

Armando Canto Garcia A01322361

Luis Alfredo Leon Villapun A01322275

Graficas computacionales

Final Project

Introduction:

For this final project, we had to implement several objects:

1)

a car that could open the trunk, its doors , rotate its wheels, move backwards and forward and also include a drone that could be attached and reattached in the car.

2) the structure of the ITESM campus puebla

3)Mix those 2 elements with the previous assignment of the amusement park

Details and explanations:

1) the car:

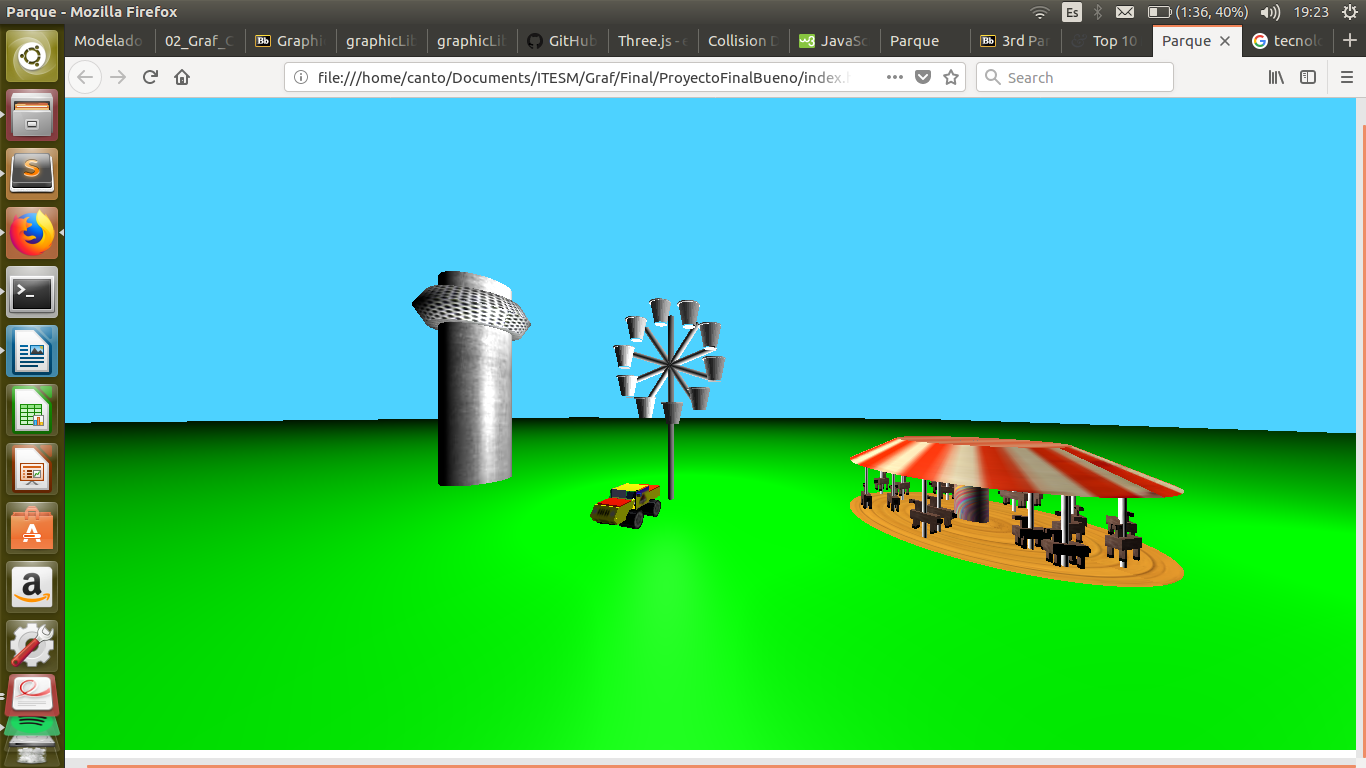
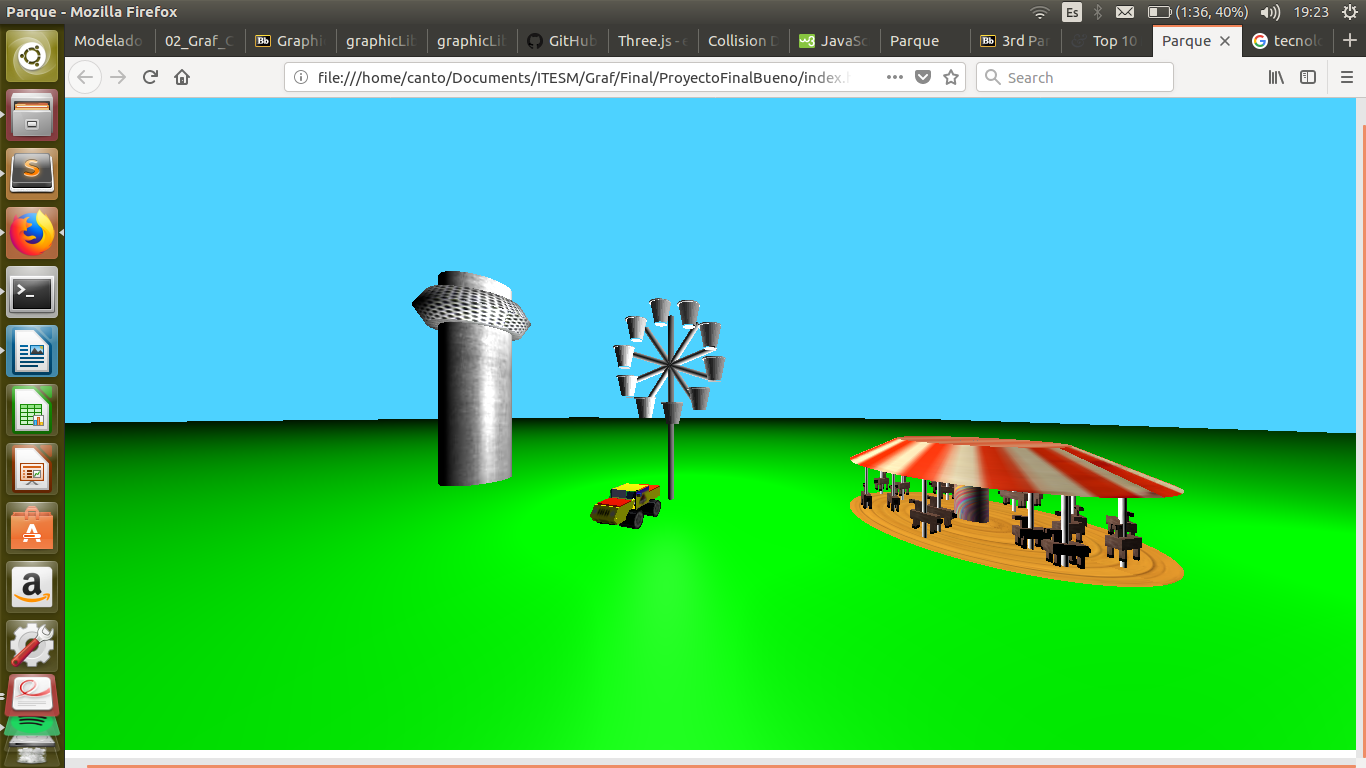
2) ITESM campus puebla:

