INSTITUTO TECNOLÓGICO DE CHIHUAHUA II

Graficación

Profesor: Alonso Salcido

EXAMEN: “Star Wars Scene”

Agosto-Diciembre 2017

Heber Jhazeel Torres Aragón
Edith Ortiz Martinez
Sergio Franco Mendoza
Adelaido Cano Lozano

INGENIERÍA EN SISTEMAS COMPUTACIONALES

INTRODUCCIÓN

El siguiente proyecto se basa en crear una escena en javascript implementando Html5, utilizando primitivas, meshes, renderizado, y básicamente en este proyecto, se desarrollan todos los conocimientos obtenidos en la clase durante todo el curso.

Ya una vez habiendo explicado las herramientas o conocimientos que se implementaran en el proyecto, podemos dar una pequeña introducción sobre lo que se va a presentar a continuación.

Se trata de una escena con la temática de Star Wars y una nave tipo destructor estelar del imperio flotando en la escena, con un fondo de estrellas y música de Star Wars.

INTRODUCCION DE COMO CREAR FUNCIONES DE MOVIMIENTO:
AGREGAR UN MODELO

Vertices_front=[[x, y, z]]; Se puede notar que cada vector tiene una profundidad, la variable z, viene siendo la profundidad de cada vector.
Este modelo es el que se carga mediante una función y para poder hacer este tipo de cargas se necesitan variables accesibles en todas partes:

- Renderer
- Scene
- Camera
- Vertices
- Figura

CREANDO LAS FUNCIONALIDADES

- inicio()

Inicia todas las cargas de los elementos gráficos.

- animar()

Función que se repite para iniciar la animación.

- cargar_modelo()

Construye los modelos o las especificaciones de los modelos.

- render_modelo()

Renderiza y modifica elementos gráficos para transformar la geometría.

3 transformaciones básicas:
Traslacion
Rotacion
Escalado

Que es renderizar?
Proceso de generar una imagen o vídeo mediante el cálculo de iluminación partiendo de un modelo en 3D.

USANDO Plugin OrbitControls.js :

Nos permite interactuar con el mouse, o bien, tomar el control mediante el mouse con nuestro elemento gráfico.

```
    // agregamos todo el escenario y la cámara al render
    controls = new THREE.OrbitControls(Camara, Render.domElement);
}
```

-INICIANDO CON EL CÓDIGO
FUNCIÓN INICIO

```
/***************************** inicio *****************/
function inicio() {
    //Tamaño del render(resultado)
    Render.setSize(Ancho, Alto);
    //Se agrega el render al documento html
    document.getElementById('render').appendChild(Render.domElement);
    //Acercamos la cámara en z es profundidad para ver el punto
    Camara.position.z = 250;
    Camara.position.y = 50;
    Camara.position.x = 0;

    //agregando la cámara al escenario
    Escenario.add(Camara);
    // agregamos todo el escenario y la cámara al render
    controls = new THREE.OrbitControls(Camara, Render.domElement);
}
```

En esta parte del código
-Se configura el tamaño del render
-Se agrega el render al documento Html
-Se configura la cámara con su profundidad en z
-Se agrega la cámara a la escena
-Se agrega la escena y la cámara al render

LLAMANDO A LAS FUNCIONES
CARGAR_CUBO

```
/************************llamado a las funcion *****************/

function cargar_cubo() {
    geometriaCubo = new THREE.CubeGeometry(2000, 2000, 2000);
    var ImgTextura = [];

    //le ponemos a cada una de las caras una textura(iamgen por dentro y f
    ImgTextura.push(new THREE.MeshBasicMaterial({ map: THREE.ImageUtils.lo
    ImgTextura.push(new THREE.MeshBasicMaterial({ map: THREE.ImageUtils.lo
    ImgTextura.push(new THREE.MeshBasicMaterial({ map: THREE.ImageUtils.lo
    ImgTextura.push(new THREE.MeshBasicMaterial({ map: THREE.ImageUtils.lo

    ImgTextura.push(new THREE.MeshBasicMaterial({ map: THREE.ImageUtils.lo

    ImgTextura.push(new THREE.MeshBasicMaterial({ map: THREE.ImageUtils.lo
    ImgTextura.push(new THREE.MeshBasicMaterial({ map: THREE.ImageUtils.lo

    var material = new THREE.MeshFaceMaterial(ImgTextura);

    mallaCubo = new THREE.Mesh(geometriaCubo, material);
    Escenario.add(mallaCubo);
    mallaCubo.position.set(0, 0, 0);
    mallaCubo.receiveShadow = true;

}
```

