UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

Facultad de Ingeniería de Producción y Servicios

Ciencia de la Computación

Laboratorio 3

ALUMNOS:
Pfuturi Huisa, Oscar David
Quispe Menor, Hermogenes
Quiñonez Lopez, Efrain German
Fernandez Mamani, Brayan Gino
Santos Apaza, Yordy Williams

DOCENTE:
MSc. Vicente Machaca Arceda

CURSO:
Computación Gráfica

5 de mayo de 2021

Índice

1. Github

- https://github.com/oscar-pfuturi-h/Comp-Grafica/tree/main/practica_3

2. Temática

La temática escogida para este trabajo fue medieval. Dado a la tematica escogida y que se representa por la ficción, los elementos creados son

- Castillo
- Torres
- Dragones

3. Funciones

3.1. Crear Castillo

Para la creación del castillo, se modulo en varias funciones, tales son:

la creación de las torres:

```
function CrearCastillo() {
    var paredMaterial = new THREE.MeshLambertMaterial({ color: 0xeeeeee });
    var roofMaterial = new THREE.MeshLambertMaterial({ color: 0xd50000 });

    function crearTorre(alturaT, alturaR, radio) {
        var torre = new THREE.Mesh(new THREE.CylinderGeometry(radio, radio, alturaT, 30), paredMaterial);
        torre.castShadow = true;
        torre.position.y = 5;

        var roof = new THREE.Mesh(new THREE.CylinderGeometry(0, radio, alturaR, 30), roofMaterial);
        roof.castShadow = true;
        roof.position.y = alturaT - (alturaT / 2 - alturaR / 2);
        torre.add(roof);

        return torre;
```

La creación de las paredes:

```
function crearpared(paredWidth) {
    var paredHeight = 38;
    var paredDepth = 10;
    var paredGeometry = new THREE.CubeGeometry(paredWidth, paredHeight, paredDepth);
    var pared = new THREE.Mesh(paredGeometry, paredMaterial);
```

```
    pared.castShadow = true;
    pared.position.y = paredHeight / 2;
    var battlementSize = 4;
    var battlementGeometry = new THREE.CubeGeometry(battlementSize, battlementSize, battlementSize);

    for (var x = 13 + -(paredWidth / 2) + battlementSize / 2; x < paredWidth / 2 - 10; x += battlementSize * 2) {
        var battlement = new THREE.Mesh(battlementGeometry, paredMaterial);
        battlement.castShadow = true;
        battlement.position.set(x, paredHeight / 2 + battlementSize / 2, paredDepth / 2 - battlementSize / 2);
        pared.add(battlement);
}
```

La creación de la entrada:

```
function crearGate() {
    var gateBuildingWidth = 50;
    var gateBuildingHeight = 50;
    var gateBuildingDepth = 40;

    var gateBuilding = new THREE.Mesh(new THREE.CubeGeometry(gateBuildingWidth, gateBuildingHeight, gateBuildingDepth), paredMaterial);
    gateBuilding.castShadow = true;

    gateBuilding.position.y = gateBuildingHeight / 2;

    var ellipse = new THREE.EllipseCurve(0, 0, 20, 30, 0, Math.PI);
    var ellipsePath = new THREE.Path(ellipse.getPoints(50));

    var gateGeometry = new THREE.ShapeGeometry(ellipsePath.toShapes()[0]);
    var gateMaterial = new THREE.MeshBasicMaterial({ color: 0x4e3100 });

    var outerGate = new THREE.Mesh(gateGeometry, gateMaterial);
    var innerGate = new THREE.Mesh(gateGeometry, gateMaterial);
```

3.2. Funciones auxiliares y básicas para crear dragones y algunos objetos

Al diseñar varios objetos complejos nos dimos cuenta la gran cantidad de código duplicado que encontramos al crear algún objeto compuesto, por lo cual, al igual que trabajar con VTK, Three js al estar basado en el lenguaje de programación Javascript, nos permite abstraer algunos conceptos para crear funciones que nos ayudaran a crear de forma mas rápida y ordenada nuevos objetos.