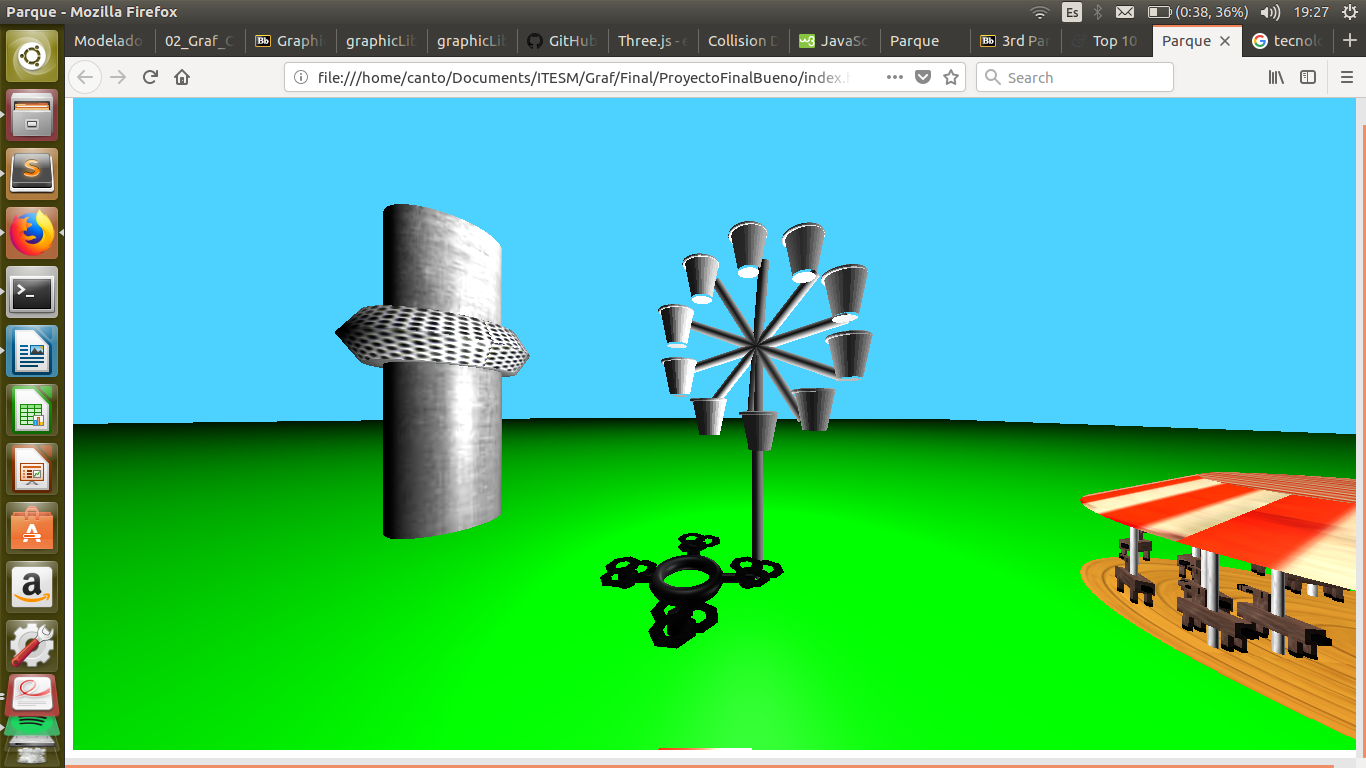
We had trouble building this object due to its complexity. We decided to focus on the car and its functionality than building the ITESM campus.