En esta parte de nuestro codigo se agrega la geometría, un cubo en 3D el cual tendrá una textura en cada una de sus caras, para esto se utilizan los “mesh”.
La textura a utilizar, es la de las estrellas para dar a nuestra escena la sensación de estar viajando en una galaxia.

FUNCION ANIMACIÓN

```
// cargar nuevos modelos
function animacion() {

    requestAnimationFrame(animacion);
    render_modelo();
    if (typeof ModeloFinal != "undefined") {
        tiempo = 0.01;
        distancia = 100;
        recorrido = distancia * tiempo;

        //botones con los que movemos el modelo
        obj_mov = ModeloFinal;
        if (teclado.pressed("down")) {
            obj_mov.position.z += recorrido;
            playSound();
        }
        if (teclado.pressed("up")) {
            obj_mov.position.z -= recorrido;
            playSound();
        }
        if (teclado.pressed("right")) {
            obj_mov.position.x += recorrido;
            playSound2();
        }
        if (teclado.pressed("left")) {
            obj_mov.position.x -= recorrido;
            playSound2();
        }
```

Hacemos llamar a una función llamada render_modelo();
En este render_modelo se agrega un renderizado especial sin el renderizado que está dentro del inicio.

```
function render_modelo() {
    controls.update();
    Render.render(Escenario, Camara);
}
```

Hasta esta parte ya se tiene la animacion requestAnimationFrame(animacion); la cual hace el ciclo de la animación.

CARGAR UN MODELO 3D EN BLENDER

El modelo 3D está constituido de 2 partes:
Material y Vértices

manejaremos el formato js para poder modificarlo y definirlo más rápido, para esto utilizamos modelos proporcionados por BLEND SWAP, de donde se puede generar algún modelo 3D a partir de alguna base.

como se trabaja con BLENDER??

Se carga un modelo con Blender y se instala un addons, plugin de THREE JS para poder exportar los modelos en 3D.

- Se carga un modelo en formato json de blender de esta manera:

  JsonMod3d = new THREE.JSONLoader();

- Posteriormente una vez que se agrega el cargador de modelos se dice que busque el modelo .js de esta manera:

  JsonMod3d.load("..js",AMod3d)

Y que ejecute una funcion para agregar al escenario.

Se le agrega una función a la gual se le agregan parámetros como la geometría y el material.

```
// cargamos los modelos dando direccion y parametros como tamaño, escala, sombras
var Modelo3D = new THREE.JSONLoader();
Modelo3D.load("benjob_vadertie.js", funcionAgregarModelo);

function funcionAgregarModelo(geometry, materials) {
    ModeloFinal = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial(materials));
    Escenario.add(ModeloFinal);
    ModeloFinal.scale.set(10, 10, 10);
    ModeloFinal.position.set(10, 50, 10);
    ModeloFinal.rotation.y = Math.PI;
    ModeloFinal.castShadow = true;
}

Modelo3D.load('benjob_tie.js', function (geometry, materials) {
    var ModeloFinal = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial(materials));
    ModeloFinal.scale.set(5, 5, 5);
    ModeloFinal.position.set(0, 0, -400);
    ModeloFinal.castShadow = true;
    Escenario.add(ModeloFinal);

});
Modelo3D.load('Falcon_Swap1.js', function (geometry, materials) {
    var ModeloFinal = new THREE.Mesh(geometry, new THREE.MeshFaceMaterial(materials));
    ModeloFinal.scale.set(10, 10, 10);
    ModeloFinal.position.set(100, 0, 200);
    ModeloFinal.castShadow = true;
    Escenario.add(ModeloFinal);

});
```

En las primeras dos lineas de este codigo, se carga el modelo 3D.