A continuación están algunas de las funciones que nos ayudaron a crear nuestro trabajo de laboratorio, tenemos funciones que nos transforman valores para manejarlos fácilmente(2 primeras funciones) y luego tenemos las demás funciones que se encargan de crear objetos de three js configurables que de forma fácil, solo pasando parámetros los obtenemos para poder agregarlos a nuestra escena general.

```
const gradosARadianes = deg => (deg * Math.PI) / 180.0;
function hexToRgb(hex) {...}
function crear_cubo(x, y, z, color){...}
function crear_cilindro(x, y, z, r, color) {...}
function crear_esfera(r, x, y, color, opacity) {...}
function crear_cono(rtop, rbotton, altura, resolucion, color) {...}
function crear_cola(scale, color_base, color_alas) {...}
function crear_cabeza(color_base, color_alas) {...}
```

Luego tenemos la función Dragón la que se encarga de usar las funciones ya descritas para crear cada objeto, algo muy útil en Three js que no encontramos de forma fácil en la librería VTK es agregar un objeto X a otro objeto Y, lo cual nos da como resultado que el objeto X posee el desplazamiento respecto a su objeto Padre, en este caso seria Objeto Y, de forma fácil no nos tenemos que preocupar al momento de generar movimiento a cada articulación.

```
function dragon(color_base, color_alas) {
    let dragon = new THREE.Object3D();
    ...
    //ADD
    dragon.add(body);
    dragon.add(ala1); dragon.add(ala2);
    dragon.add(p1); dragon.add(p2); dragon.add(p3); dragon.add(p4);
    dragon.add(cola);
    dragon.add(cabeza);
    return dragon;
}
```

Para crear la casa básica, de igual manera se usa las distintas funciones para armar poco a poco la edificación.

4. Resultados

Figura 1: Castillo

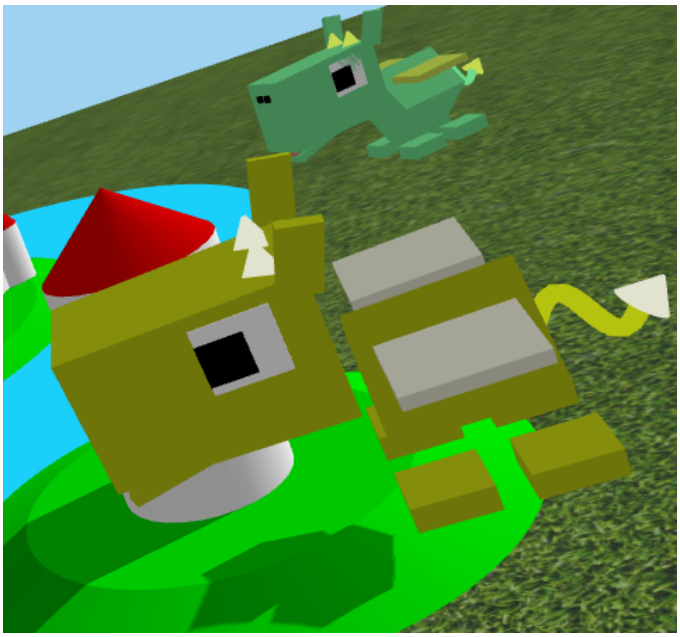

Figura 2: Dragones

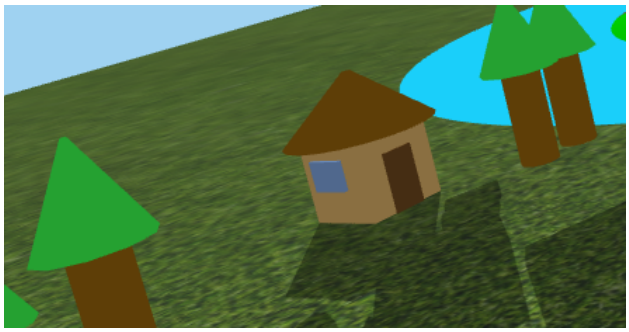

Figura 3: Casa básica

Figura 4: Panorama completo