3) Amusement park:

The park was the previous assignment for this class. It consists of 3 elements. A carousel, A wheel, and a Gyro tower. For the carousel, we had to make several functions for making horses and poles. Once we had them created, we put them in a circular form. Then we made up the carousel body and integrate it to the rest of the horses. We had a little trouble with the horse movement because the horses didnt obey the limits we here putting. For the gyro tower it was not that hard. We build the geometries and the meshes. Then we just mixed them and use as a reference the horse movement.

Screenshots





Code:

<!----

Armando Canto Garcia A01322361

Luis Alfredo Leon Villapun A01322275

-->

<html>

<head>

<meta charset="utf-8">

<title> Parque</title>

<style>

</style>

</head>

<body>

<script src="three.min.js"> </script>

<script src="three.js/build/three.min.js"> </script>

<script src="three.js/examples/js/loaders/DDSLoader.js"></script>

<script src="three.js/examples/js/loaders/MTLLoader.js"></script>

<script src="three.js/examples/js/loaders/OBJLoader.js"></script>

<script>

var baskets = [];

//Camera

var scene = new THREE.Scene();

var camera = new THREE.PerspectiveCamera(100, window.innerWidth/window.innerHeight, 0.01, 3000);

var renderer = new THREE.WebGLRenderer();

renderer.setSize(window.innerWidth, window.innerHeight);

document.body.appendChild(renderer.domElement);

camera.position.set(0,2,17);

renderer.setClearColor(0x4dd2ff); //Sky

var angle = 0.0;

var cameraMove = 1

var grassGeo = new THREE.CubeGeometry( 1000, 1, 1000); //Grass declaration

var grassMesh = new THREE.MeshPhongMaterial( {color: 0x00b300,wireframe: false} );

var floor = new THREE.Mesh( grassGeo, grassMesh );

floor.position.set(-250,-10,-250);

scene.add(floor);

var ang\_rad = 0;

var canRotateWheel = 1;

var doorsOpen = false;

var hoodOpen = false;

var trunkOpen = false;

var deployDrone = 0 ;

function toRadians(angle){ //Function to get radians

return (Math.PI \* angle) / 180;

}

window.addEventListener('keydown',doKeyDown,true);

function doKeyDown(evt){

switch (evt.keyCode) {

case 87: //W

camera.position.y+=cameraMove;

break;

case 88: //W

if(deployDrone == 0){

deployDrone = 1;

}

else if (deployDrone== 1){

deployDrone = 0;

}

break;

case 65: //A

angle+=90;

camera.position.z -= (cameraMove\*Math.cos( (angle\*(Math.PI/180)) ));

camera.position.x -= (cameraMove\*Math.sin( (angle\*(Math.PI/180)) ));

angle-=90;

break;

case 83: //S

camera.position.y-=cameraMove;

break;

case 68: //D

angle-=90;

camera.position.z -= (cameraMove\*Math.cos( (angle\*(Math.PI/180)) ));

camera.position.x -= (cameraMove\*Math.sin( (angle\*(Math.PI/180)) ));

angle+=90;

break;

case 38://Up

camera.position.z -= (cameraMove\*Math.cos( (angle\*(Math.PI/180)) ));

camera.position.x -= (cameraMove\*Math.sin( (angle\*(Math.PI/180)) ));

break;

case 37://Left

camera.rotation.y += (10 \* Math.PI / 180 );

angle+=10;

break;

case 40://Down

camera.position.z += (cameraMove\*Math.cos( (angle\*(Math.PI/180)) ));

camera.position.x += (cameraMove\*Math.sin( (angle\*(Math.PI/180)) ));

break;

case 39://Right

camera.rotation.y -= (10 \* Math.PI / 180 );

angle-=10;

break;

case 71: //G go backward car

carGroup.position.z -= Math.cos(carWheelFrontLeftGroup.rotation.y );

carGroup.position.x -= Math.sin(carWheelFrontLeftGroup.rotation.y );

carGroup.rotation.y = carWheelFrontLeftGroup.rotation.y;

if(carGroup.rotation.y){

}

break;

case 84: //T go forward car

carGroup.position.z += Math.cos(carWheelFrontLeftGroup.rotation.y );

carGroup.position.x += Math.sin(carWheelFrontLeftGroup.rotation.y );

carGroup.rotation.y = carWheelFrontLeftGroup.rotation.y;

break;

case 70: //F rotate wheels

if(canRotateWheel <= 3){

carWheelFrontLeftGroup.rotation.y -= toRadians(10);

carWheelFrontRightGroup.rotation.y -= toRadians(10);

canRotateWheel++;

}

break;

case 72: //H rotate wheels

if(canRotateWheel >= 0){

carWheelFrontLeftGroup.rotation.y += toRadians(10);

carWheelFrontRightGroup.rotation.y += toRadians(10);

canRotateWheel--;