Luego se agrega una función AgregarModelo, dentro de esta función se cargan materiales y el modelo final, lo cual nos permite agregar al escenario el modelo final,

Después de agregar el modelo final podemos generar las transformaciones, escala, posición.

De esta manera tenemos nuestra nave en la escena, lo que sigue a continuación es la parte del movimiento de nuestra nave:

Para esto nos vamos a la parte de la animación , hacemos que el modelo, en este caso el obj sea el objeto final, para que haya movimiento del objeto.

```
// cargar nuevos modelos
function animacion() {

    requestAnimationFrame(animacion);
    render_modelo();
    if (typeof ModeloFinal != "undefined") {
        tiempo = 0.01;
        distancia = 100;
        recorrido = distancia * tiempo;

        //botones con los que movemos el modelo
        obj_mov = ModeloFinal;
        if (teclado.pressed("down")) {
            obj_mov.position.z += recorrido;
            playSound();
        }
```

En esta parte de nuestro codigo implementamos los botones con los que se moverá nuestro modelo:

```
//botones con los que movemos el modelo
obj_mov = ModeloFinal;
if (teclado.pressed("down")) {
    obj_mov.position.z += recorrido;
    playSound();
}
if (teclado.pressed("up")) {
    obj_mov.position.z -= recorrido;
    playSound();
}
if (teclado.pressed("right")) {
    obj_mov.position.x += recorrido;
    playSound2();
}
if (teclado.pressed("left")) {
    obj_mov.position.x -= recorrido;
    playSound2();
}
```

Y en la siguiente parte de nuestro codigo se implementan los botones con los que rota nuestro modelo:

```
        //rotaciones con botones
        tiempo_rotacion = 0.00009;
        distancia = 1000;
        recorrido_rotacion = distancia * tiempo_rotacion;
        if (teclado.pressed("z")) {
            obj_mov.rotation.z += recorrido_rotacion;
        }
        if (teclado.pressed("x")) {
            obj_mov.rotation.x += recorrido_rotacion;
        }
        if (teclado.pressed("c")) {
            obj_mov.rotation.y += recorrido_rotacion;
        }
        if (teclado.pressed("a")) {
            obj_mov.rotation.z -= recorrido_rotacion;
        }
        if (teclado.pressed("s")) {
            obj_mov.rotation.x -= recorrido_rotacion;
        }
        if (teclado.pressed("d")) {
            obj_mov.rotation.y -= recorrido_rotacion;
            console.log(controls);
        }

        controls.target.set(obj_mov.position.x, obj_mov.position.y, obj_mov.position.z);
    }
    if (typeof mallaCubo != "undefined") {
        mallaCubo.position.set(controlsBox.w, controlsBox.h, controlsBox.d);
    }

}
function render_modelo() {
    controls.update();
```

En la parte de la luz, se agrega lo que llamamos luz de sol, y se agrega a un escenario.

```
Luz();
//cargamos las sombras
function Luz() {
    var luz = new THREE.PointLight(0xffffff);
    luz.position.set(-100, 200, 100);
    Escenario.add(luz);
    //luz de ambiente
    var Luzambiente = new THREE.AmbientLight(0x000000);
    Escenario.add(Luzambiente);
    // más luz
    var sunlight = new THREE.DirectionalLight();
    sunlight.position.set(500, 500, -500);
    sunlight.intensity = 1.3;


    sunlight.castShadow = true;
    sunlight.shadowCameraVisible = true;

    sunlight.shadowCameraNear = 250;
    sunlight.shadowCameraFar = 20000;

    intensidad = 50;

    sunlight.shadowCameraLeft = -intensidad;
    sunlight.shadowCameraRight = intensidad;
    sunlight.shadowCameraTop = intensidad;
    sunlight.shadowCameraBottom = -intensidad;

    Escenario.add(sunlight);

}
```

La luz puede tener efecto sobre el material del modelo.

