INSTITUTO TECNOLÓGICO DE CHIHUAHUA II

Graficacion

Profesor: Alonso Salcido

TAREA: “The Aviator”

Agosto-Diciembre 2017

Edith Ortiz Martínez #13550419

INGENIERÍA EN SISTEMAS COMPUTACIONALES

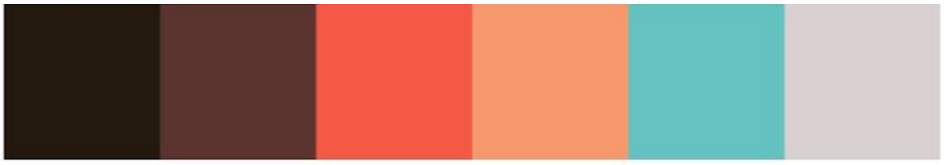
HTML & CSS

Comenzamos importando nustra libreria en nuestro HTML 5, con esta linea de codigo:

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

JAVASCRIPT

Implementamos nuestra paleta de colores:



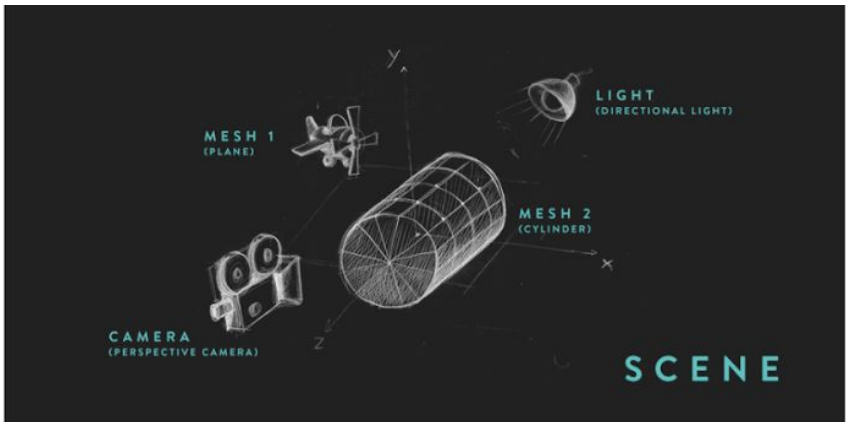
Antes de empezar con la escena, elegimos a siguiente paleta de colores:

```
//COLORS
var Colors = {
    red:0x212F3C,
    white:0xC0392B  ,
    brown:0x27AE60,
    brownDark:0x23190f,
    pink:0xF5986E,
    yellow:0xf4ce93,
    blue:0xDC7633,

};

```

CONFIGURANDO LA ESCENA

Para crear un projecto de Three.js lo que se necesita es lo siguiente:

1. Una escena
2. Una camara
3. Un renderer
4. Uno o más objetos
5. Una o más luces



La escena, la camara y el renderer son creados en la funcion “createScene” de esta manera:

```
var scene,
    camera, fieldOfView, aspectRatio, nearPlane, farPlane, HEIGHT, WIDTH,
    renderer, container;

function createScene() {
  // Get the width and the height of the screen,
  // use them to set up the aspect ratio of the camera
  // and the size of the renderer.
  HEIGHT = window.innerHeight;
  WIDTH = window.innerWidth;

  // Create the scene
  scene = new THREE.Scene();

  // Add a fog effect to the scene; same color as the
  // background color used in the style sheet
  scene.fog = new THREE.Fog(0xf7d9aa, 100, 950);

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

```
  // Set the position of the camera
  camera.position.x = 0;
  camera.position.z = 200;
  camera.position.y = 100;

  // Create the renderer
  renderer = new THREE.WebGLRenderer({
    // Allow transparency to show the gradient background
    // we defined in the CSS
    alpha: true,

    // Activate the anti-aliasing; this is less performant,
    // but, as our project is low-poly based, it should be fine :)
    antialias: true
  });

  // Define the size of the renderer; in this case,
  // it will fill the entire screen
  renderer.setSize(WIDTH, HEIGHT);

  // Enable shadow rendering
  renderer.shadowMap.enabled = true;

  // Add the DOM element of the renderer to the
  // container we created in the HTML
  container = document.getElementById('world');
  container.appendChild(renderer.domElement);

  // Listen to the screen: if the user resizes it
  // we have to update the camera and the renderer size
```

```
  // Listen to the screen: if the user resizes it
  // we have to update the camera and the renderer size
  window.addEventListener('resize', handleWindowResize, false);
}
```

Como el tama;o de la pantalla cambia, necesitamos especificar el tama;o del renderer y el aspect ramtio de la camara:

```
function handleWindowResize() {
  // update height and width of the renderer and the camera
  HEIGHT = window.innerHeight;
  WIDTH = window.innerWidth;
  renderer.setSize(WIDTH, HEIGHT);
  camera.aspect = WIDTH / HEIGHT;
  camera.updateProjectionMatrix();
}
```

LAS LUCES

La iluminacion es sin duda una de las partes más complicadas a la hora de configurar una escena. Las luces establecerán el ambiente de toda la escena y deben determinarse cuidadosamente. En este paso del proyecto, intentaremos hacer que el rayo sea lo suficientemente bueno para hacer que los objetos sean visibles:

```
var hemisphereLight, shadowLight;

function createLights() {
  // A hemisphere light is a gradient colored light;
  // the first parameter is the sky color, the second parameter is the groun
  // the third parameter is the intensity of the light
  hemisphereLight = new THREE.HemisphereLight(0xaaaaaa,0x000000, .9)

  // A directional light shines from a specific direction.
  // It acts like the sun, that means that all the rays produced are paralle
  shadowLight = new THREE.DirectionalLight(0xffffff, .9);

  // Set the direction of the light
  shadowLight.position.set(150, 350, 350);

  // Allow shadow casting
  shadowLight.castShadow = true;

  // define the visible area of the projected shadow
  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;

  // define the resolution of the shadow; the higher the better,
  // but also the more expensive and less performant
  shadowLight.shadow.mapSize.width = 2048;
  shadowLight.shadow.mapSize.height = 2048;

  // to activate the lights, just add them to the scene
  scene.add(hemisphereLight);
  scene.add(shadowLight);
}
```

Como puede ver aquí, se usan muchos parámetros para crear las luces. No dude en experimentar con los colores, las intensidades y la cantidad de luces; descubrirá ambientes y ambientes interesantes para su escena y tendrá una idea de cómo ajustarlos para sus necesidades.

UN SIMPLE CILINDRO PARA EL OCEANO

Comencemos creando el mar, ya que es el objeto más fácil con el que tenemos que lidiar. Para simplificar las cosas por el momento, ilustraremos el mar como un simple cilindro azul colocado en la parte inferior de la pantalla. Más adelante profundizaremos en algunos detalles sobre cómo refinar esta forma.
A continuación, hagamos que el mar se vea un poco más atractivo y las olas más realistas:

```
// First let's define a Sea object :
Sea = function(){

  // create the geometry (shape) of the cylinder;
  // the parameters are:
  // radius top, radius bottom, height, number of segments on the radius
  var geom = new THREE.CylinderGeometry(600,600,800,40,10);

  // rotate the geometry on the x axis
  geom.applyMatrix(new THREE.Matrix4().makeRotationX(-Math.PI/2));

  // create the material
  var mat = new THREE.MeshPhongMaterial({
    color:Colors.blue,
    transparent:true,
    opacity:.6,
    shading:THREE.FlatShading,
  });

  // To create an object in Three.js, we have to create a mesh
  // which is a combination of a geometry and some material
  this.mesh = new THREE.Mesh(geom, mat);

  // Allow the sea to receive shadows
  this.mesh.receiveShadow = true;
}

// Instantiate the sea and add it to the scene:
```

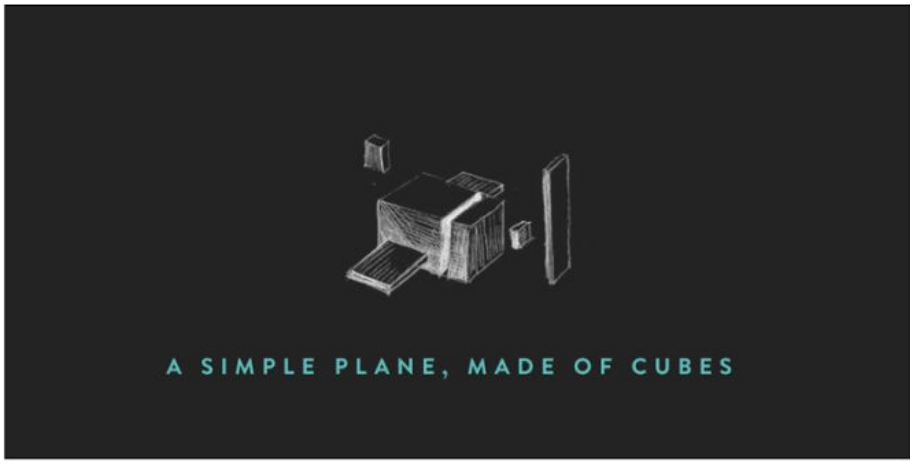
```
var sea;

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

  // push it a little bit at the bottom of the scene
  sea.mesh.position.y = -600;

  // add the mesh of the sea to the scene
  scene.add(sea.mesh);
}
```

Resumamos lo que necesitamos para crear un objeto.
Necesitamos:
-crear una geometría
-crear un material
-pasarlos en un mesh
-agregar el mesh a nuestra escena

Con estos pasos básicos, podemos crear muchos tipos diferentes de objetos primitivos. Ahora, si los combinamos, podemos crear formas mucho más complejas. En los siguientes pasos, aprenderemos cómo hacer eso con precisión.

CREANDO LA AERONAVE

La mala noticia es que el código para crear el avión es un poco más largo y complejo. ¡Pero la buena noticia es que ya aprendimos todo lo que necesitamos saber para poder hacerlo! Se trata de combinar y encapsular formas.



ANIMACION

```
function loop(){
  // Rotate the propeller, the sea and the sky
  airplane.propeller.rotation.x += 0.3;
  sea.mesh.rotation.z += .005;
  sky.mesh.rotation.z += .01;

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

  // call the loop function again
  requestAnimationFrame(loop);
}
```

Como puede ver, hemos trasladado la llamada al método de renderizado a la función de bucle. Esto se debe a que cada cambio que realizamos en un objeto debe volver a representarse.

SIGUIENDO EL MOUSE: AGREGANDO INTERACCION

En este momento, podemos ver nuestro avión colocado en el centro de la escena. Lo que queremos lograr a continuación es hacer que siga los movimientos del mouse.
Una vez que se carga el documento, necesitamos agregar un detector al documento para verificar si el mouse se está moviendo.
Para eso, modificaremos la función init de la siguiente manera:

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

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

  loop();
}
```

Además, crearemos una nueva función para manejar el evento mousemove:

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

// now handle the mousemove event

function handleMouseMove(event) {

  // here we are converting the mouse position value received
  // to a normalized value varying between -1 and 1;
  // this is the formula for the horizontal axis:

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

  // for the vertical axis, we need to inverse the formula
  // because the 2D y-axis goes the opposite direction of the 3D y-axis

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

}
```

AGREGAMOS UN FEATURE EXTRA (UN TOROIDE TRIDIMENSIONAL)

Se agrega de manera que de la impresion de que es un spol en forma de toroide:

```
}
Sun = function(){

  var geom = new THREE.TorusGeometry(100,30,160,100);

  // create the material
  var mat = new THREE.MeshPhongMaterial({
    color:Colors.ySun,
  });

  // To create an object in Three.js, we have to create a mesh
  // which is a combination of a geometry and some material
  this.mesh = new THREE.Mesh(geom, mat);
}
```

EL RESULTADO FINAL