}

break;

case 74: //J close doors

if(doorsOpen){

carLeftDoorGroup.rotation.y += toRadians(70);

carRightDoorGroup.rotation.y -= toRadians(70);

doorsOpen = false;

}

break;

case 85: //U open doors

if(!doorsOpen){

carLeftDoorGroup.rotation.y -= toRadians(70);

carRightDoorGroup.rotation.y += toRadians(70);

doorsOpen = true;

}

break;

case 89: //Y open hood

if(trunkOpen){

carTrunkGroup.rotation.x += toRadians(50);

trunkOpen = false;

}

break;

case 82: //R close hood

if(!trunkOpen){

carTrunkGroup.rotation.x -= toRadians(50);

trunkOpen = true;

}

break;

case 86: //V open trunk

if(hoodOpen){

carHoodGroup.rotation.x -= toRadians(50);

hoodOpen = false;

}

break;

case 66: //B close trunk

if(!hoodOpen){

carHoodGroup.rotation.x += toRadians(50);

hoodOpen = true;

}

break;

}

}

//Light

var lightpoint = new THREE.PointLight(0xCCCCCC);

var lightpoint2 = new THREE.PointLight(0xCCCCCC);

var lightpoint3 = new THREE.PointLight(0xCCCCCC);

var lightpoint4 = new THREE.PointLight(0xCCCCCC);

scene.add(lightpoint);

scene.add(lightpoint2);

scene.add(lightpoint3);

scene.add(lightpoint4);

lightpoint.position.set(0,100,0);

lightpoint2.position.set(0,-100,0);

lightpoint3.position.set(0,0,100);

lightpoint3.position.set(50,0,0);

//ALL MATERIALS:

var BlackMt = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/wood.jpg"),color: 0x000000,wireframe: false})

var Marmol = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/marmolcafe.jpg"),color: 0xFFFFFF,wireframe: false})

var MarmolCylMaterial = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/marmol.jpg"),color: 0xFFFFFF,wireframe: false});

var woodPlanks = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/shwood.jpeg"),color: 0xCC9933,wireframe: false});

var Lollipop = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/lp.jpg"),color: 0xFFFFFF,wireframe: false})

var circusTop = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/circus.jpg"),color: 0xFFFFFF,wireframe: false});

var cabina = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/cabina.jpg"),color: 0xFFFFFF, wireframe: false})

var gyroBaseMat = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/metal.jpg"),color: 0xFFFFFF, wireframe: false})

var DroneSkin = new THREE.MeshPhongMaterial({map:THREE.ImageUtils.loadTexture("img/droneskin.jpg"),color: 0xFFFFFF, wireframe: false})

var redMaterial = new THREE.MeshLambertMaterial({

color: 0xaaaaaa });

//Drone implementation

//Creation of the drone body

var Drone = new THREE.Object3D();

var DroneGeo =new THREE.TorusGeometry( 2, 0.5, 6, 100 );

var baseWingGeo =new THREE.CylinderGeometry(0.28, 0.28, 0.7, 100);

var baseWingGeoBack =new THREE.CylinderGeometry(0.28, 0.28, 1.5, 100);

var wingSupportGeo =new THREE.CylinderGeometry(0.55, 0.55, 0.8, 100);

var DroneMesh = new THREE.Mesh( DroneGeo, DroneSkin );

var baseWing1 = new THREE.Mesh( baseWingGeo, BlackMt );

var baseWing2 = new THREE.Mesh( baseWingGeo, BlackMt );

var baseWing3 = new THREE.Mesh( baseWingGeoBack, BlackMt );

var baseWing4 = new THREE.Mesh( baseWingGeoBack, BlackMt );

var wingSupport1 = new THREE.Mesh( wingSupportGeo, DroneSkin );

var wingSupport2 = new THREE.Mesh( wingSupportGeo, DroneSkin );

var wingSupport3 = new THREE.Mesh( wingSupportGeo, DroneSkin );

var wingSupport4 = new THREE.Mesh( wingSupportGeo, DroneSkin );

//Function that makes wings

function makeWing(){

var Wing = new THREE.Object3D();

var geometryWing = new THREE.RingGeometry( 12, 5, 7 );

var wingMesh = new THREE.Mesh( geometryWing, DroneSkin );

var wingMesh2 = new THREE.Mesh( geometryWing, DroneSkin );

var wingMesh3 = new THREE.Mesh( geometryWing, DroneSkin );

wingMesh2.position.x = 7;

wingMesh2.position.y = -12;

wingMesh3.position.x = -7;

wingMesh3.position.y = -12;

Wing.add(wingMesh);

Wing.add(wingMesh2);

Wing.add(wingMesh3);

Wing.scale.set(0.1,0.1,0.1);

Wing.rotation.x+=Math.PI/2;

return Wing;

}

//Wings creation

WingGroup = [];

for(var i = 0;i<4;i++){

WingGroup[i] = makeWing();

}

WingGroup[0].position.y = 0.5;

WingGroup[0].position.x = 4.3;

WingGroup[0].position.z= 0.7;

WingGroup[1].position.y = 0.5;

WingGroup[1].position.x = -4.3;

WingGroup[1].position.z= 0.7;

WingGroup[2].position.y = 0.5;

WingGroup[2].position.x = 0;

WingGroup[2].position.z= 5.2;

WingGroup[3].position.y = 0.5;

WingGroup[3].position.x = 0;

WingGroup[3].position.z= -3.5;

Drone.add(WingGroup[0]);

Drone.add(WingGroup[1]);

Drone.add(WingGroup[2]);

Drone.add(WingGroup[3]);

DroneMesh.rotation.x+=Math.PI/2;

Drone.add(baseWing1);

Drone.add(baseWing2);

Drone.add(baseWing3);

Drone.add(baseWing4);

Drone.add(wingSupport1);

Drone.add(wingSupport2);

Drone.add(wingSupport3);

Drone.add(wingSupport4);

Drone.add(DroneMesh);

baseWing1.rotation.x+=Math.PI/2;

baseWing2.rotation.x+=Math.PI/2;

baseWing3.rotation.z+=Math.PI/2;

baseWing4.rotation.z+=Math.PI/2;

baseWing1.position.z= 3;

baseWing2.position.z= -3;

baseWing3.position.x= -2.8;

baseWing4.position.x= 2.8;

Drone.position.y= 10;

wingSupport1.position.x = 4.04;

wingSupport2.position.x = -4.04;

wingSupport3.position.z = 4.04;

wingSupport4.position.z = -4.04;

Drone.scale.set(0.16,0.16,0.16);

Drone.position.y=-8;

Drone.position.z= 6;

//Gyro tower

var GyroGroup = new THREE.Object3D();

var gyroGeo =new THREE.TorusGeometry( 5, 3, 6, 100 );

var torus = new THREE.Mesh( gyroGeo, cabina );

var baseGeometry = new THREE.CylinderGeometry( 5, 5, 35, 64 );

var cylinderGyro = new THREE.Mesh( baseGeometry, gyroBaseMat );

GyroGroup.add( torus);

torus.rotation.x+=Math.PI/2;

torus.position.y-=14;

GyroGroup.add( cylinderGyro );

GyroGroup.scale.set(0.75,0.75,0.75);

GyroGroup.position.x= -9.2;

scene.add(GyroGroup);

GyroGroup.position.x = -28;

GyroGroup.position.y = 3;

//Carrousel

var CarrouselGroup = new THREE.Object3D();

function makeHorse(){

//Geometries

var HorseBody = new THREE.CubeGeometry(13,5,5);

var HorseNeck = new THREE.CubeGeometry(7,3,1);

var HorseHead = new THREE.CubeGeometry(5,2,2);

var HorseTail = new THREE.CubeGeometry(2,2,2);

var HorseLeg = new THREE.CubeGeometry(1.5,5,1.5);

var HorseEye = new THREE.CubeGeometry(1,1,1);

var HorseHoof = new THREE.CubeGeometry(1.5,1.5,1.5);

var HorseNose = new THREE.CubeGeometry(1.01,2.01,2.01);

