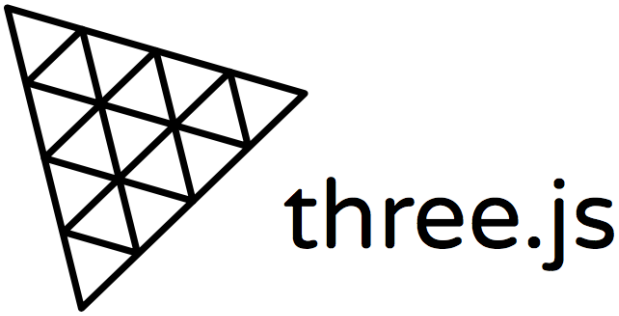
UNIVERSIDAD MAYOR DE SAN ANDRES
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

PROGRAMACIÓN MULTIMEDIAL

WebGL



PROYECTO: MODELADO LABORATORIO CARRERDA DE INFORMATICA CON THREE.JS

Estudiante: Horacio Andrés Flores Alberto

Correo Electrónico: foles4h@gmail.com

Teléfono: 76516641

Repositorio del Proyecto:
https://github.com/foles/ProyectoINF324

MODELADO LABORATORIO CARRERDA DE INFORMATICA CON THREE.JS

1.- DESCRIPCION DEL PROYECTO

El presente proyecto tiene como objetivo modelar uno de los Laboratorios de la Carrera de Informática de la Universidad Mayor de San Andrés. Se clasifico los componentes mas importantes de este laboratorio, como los ordenadores, monitores, teclados, escritorios, sillas y otros, para tratar la mayor similitud a este, para una buena experiencia del usuario. Se logro que el usuario pueda interactuar con el laboratorio, desplazándose por el laboratorio y observando cada uno de los componentes de este laboratorio. Se tomo como referencia la siguiente imagen del laboratorio de Ingles de la Carrera de Informática.

Mediante la imagen referencia del Laboratorio, se logro el siguiente resultado de modelado.

2.- TÉCNICAS Y HERRAMIENTAS

Para el modelado del Laboratorio, se hizo uso del lenguaje de programación JavaScript, mediante la librería Three.js.

2.1.- Three.js

Three.js es una librería de Javascript desarrollada por Ricardo Cabello en 2010 (actualmente cuenta muchos contribuidores en Github). Esta increíble herramienta nos permite trabajar con gráficos 3D en el navegador, utilizando WebGL, de una manera muy sencilla e intuitiva.

3.- ANÁLISIS Y DISEÑO

La librería Three.js nos ofrece muchas herramientas de trabajo para lograr nuestro modelado deseado. Primero comenzaremos definiendo nuestra Escena o Scene, que será donde estará todo nuestro laboratorio.

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

Luego instanciamos nuestra perspectiva de Camara, será por esta que el usuario se desplace por el laboratorio.

```
    camera = new THREE.PerspectiveCamera(90, 1080 / 620, 0.1, 1000);
```

Luego necesitamos instanciar el renderer, que se encargara de dibujar nuestra escena y establecer la cámara.

```
    renderer = new THREE.WebGLRenderer();
```

Al final llamamos la función animate() que será donde se estará cargando en un bucle nuestra escena y los nuevos eventos.

```
function animate() {
    renderer.render(scene, camera)
}
```

Ahora podemos agregar todos los elementos necesarios a nuestra escena.

Para las paredes y techo utilizamos un mesh de tipo BoxGeometry. Un mesh es un objeto o elemento que agregamos a nuestra escena.

Definimos nuestro mesh

```
    meshTecho = new THREE.Mesh(
```

De tipo BoxGeometry y con las siguientes medidas (x,y,z) y Tipo de Material donde definimos el color.

```
        new THREE.BoxGeometry(20, 30, 1),
        new THREE.MeshPhongMaterial({ color: 0xffffff, wireframe: USE_WIREFRAME })
    );
```

Cambiamos la rotación de nuestro mesh en el eje X y la posición en el eje Y de la siguiente manera.

```
    meshTecho.rotation.x -= Math.PI / 2;
    meshTecho.position.y += 10;
```

Las siguientes propiedades de Meshnos ayudan con las sombras.

Le decimos que nuestro mesh reciba sombras de otros objetos o elementos.

```
    meshTecho.receiveShadow = true;
```

Le decimos que genere sombras en otros objetos u elementos.

```
    meshTecho.castShadow = true;
```

Finalmente agregamos nuestro mesh a nuestra scene para que sea renderizado en nuestro navegador.

```
    scene.add(meshTecho);
```

Este proceso se repite con todas las paredes, techo, piso y la pizarra.

Para los ordenadores, mesas y escritorios nos ayudamos de assets de tipo OBT y MTL.

OBJ es un archivo que define cada vértice de nuestro objeto y MTL es otro archivo que se encarga de la definiciones de materiales del objeto, es decir su textura, color y otros.

Para usar estos archivos en Three.js necesitamos de los siguientes módulos, que podemos encontrarlos en el repositorio oficial de Three.js.

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

La manera de agregar un asset de este tipo es muy similar con la que se trabajan los mesh, solo que todo se encapsula en la funcion “load”, y se mandan por parámetros primero el archivo MTL y luego por medio de un Callback el archivo OBJ.

```
mtlLoader.load("models/desk.mtl", function(materials) {
            materials.preload();
            var objLoader = new THREE.OBJLoader();
            objLoader.setMaterials(materials);
            objLoader.load("models/desk.obj", function(mesh) {

                mesh.traverse(function(node) {
                    if (node instanceof THREE.Mesh) {
                        node.castShadow = true;
                        node.receiveShadow = true;
                    }
                })
                scene.add(mesh);
                mesh.scale.x = 6;
                mesh.scale.y = 5;
                mesh.scale.z = 5;
                mesh.position.set(-5, 0, -12);
                mesh.rotation.y = -Math.PI / 64;

            })

        })
```

También podemos agregar luces a nuestro scena de la siguiente manera:

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

Es así como se trabaja con Three.js de una manera divertida, todo se resume en agregar objetos u elementos a nuestro SCENE, donde el RENDER se encarga de dibujar toda nuestro SCENE y mostrarlo en nuestro navegador.