ModeloFinal.castShadow = true; : esta parte emite la sombra y la refleja en los elementos de la escena.
sunlight.castShadow = true; : complementos de la luz

sunlight.shadowCameraVisible = false; : complementos de la luz, cuando se activa en true se ve como las líneas se proyectan.

Pasos:
Se agrega esta linea de codigo al render:
Con esto se activa la compatibilidad del render para obtener sombras.

```
var Render = new THREE.WebGLRenderer(
Render.shadowMapEnabled = true;
```

Ahora se agrega a nuestro modelo el método de emitir sombras:

```
    ModeloFinal.rotation.y = Math.PI;
    ModeloFinal.castShadow = true;
}
```

Se localiza parte de la luz, la luz se divide por la proyección de las sombras, la más lejana y la más cercana, también se encuentran 2 métodos castShadow y se activa la propiedad de shadowCameraVisible, con esto se puede observar la sombra de la luz direccional.

```
sunlight.castShadow = true;
sunlight.shadowCameraVisible = true;

sunlight.shadowCameraNear = 250;
sunlight.shadowCameraFar = 20000;

intensidad = 50;

sunlight.shadowCameraLeft = -intensidad;
sunlight.shadowCameraRight = intensidad;
sunlight.shadowCameraTop = intensidad;
sunlight.shadowCameraBottom = -intensidad;

Escenario.add(sunlight);
```

Con la variable intensidad definimos cuánta parte de la sombra queremos proyectar, estas son las propiedades de la sombra, es este caso 50, emite una sombra más clara:

```
intensidad = 50;

sunlight.shadowCameraLeft = -intensidad;
sunlight.shadowCameraRight = intensidad;
sunlight.shadowCameraTop = intensidad;
sunlight.shadowCameraBottom = -intensidad;

Escenario.add(sunlight);
```

SONIDOS

En esta parte de nuestro index se cargan los sonidos que van a ser implementados en nuestra escena, de esta manera:

```
<head>
    <!--<audio id="space" src="sounds/star-wars-main-song.mp3" autostart="true"></audio>
    <audio id="cantina" src="sounds/star-wars-cantina-song.mp3" autostart="true"></audio>
    <audio id="rJedi" src="sounds/star-wars-return-song.mp3" autoplay="true"></audio>
    -->
    <audio id="flying1" src="sounds/flying1.wav" autostart="true"></audio>
    <audio id="flying2" src="sounds/flying2.wav" autostart="true"></audio>
    <title>Star Wars</title>
```

Luego se quiso incluir que al mover la nave con las teclas Up, Down, Left y Right del teclado, se escuchara cierto sonido de una nave, esto se llevó a cabo de la siguiente manera:
Con nuestras funciones playSound() obtenemos nuestros 2 sonidos

```
function playSound() {
    var sExt = document.getElementById("flying1");
    sExt.play();
    var mediaElement = document.getElementById("flying2");
    mediaElement.pause();
}

function playSound2() {
    var sInt = document.getElementById("flying2");
    sInt.play();
    var mediaElement = document.getElementById("flying1");
    mediaElement.pause();
}
```

Luego para poder activarlos cada que presionamos nuestras teclas, mandamos llamar a la función correspondiente dependiendo de la tecla que vayamos a presionar:

```
//botones con los que movemos el modelo
obj_mov = ModeloFinal;
if (teclado.pressed("down")) {
    obj_mov.position.z += recorrido;
    playSound();
}
if (teclado.pressed("up")) {
    obj_mov.position.z -= recorrido;
    playSound();
}
if (teclado.pressed("right")) {
    obj_mov.position.x += recorrido;
    playSound2();
}
if (teclado.pressed("left")) {
    obj_mov.position.x -= recorrido;
    playSound2();
}
```

Nuestra escena final, se ve entonces de la siguiente manera: