với Three.js

[Game](#) [3D](#) [Three.js](#) [WebGL](#)

Jun 24th, 2016 9:06 AM

2325 2 4 Report



Chúng ta chuẩn bị tạo ra một chiếc máy bay 3D đơn giản sử dụng `three.js` - thư viện 3D giúp việc thao tác với `webGL` trở nên nhẹ nhàng hơn. `webGL` còn khá lạ lẫm với nhiều người vì sự phức tạp cũng như cú pháp GLSL. Nhưng với `three.js`, 3D trong trình duyệt trở nên rất dễ dàng.

Trong hướng dẫn này, chúng ta sẽ tạo ra một cảnh 3D đơn giản với một vài tương tác trong hai phần chính. Ở phần đầu tiên, chúng ta sẽ giải thích những khái niệm cơ bản của `three.js` và cách cài đặt. Phần thứ hai sẽ đi sâu hơn vào tinh chỉnh các hình khối, thêm các chi tiết và các chuyển động tốt hơn cho các thành phần.

Bắt đầu thôi!

## HTML & CSS

Hướng dẫn này sử dụng thư viện `three.js`. Xem trên [trang web](#) và [GitHub](#) để biết thêm thông tin chi tiết và [tải về](#).

Đầu tiên cần import thư viện vào trong `header HTML`:

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

Cần có 1 cái container để `render`:

```
<div id="world"></div>
```

Style cho cái div vừa tạo:

```
#world {  
  position: absolute;  
  width: 100%;  
  height: 100%;  
  overflow: hidden;  
  background: linear-gradient(#e4e0ba, #f7d9aa);  
}
```

Màu nền bầu trời. Xong phần `HTML` và `CSS` !

## JavaScript

`Three.js` rất dễ sử dụng nếu bạn có kiến thức cơ bản về `JavaScript` .

## Bảng màu



Sẽ rất hữu ích nếu chúng ta xác định bảng màu sẽ sử dụng xuyên suốt project. Đối với project này, chúng ta lựa chọn các màu sắc như sau:

```
var Colors = {  
  red:0xf25346,  
  white:0xd8d0d1,  
  brown:0x59332e,  
  pink:0xF5986E,  
  brownDark:0x23190f,  
  blue:0x68c3c0,  
};
```

## Cấu trúc Code

JavaScript thì dài dòng nhưng cấu trúc của nó khá đơn giản. Tất cả các hàm chính chúng ta cần phải tạo được đưa vào hàm `init`:

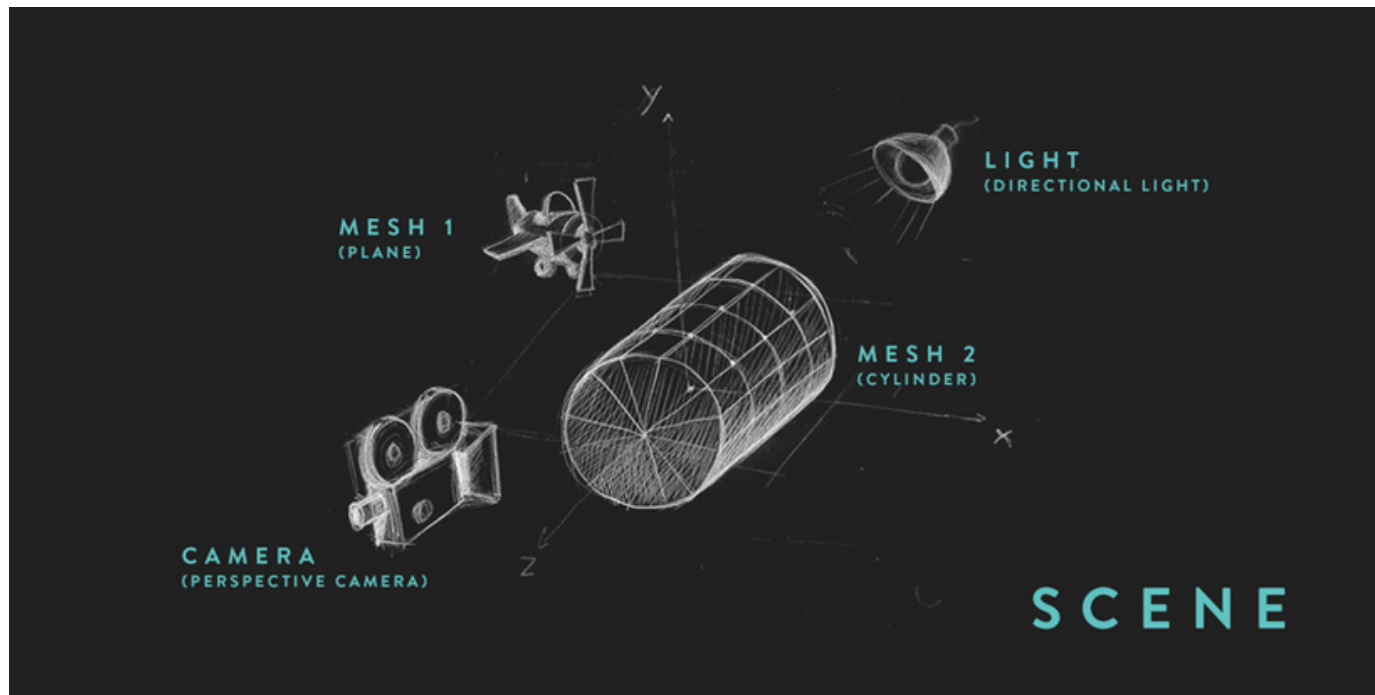
```
window.addEventListener('load', init, false);  
  
function init() {  
  // cài đặt scene, camera, và renderer  
  createScene();  
  
  // thêm ánh sáng  
  createLights();  
  
  // thêm các đối tượng  
  createPlane();  
  createSea();  
  createSky();  
}
```

```
// bắt đầu vòng lặp cập nhật vị trí các đối tượng  
// và render scene trong mỗi khung hình  
loop();  
}
```

## Dựng scene

Để tạo một project `three.js`, chúng ta cần tối thiểu những thứ sau:

1. Một `scene` (khung cảnh): giống như sân khấu để đặt các đối tượng lên
2. Một `camera`: ở đây chúng ta dùng `camera` theo góc nhìn, cũng có thể dùng `camera` trực giao
3. Một `renderer` (bộ dựng hình): hiển thị toàn bộ `scene` sử dụng `WebGL`.
4. Một hoặc nhiều đối tượng để `render`, trong trường hợp của chúng ta, cần tạo ra một chiếc máy bay, bãi biển và bầu trời (một vài đám mây)
5. Một hoặc nhiều nguồn sáng: có các loại nguồn sáng khác nhau sẵn có. Trong project này, chúng ta sẽ sử dụng chủ yếu nguồn sáng bán cầu cho môi trường và nguồn sáng có hướng cho bóng đổ.



scene, camera, and renderer được tạo trong hàm **createScene**:

```
var scene,  
    camera, fieldOfView, aspectRatio, nearPlane, farPlane, HEIGHT, WIDTH,  
    renderer, container;  
  
function createScene() {  
    // Lấy ra width và height của màn hình,  
    // dùng để cài đặt tỉ lệ khung hình (aspect ratio) cho camera  
    // và size của renderer.  
    HEIGHT = window.innerHeight;  
    WIDTH = window.innerWidth;
```

```
// Tạo scene
scene = new THREE.Scene();

// Thêm hiệu ứng sương mù vào scene, cùng màu với
// màu nền đã style trước đó
scene.fog = new THREE.Fog(0xf7d9aa, 100, 950);

// Tạo camera
aspectRatio = WIDTH / HEIGHT;
fieldOfView = 60;
nearPlane = 1;
farPlane = 10000;
camera = new THREE.PerspectiveCamera(
    fieldOfView,
    aspectRatio,
    nearPlane,
    farPlane
);

// Đặt vị trí cho camera
camera.position.x = 0;
camera.position.z = 200;
camera.position.y = 100;

// Tạo renderer
renderer = new THREE.WebGLRenderer({
    // Cho phép trong suốt để hiển thị màu nền
    // đã định nghĩa trong CSS
    alpha: true,

    // Bật khử răng cưa; hiệu năng sẽ giảm
    // nhưng sẽ ổn thôi vì project này ít đối tượng
});
```



```
        antialias: true
    });

    // Xác định kích cỡ của renderer; trong trường hợp này,
    // là full toàn màn hình
    renderer.setSize(WIDTH, HEIGHT);

    // Cho phép render bóng đổ
    renderer.shadowMap.enabled = true;

    // Thêm DOM của renderer vào
    // container ta đã tạo trong HTML
    container = document.getElementById('world');
    container.appendChild(renderer.domElement);

    // Nếu người dùng resize trình duyệt
    // cần cập nhật lại camera và size của renderer
    window.addEventListener('resize', handleWindowResize, false);
}
```

