Proyecto final: Gráficas computacionales

07 de febrero 2020

Saul Neri Ortiz, A01652526

Implementación del auto

El auto fue diseñado de manera manual utilizando la geometría `ExtrudeGeometry`, para la cual se necesita definir una forma describiendo los vértices por medio de coordenadas. De esta maner se pueden crear bloques de construcción con formas irregulares.

Utilizando esta técnica, se diseñaron la puertas, cajuela, láminas laterales, capó, y ventanas del auto.

La geometría del lado posterior del auto se definió de la siguiente manera:

```
var carSideShape = new THREE.Shape();
carSideShape.moveTo(-1.3, 0.2);
carSideShape.lineTo(-0.5, 0.2);
carSideShape.lineTo(-0.2, 0.4);
carSideShape.lineTo(-0.2, 0);
carSideShape.lineTo(0.4, 0);
carSideShape.lineTo(0.5, 0.3);
carSideShape.lineTo(1.3, 0.3);
carSideShape.lineTo(1.3, -0.3);
carSideShape.lineTo(1.1, -0.3);
carSideShape.lineTo(1.1, -0.15);
carSideShape.lineTo(0.9, 0);
carSideShape.lineTo(0.7, 0);
carSideShape.lineTo(0.5, -0.15);
carSideShape.lineTo(0.5, -0.3);
carSideShape.lineTo(-0.5, -0.3);
carSideShape.lineTo(-0.5, -0.15);
carSideShape.lineTo(-0.7, 0);
carSideShape.lineTo(-0.9, 0);
carSideShape.lineTo(-1.1, -0.15);
carSideShape.lineTo(-1.1, -0.3);
carSideShape.lineTo(-1.3, -0.3);
carSideShape.lineTo(-1.3, 0.2);
var carSideGeom = new THREE.ExtrudeGeometry(carSideShape, extrudeSettings);
```

Repitiendo la técnica, se diseñó el auto completamente desde cero.

Drone

El drone, fue implementado usando un modelo OBJ ya existente y las hélices son geometrías básicas de THREE.js para poder darles movimiento independiente.

El modelo OBJ se carga de la siguiente manera:

1. Importar OBJLoader.js
2. Instanciar un `THREE.LoadingManager`
3. Implementar los métodos, `onLoad, onError, onProgress.`
4. Instanciar un `THREE.OBJLoader`
5. Importar el modelo

```
var drone = new THREE.Object3D();
var loader = new THREE.OBJLoader(manager);
loader.load('drone/model.obj', function(object) {
    drone = object;
    prop1.position.y += 0.1;
    prop1.position.x += 0.15;
    prop1.position.z += 0.57;
    object.add(prop1);
    prop2.position.y += 0.1;
    prop2.position.x -= 0.15;
    prop2.position.z -= 0.57;
    object.add(prop2);
    prop3.position.y += 0.1;
    prop3.position.x -= 0.57;
    prop3.position.z += 0.15;
    object.add(prop3);
    prop4.position.y += 0.1;
    prop4.position.x += 0.57;
    prop4.position.z -= 0.15;
    object.add(prop4);
    object.position.y += 1.45;
    car.add(object);
});
```

Parque de diversiones

Rueda de la fortuna

Se utilizó el método usado en la tarea 3.1 para diseñar engranes y se adaptó para poder crear la estrella central de la estructura.

```
function gearShape(r1, r2) {
    var shape = new THREE.Shape();
    shape.moveTo(r1, 0);
    var angle = 0;
    for (i = 0; i < 12; i++) {
        shape.lineTo(r1 * Math.cos(angle), r1 * Math.sin(angle));
        angle += 1 / 18 * Math.PI;
        shape.lineTo(r1 * Math.cos(angle), r1 * Math.sin(angle));
        angle += 1 / 36 * Math.PI;
        shape.lineTo(r2 * Math.cos(angle), r2 * Math.sin(angle));
        angle += 1 / 18 * Math.PI;
        shape.lineTo(r2 * Math.cos(angle), r2 * Math.sin(angle));
        angle += 1 / 36 * Math.PI;
    }
    shape.lineTo(r1, 0);

    return (shape);
}
```

Para que las canastas no se muevan y obedezcan las leyes de gravedad, se les dió un movimiento en el eje z con sentido inverso al movimiento giratorio de la rueda, pero con la misma magnitud.

```
        ferrisWheel.rotation.z += 0.01;
        basket1.rotation.z -= 0.01
        basket2.rotation.z -= 0.01
        basket3.rotation.z -= 0.01
        basket4.rotation.z -= 0.01
```

Carrusel

Los caballos se mueven de manera ascendente y descente de manera periódica. Se implementó una serie de condiciones para verificar la dirección del movimiento vertical de cada caballo.

```
            if (horse1Up && horse1.position.y < 0.9) {
                horse1.position.y += 0.02;
            } else if (horse1Up) {
                horse1Up = !horse1Up;
            } else if (!horse1Up && horse1.position.y > 0.1) {
                horse1.position.y -= 0.02;
            } else {
                horse1Up = !horse1Up;
            }
```

Adicionalmente se verifican los límites superior e inferior para evitar colisiones.

Frisbee

El mecanismo del frisbee es muy simple. Existen dos ejes rotacionales independientes y al organizar el `Object3D` de manera modular, es posible lograr los movimientos deseados.

```
var frisbee = new THREE.Object3D();
var frisbeeArm = new THREE.Mesh(new THREE.CylinderGeometry(0.2, 0.2, 4, 128, 1), googleGreenMaterial);
frisbee.add(frisbeeArm);
frisbeeArm.position.y += 2;

var frisbeeDisc = new THREE.Mesh(new THREE.TorusGeometry(1.4, 0.16, 128, 128), googleRedMaterial);
frisbeeDisc.rotateX(Math.PI / 2);
frisbee.add(frisbeeDisc);
frisbeeDisc.position.y += 4;
```

```
        frisbeeComplete.add(frisbeBase);
        frisbeeComplete.add(frisbee);
```

```
        frisbee.rotation.z += 0.03;
        frisbeeDiscStar.rotation.z += 0.05;
```

Torre de Pisa

Está diseñada de manera modular. Se apilan capas de elementos anidados para incrementar su altura.
Cada capa consta de dos partes. La división con mayor altura y que contiene las columna. Y un pequeño disco que divide los pisos entre sí.

La sección con las columnas es construida de manera dinámica con un método que crea y agrega las columnas al grupo indicado.

```
function addColumns(object, n, objectRadius, columnRadius, columnHeight, material)
{
            let divAngle = Math.PI * 2 / n;
            for (let i = 0; i < n; i++) {
                let x = objectRadius * Math.cos(divAngle * i);
                let z = objectRadius * Math.sin(divAngle * i);
                let c = new THREE.Mesh(new THREE.CylinderGeometry(
                                                    columnRadius,
                                                    columnRadius,
                                                    columnHeight,
                                                    128,
                                                    1), material);
                object.add(c);
                c.position.x = x;
                c.position.z = z;
            }
        }
```

```
pisaFloor = new THREE.Mesh(new THREE.CylinderGeometry(1.2,1.2,1,128,1), marble1Material);
addColumns(pisaFloor, 30, 1.50, 0.1, 1, marble2Material);
pisaFloor.position.y += 8.9;
pisa.add(pisaFloor);
```