Khi kích thước màn hình thay đổi, chúng ta cần cập nhật kích thước `renderer` và tỉ lệ khung hình của `camera`:

```
function handleWindowResize() {
    // cập nhật lại kích thước của renderer và camera
    HEIGHT = window.innerHeight;
    WIDTH = window.innerWidth;
    renderer.setSize(WIDTH, HEIGHT);
    camera.aspect = WIDTH / HEIGHT;
```

```
camera.updateProjectionMatrix();  
}
```

## Ánh sáng

Ánh sáng là một trong những phần khó nhất khi vẽ một `scene`. Ánh sáng tạo nên cảm xúc cho toàn bộ khung cảnh và cần được xác định một cách cẩn thận. Nhưng tại bước này, chúng ta sẽ chỉ làm ánh sáng vừa đủ tốt để nhìn thấy các đối tượng.

```
var hemisphereLight, shadowLight;  
  
function createLights() {  
    // Nguồn sáng bán cầu là loại có màu tô chuyển (gradient)  
    // tham số đầu tiên là màu trời, thứ 2 là màu đất,  
    // thứ 3 là cường độ ánh sáng  
    hemisphereLight = new THREE.HemisphereLight(0xaaaaaa, 0x000000, .9)  
  
    // Nguồn sáng có hướng tỏa ra từ 1 vị trí nhất định  
    // Nó giống như mặt trời, nghĩa là các tia được tạo ra song song với nh  
    shadowLight = new THREE.DirectionalLight(0xffffff, .9);  
  
    // Đặt vị trí cho nguồn sáng  
    shadowLight.position.set(150, 350, 350);  
  
    // Cho phép phủ bóng  
    shadowLight.castShadow = true;  
  
    // cài đặt vùng nhìn thấy của bóng đổ  
    shadowLight.shadow.camera.left = -400;
```

```
shadowLight.shadow.camera.right = 400;
shadowLight.shadow.camera.top = 400;
shadowLight.shadow.camera.bottom = -400;
shadowLight.shadow.camera.near = 1;
shadowLight.shadow.camera.far = 1000;

// cài đặt độ phân giải của bóng đổ; càng cao càng đẹp,
// nhưng cũng càng nặng nề hơn
shadowLight.shadow.mapSize.width = 2048;
shadowLight.shadow.mapSize.height = 2048;

// thêm vào scene để kích hoạt
scene.add(hemisphereLight);
scene.add(shadowLight);
}
```

Như bạn thấy, rất nhiều thông số được sử dụng để tạo ra ánh sáng. Đừng ngần ngại thử nghiệm với màu sắc, cường độ và số lượng nguồn sáng; bạn sẽ học được cách điều chỉnh chúng cho các nhu cầu khác nhau.

## Tạo một đối tượng với **Three.js**

**Three.js** đã có một số lượng lớn các đối tượng nguyên thủy sẵn có như khối lập phương, hình cầu, hình xuyên, hình trụ và mặt phẳng.

Đối với project của chúng ta, tất cả các đối tượng ta tạo ra chỉ đơn giản là sự kết hợp các đối tượng nguyên thủy. Điều đó sẽ giúp chúng ta khỏi phải mô hình hóa các đối tượng bằng một phần mềm 3D.

## Biển hình trụ

Chúng ta bắt đầu bằng cách tạo ra biển vì nó là đối tượng dễ tạo nhất. Để giữ cho mọi thứ đơn giản, biển được minh họa như một hình trụ màu xanh đặt ở dưới cùng màn hình. Chúng ta sẽ tìm hiểu làm thế nào để tinh chỉnh hình khối này sau, để biển nhìn hấp dẫn hơn và sóng trông thực tế hơn.

```
// Định nghĩa đối tượng Sea
Sea = function(){

    // tạo khối hình trụ
    // các tham số:
    // bán kính mặt trên, bán kính mặt đáy, chiều cao, số lượng phân khúc t
    var geom = new THREE.CylinderGeometry(600,600,800,40,10);

    // xoay trên trục X
    geom.applyMatrix(new THREE.Matrix4().makeRotationX(-Math.PI/2));

    // tạo chất liệu
    var mat = new THREE.MeshPhongMaterial({
        color:Colors.blue,
        transparent:true,
        opacity:.6,
```

```
        shading:THREE.FlatShading,
    });

    // Để tạo một đối tượng trong Three.js, ta phải tạo ra một lưới (mesh)
    // là sự kết hợp của một hình khối và chất liệu
    this.mesh = new THREE.Mesh(geom, mat);

    // Cho phép bóng đổ trên bề mặt
    this.mesh.receiveShadow = true;
}

// Khởi tạo và thêm vào scene

var sea;

function createSea(){
    sea = new Sea();

    // đặt vị trí phía dưới scene
    sea.mesh.position.y = -600;

    // thêm lưới này vào scene
    scene.add(sea.mesh);
}
```

Hãy tóm tắt những gì chúng ta cần để tạo ra một đối tượng. Chúng ta cần phải

1. tạo khối hình học ( geometry )

2. tạo chất liệu ( `material` )

3. chuyển chúng vào lưới ( `mesh` )

4. thêm lưới vào `scene`

Với những bước cơ bản này, chúng ta có thể tạo ra nhiều loại đối tượng nguyên thủy khác nhau. Bây giờ, nếu kết hợp chúng, ta có thể tạo ra các hình dạng phức tạp hơn nhiều. Trong các bước sau đây chúng ta tìm hiểu làm thế nào để làm điều đó.

## **Kết hợp các khối đơn giản để tạo một hình thù phức tạp**

Những đám mây phức tạp hơn một chút, vì chúng là các khối lập phương được lắp ráp một cách ngẫu nhiên.



CUBE + CUBE + CUBE + CUBE = ... CLOUD?

```
Cloud = function(){  
  // Tạo một container rỗng để chứa các phần của đám mây  
  this.mesh = new THREE.Object3D();  
  
  // tạo khối lập phương;  
  // khối này sẽ được nhân ra để tạo ra các đám mây  
  var geom = new THREE.BoxGeometry(20,20,20);  
  
  // tạo chất liệu; chất liệu màu trắng là đủ  
  var mat = new THREE.MeshPhongMaterial({  
    color:Colors.white,  
  });
```

```

// nhân hình khối lên một số ngẫu nhiên lần
var nBlocs = 3+Math.floor(Math.random()*3);
for (var i=0; i<nBlocs; i++){

    // tạo lưới từ hình khối
    var m = new THREE.Mesh(geom, mat);

    // đặt vị trí và góc quay của mỗi khối ngẫu nhiên
    m.position.x = i*15;
    m.position.y = Math.random()*10;
    m.position.z = Math.random()*10;
    m.rotation.z = Math.random()*Math.PI*2;
    m.rotation.y = Math.random()*Math.PI*2;

    // đặt kích thước của khối ngẫu nhiên
    var s = .1 + Math.random()*0.9;
    m.scale.set(s,s,s);

    // cho phép mỗi khối phủ và nhận bóng đổ
    m.castShadow = true;
    m.receiveShadow = true;

    // thêm khối vào container
    this.mesh.add(m);
}
}

```

Bây giờ chúng ta đã có một đám mây, chúng ta sẽ sử dụng nó để tạo ra bầu trời bằng cách sao chép và đặt nó ở vị trí ngẫu nhiên quanh trục z:



```

// Định nghĩa đối tượng Sky
Sky = function(){
    // Tạo container rỗng
    this.mesh = new THREE.Object3D();

    // chọn số lượng đám mây rải rác trên bầu trời
    this.nClouds = 20;

    // Để phân phối các đám mây một cách nhất quán,
    // chúng ta phải đặt chúng theo một góc thống nhất
    var stepAngle = Math.PI*2 / this.nClouds;

    // tạo các đám mây
    for(var i=0; i<this.nClouds; i++){
        var c = new Cloud();

        // đặt góc xoay và vị trí của mỗi đám mây;
        // chúng ta sẽ dùng chút lượng giác
        var a = stepAngle*i; // đây là góc của đám mây
        var h = 750 + Math.random()*200; // đây là khoảng cách giữa trung d

        // Lượng giác!!! Hi vọng bạn vẫn còn nhớ chút Toán học :)
        // nếu không thì:
        // đơn giản là chuyển đổi tọa độ cực (góc, khoảng cách) sang tọa độ
        c.mesh.position.y = Math.sin(a)*h;
        c.mesh.position.x = Math.cos(a)*h;

        // xoay đám mây theo vị trí của nó
        c.mesh.rotation.z = a + Math.PI/2;

        // để kết quả tốt hơn, ta đặt các đám mây

```

```
// ở độ sâu ngẫu nhiên trong scene
c.mesh.position.z = -400-Math.random()*400;

// và cả độ phóng ngẫu nhiên cho mỗi đám mây
var s = 1+Math.random()*2;
c.mesh.scale.set(s,s,s);

// đừng quên thêm lưới vào container
this.mesh.add(c.mesh);
}
}

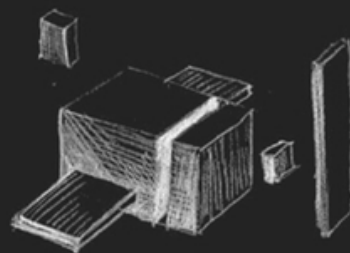
// Bây giờ ta khởi tạo bầu trời và đẩy tâm của nó
// về phía dưới màn hình một chút

var sky;

function createSky(){
    sky = new Sky();
    sky.mesh.position.y = -600;
    scene.add(sky.mesh);
}
```

## Thậm chí còn phức tạp hơn: Tạo máy bay

Tin xấu là code để tạo máy bay thì hơi dài dòng và phức tạp hơn. Nhưng tin tốt là chúng ta đã học mọi thứ chúng ta cần biết để làm điều đó! Tất cả về việc kết hợp và đóng gói các hình dạng.



## A SIMPLE PLANE, MADE OF CUBES

```
var AirPlane = function() {  
  
    this.mesh = new THREE.Object3D();  
  
    // Tạo cabin  
    var geomCockpit = new THREE.BoxGeometry(60,50,50,1,1,1);  
    var matCockpit = new THREE.MeshPhongMaterial({color:Colors.red, shading  
    var cockpit = new THREE.Mesh(geomCockpit, matCockpit);  
    cockpit.castShadow = true;  
    cockpit.receiveShadow = true;  
    this.mesh.add(cockpit);  
  
    // Tạo động cơ
```

```

var geomEngine = new THREE.BoxGeometry(20,50,50,1,1,1);
var matEngine = new THREE.MeshPhongMaterial({color:Colors.white, shininess:100});
var engine = new THREE.Mesh(geomEngine, matEngine);
engine.position.x = 40;
engine.castShadow = true;
engine.receiveShadow = true;
this.mesh.add(engine);

// Tạo đuôi
var geomTailPlane = new THREE.BoxGeometry(15,20,5,1,1,1);
var matTailPlane = new THREE.MeshPhongMaterial({color:Colors.red, shininess:100});
var tailPlane = new THREE.Mesh(geomTailPlane, matTailPlane);
tailPlane.position.set(-35,25,0);
tailPlane.castShadow = true;
tailPlane.receiveShadow = true;
this.mesh.add(tailPlane);

// Tạo cánh
var geomSideWing = new THREE.BoxGeometry(40,8,150,1,1,1);
var matSideWing = new THREE.MeshPhongMaterial({color:Colors.red, shininess:100});
var sideWing = new THREE.Mesh(geomSideWing, matSideWing);
sideWing.castShadow = true;
sideWing.receiveShadow = true;
this.mesh.add(sideWing);

// phần quạt
var geomPropeller = new THREE.BoxGeometry(20,10,10,1,1,1);
var matPropeller = new THREE.MeshPhongMaterial({color:Colors.brown, shininess:100});
this.propeller = new THREE.Mesh(geomPropeller, matPropeller);
this.propeller.castShadow = true;
this.propeller.receiveShadow = true;

```

```
// cánh quạt
var geomBlade = new THREE.BoxGeometry(1,100,20,1,1,1);
var matBlade = new THREE.MeshPhongMaterial({color:Colors.brownDark, sha

var blade = new THREE.Mesh(geomBlade, matBlade);
blade.position.set(8,0,0);
blade.castShadow = true;
blade.receiveShadow = true;
this.propeller.add(blade);
this.propeller.position.set(50,0,0);
this.mesh.add(this.propeller);
};
```

Bây giờ, chúng ta có thể khởi tạo và thêm vào `scene` :

```
var airplane;

function createPlane(){
    airplane = new AirPlane();
    airplane.mesh.scale.set(.25,.25,.25);
    airplane.mesh.position.y = 100;
    scene.add(airplane.mesh);
}
```

## Rendering

Chúng ta đã tạo ra một vài đối tượng và cho vào `scene` . Nhưng nếu bạn cố gắng để chạy nó lên, bạn không thể nhìn thấy bất cứ thứ gì! Đó là vì chúng ta chưa `render` . Có thể làm điều đó bằng dòng này:

```
renderer.render(scene, camera);
```

## Chuyển động

Hãy mang lại sức sống cho khung cảnh của chúng ta bằng cách làm quay cánh quạt của máy bay, xoay biển và những đám mây.

Để làm điều này, chúng ta cần một vòng lặp vô hạn:

```
function loop(){
  // Xoay cánh quạt, biển và bầu trời
  airplane.propeller.rotation.x += 0.3;
  sea.mesh.rotation.z += .005;
  sky.mesh.rotation.z += .01;

  // render the scene
  renderer.render(scene, camera);

  // gọi hàm lặp lại
  requestAnimationFrame(loop);
}
```

Như bạn thấy, chúng ta đã chuyển lệnh gọi hàm `render` vào trong vòng lặp. Vì mỗi thay đổi với các đối tượng cần phải được `render` lại.

## Thêm tương tác: Chạy theo trỏ chuột

Tại thời điểm này, có thể thấy máy bay của chúng ta được đặt ở trung tâm của khung cảnh. Những gì chúng ta muốn, là làm nó chạy theo các chuyển động của chuột.

Một khi trang được load xong, chúng ta thêm `listener` để kiểm tra xem chuột có di chuyển không.

Sửa đổi hàm `init` như sau:

```
function init(event){
  createScene();
  createLights();
  createPlane();
  createSea();
  createSky();

  //thêm listener
  document.addEventListener('mousemove', handleMouseMove, false);

  loop();
}
```

Ngoài ra, tạo một hàm mới để xử lý sự kiện chuột di chuyển:

```
var mousePos={x:0, y:0};

// xử lý sự kiện mousemove

function handleMouseMove(event) {

    // ở đây chúng ta đang chuyển đổi giá trị vị trí chuột nhận được
    // đến một giá trị chuẩn hoá giữa -1 và 1;
    // đây là công thức cho trục ngang:

    var tx = -1 + (event.clientX / WIDTH)*2;

    // với trục dọc, chúng ta cần phải đảo ngược công thức
    // vì trục tung 2D đi theo hướng ngược lại trục tung 3D

    var ty = 1 - (event.clientY / HEIGHT)*2;
    mousePos = {x:tx, y:ty};

}
```

Bây giờ chúng ta có vị trí x và y của chuột đã chuẩn hoá, chúng ta có thể di chuyển máy bay chính xác.

Cần sửa vòng lặp và thêm hàm mới để cập nhật máy bay:



```

function loop(){
    sea.mesh.rotation.z += .005;
    sky.mesh.rotation.z += .01;

    // cập nhật máy bay trong mỗi khung hình
    updatePlane();

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

function updatePlane(){

    // di chuyển máy bay trong khoảng -100 tới 100 trên trục ngang,
    // và từ 25 đến 175 trên trục dọc,
    // tùy thuộc vào vị trí chuột trong khoảng từ -1 đến 1 trên cả hai trục
    // để làm điều đó chúng ta sử dụng một hàm chuẩn hoá (bên dưới)

    var targetX = normalize(mousePos.x, -1, 1, -100, 100);
    var targetY = normalize(mousePos.y, -1, 1, 25, 175);

    // cập nhật vị trí máy bay
    airplane.mesh.position.y = targetY;
    airplane.mesh.position.x = targetX;
    airplane.propeller.rotation.x += 0.3;
}

function normalize(v,vmin,vmax,tmin, tmax){

    var nv = Math.max(Math.min(v,vmax), vmin);
    var dv = vmax-vmin;

```

```
var pc = (nv-vmin)/dv;  
var dt = tmax-tmin;  
var tv = tmin + (pc*dt);  
return tv;
```

```
}
```

Xin chúc mừng, bạn đã làm cho máy bay di chuyển theo chuột! Hãy xem những gì chúng ta đã làm đến nay: [Demo của phần 1](#).

## (Sắp) Xong!

Như bạn thấy đó, `Three.js` rất hữu ích trong việc tạo ra nội dung WebGL. Bạn không cần phải biết quá nhiều để thiết lập một `scene` và `render` vài đối tượng tùy chỉnh. Cho đến bây giờ bạn đã học được một số khái niệm cơ bản và bạn đã có thể bắt đầu tinh chỉnh một vài thông số như cường độ ánh sáng, màu sắc sương mù và kích thước của các đối tượng. Bạn thậm chí còn quen với việc tạo ra các đối tượng nữa nhỉ?

*Chúng ta vừa hoàn thành xong phần 1 bài hướng dẫn "Tạo 1 cảnh động 3D đơn giản trên web với Three.js". Bài post được dịch từ bài viết gốc của tác giả [Karim Maaloul](#) tại địa chỉ <http://tympanus.net/codrops/2016/04/26/the-aviator-animating-basic-3d-scene-threejs>. Cảm ơn các bạn đã theo dõi!*

Have problems with [Game, 3D or Three.js](#)? [Ask on Viblo »](#)

## More from Tan Nguyen

### Mạng nơ-ron tích chập (P2-hết)

Tan Nguyen

 2840  2  2  3

### Mạng nơ-ron tích chập (P1)

Tan Nguyen

 5286  5  1  5

### Debug code PHP với Xdebug và Subl...

Tan Nguyen

 3475  5  1  2

### Làm việc với code của người khác

Tan Nguyen

 481  1  0  0

## Comments

 Login to comment

 **Vu Vuong** @hatgaolangta05293

Jul 19th, 2016 12:14 AM

Anh cho em hỏi: Hiện tại em load 1 tấm hình 3D-VR(hình chụp 360). Em muốn ghi 1 đoạn text vào 1 vị trí cố định trên tấm hình. Anh có thể cho em biết nên sử dụng thư viện gì không ạ ?

^ 0 v | Reply share ↗

 **Tan Nguyen** @tannm13

Jul 19th, 2016 12:29 AM

Mình chưa thử bao h nhưng bạn đã load hình bằng Three.js thì muốn vẽ thêm cái j thì dùng luôn Three.js mà vẽ thôi tham khảo link: <https://stemkoski.github.io/Three.js/Sprite-Text-Labels.html>

^ 0 v | Reply share ↗

 **Vu Vuong** @hatgaolangta05293

Jul 19th, 2016 7:32 PM

tk's anh nha

^ 0 v | Reply share ↗

 **Phuong Nguyen** @phuong2507

Sep 8th, 2016 5:03 AM

Chào bạn Tan Nguyen Tôi muốn mời bạn tham gia thiết kế 1 website có thể view được file 3D OBJ

Không biết bạn có thời gian cho việc này không ạ

nếu bạn hứng thú, vui lòng liên hệ với tôi: 0986822317 - Phương hoặc skype: phuong2507

rất vui được hợp tác làm việc cùng bạn

^ 0 v | Reply share ↗

## RESOURCES

[Posts](#)

[Questions](#)

[Videos](#)

[Tags](#)


[Authors](#)


[Tools](#)

## LINKS

 [Facebook](#)

 [GitHub](#)

 [Browser extension](#)

 [Atom plugin](#)

## MOBILE APP



---

© 2018 **Viblo**. All rights reserved.

[Feedback](#)[Help](#)[FAQs](#)[Terms](#)