//Mesh

var BodyMesh = new THREE.Mesh(HorseBody,Marmol);

var NeckMesh = new THREE.Mesh(HorseNeck,Marmol);

var HeadMesh = new THREE.Mesh(HorseHead,Marmol);

var RightEye = new THREE.Mesh(HorseEye,BlackMt);

var LeftEye = new THREE.Mesh(HorseEye,BlackMt);

var Nose = new THREE.Mesh(HorseNose,BlackMt);

var Tail = new THREE.Mesh(HorseTail,BlackMt);

var FrontLeg1 = new THREE.Mesh(HorseLeg,Marmol);

var FrontLeg2 = new THREE.Mesh(HorseLeg,Marmol);

var BackLeg1 = new THREE.Mesh(HorseLeg,Marmol);

var BackLeg2 = new THREE.Mesh(HorseLeg,Marmol);

var FrontHoof1 = new THREE.Mesh(HorseHoof,BlackMt);

var FrontHoof2 = new THREE.Mesh(HorseHoof,BlackMt);

var BackHoof1 = new THREE.Mesh(HorseHoof,BlackMt);

var BackHoof2 = new THREE.Mesh(HorseHoof,BlackMt);

//Groups

var Group1 = new THREE.Object3D(); //eyes

var Group2 = new THREE.Object3D(); //Legs

var Group3 = new THREE.Object3D(); //Legs front

var Group4 = new THREE.Object3D(); //Legs back

var Group5 = new THREE.Object3D(); //Head

var GeneralGroup = new THREE.Object3D();

Group1.add(RightEye);

Group1.add(LeftEye);

Group5.add(Nose);

Group5.add(NeckMesh);

Group5.add(HeadMesh);

Group5.add(Group1);

Group3.add(FrontLeg1);

Group3.add(FrontLeg2);

Group3.add(FrontHoof1);

Group3.add(FrontHoof2);

Group4.add(BackLeg1);

Group4.add(BackLeg2);

Group4.add(BackHoof1);

Group4.add(BackHoof2);

Group2.add(Group3);

Group2.add(Group4);

GeneralGroup.add(Group2);

GeneralGroup.add(Group5);

GeneralGroup.add(BodyMesh);

GeneralGroup.add(Tail);

//Rotations and positions

RightEye.position.z+=1;

LeftEye.position.z-=1;

RightEye.position.x-=0.3;

LeftEye.position.x-=0.3;

Group1.position.x-=4.5;

Group1.position.y+=7.5;

Group3.rotation.z-=(Math.PI/180);

Group4.rotation.z+=(Math.PI/180);

FrontLeg1.position.y-=2.5;

FrontLeg2.position.y-=2.5;

BackLeg1.position.y-=2.5;

BackLeg2.position.y-=2.5;

FrontHoof1.position.y-=5.7;

FrontHoof2.position.y-=5.7;

BackHoof1.position.y-=5.7;

BackHoof2.position.y-=5.7;

Group3.position.y+=2.5;

Group4.position.y+=2.5;

FrontLeg1.position.z+=1.8;

FrontLeg2.position.z-=1.8;

BackLeg1.position.z+=1.8;

BackLeg2.position.z-=1.8;

FrontHoof1.position.z+=1.8;

FrontHoof2.position.z-=1.8;

BackHoof1.position.z+=1.8;

BackHoof2.position.z-=1.8;

NeckMesh.rotation.z+=(90\*Math.PI/180);

NeckMesh.position.x-=3;

NeckMesh.position.y+=5;

Group2.position.y-=4;

HeadMesh.position.y+=7.5;

HeadMesh.position.x-=5;

Nose.position.y+=7.5;

Nose.position.x-=7;

Tail.position.x+=7.5;

Group3.position.x-=4.3;

Group4.position.x+=4.3;

GeneralGroup.scale.set(0.28,0.28,0.28);

return GeneralGroup;

}

function makeCyl(){ //Function to make horse cylinders

var cylGeoHorse = new THREE.CylinderGeometry(0.30,0.30,10);

var cylMesh = new THREE.Mesh(cylGeoHorse,MarmolCylMaterial);

cylMesh.position.y+=2.5;

return cylMesh;

}

var HorseCylUnion = new THREE.Object3D();

var unionAngle = 0;

var updownHorse = 360/(15/2);

var HorseDownGroup = [];

var HorseUpGroup = [];

var Cylgroup = [];

var GeneralGroups = [];

var aux = 0; //Aux var

for(var i = 0; i < (15/2); i++){

HorseUpGroup[i] = makeHorse();

Cylgroup[aux] = makeCyl();

GeneralGroups[aux] = new THREE.Object3D();

GeneralGroups[aux].add(HorseUpGroup[i]);

GeneralGroups[aux].add(Cylgroup[aux]);

var x = (18\*(Math.sin(unionAngle\*Math.PI/180)));

var z = (18\*(Math.cos(unionAngle\*Math.PI/180))); //First group of horses positioning

GeneralGroups[aux].position.set(x,0,z); //Positioning

GeneralGroups[aux].rotation.y+=(unionAngle\*Math.PI/180); //Rotation

HorseCylUnion.add(GeneralGroups[aux]);

aux++;

HorseDownGroup[i] = makeHorse();

Cylgroup[aux] = makeCyl();

GeneralGroups[aux] = new THREE.Object3D();

GeneralGroups[aux].add(HorseDownGroup[i]);

GeneralGroups[aux].add(Cylgroup[aux]);

x = (13\*(Math.sin(unionAngle\*Math.PI/180)));

z = (13\*(Math.cos(unionAngle\*Math.PI/180))); //Second group of horses positioning

GeneralGroups[aux].position.set(x,0,z);

GeneralGroups[aux].rotation.y+=(unionAngle\*Math.PI/180);

HorseCylUnion.add(GeneralGroups[aux]);

unionAngle+=updownHorse;

}

//Carrousel body contruction

//Geometry

var floorCarrouselGeometry = new THREE.CylinderGeometry(1,21,0.25,360,9,false,0,Math.PI\*2);

var cylMeshGeometry = new THREE.CylinderGeometry(2.5,2.5,15,360,1,false,0,Math.PI\*2);

var RoofCarrouselGeometry = new THREE.CylinderGeometry(12,21,5,300,1,false,0,Math.PI\*2);

//Mesh

var floorCarrousel = new THREE.Mesh(floorCarrouselGeometry, woodPlanks);

floorCarrousel.position.y-=3;

var centerCyl = new THREE.Mesh(cylMeshGeometry, Lollipop);

centerCyl.position.y+=5;

var RoofCarrousel = new THREE.Mesh(RoofCarrouselGeometry, circusTop);

RoofCarrousel.position.y+=10;

CarrouselGroup.add(floorCarrousel);

CarrouselGroup.add(centerCyl);

CarrouselGroup.add(RoofCarrousel);

CarrouselGroup.add(HorseCylUnion);

CarrouselGroup.position.x+=25;

CarrouselGroup.position.y-=6.5;

CarrouselGroup.scale.set(0.40,0.40,0.40);

scene.add(CarrouselGroup);

CarrouselGroup.position.z=-3;

CarrouselGroup.position.y= -8;

let sube = true;

let torusPos = 0;

var speed = 0.015;

var limites = 2;

var arriba = false;

var abajo= 0;

//Wheel implementation

var basket1 = new THREE.Mesh();

function loadBasket(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round(percentComplete, 2) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

var mtlLoader = new THREE.MTLLoader();

//LOAD

var obj;

mtlLoader.load( 'Texturas/basket.mtl', function( materials ) {

materials.preload();

var objLoader = new THREE.OBJLoader();

objLoader.setMaterials( materials );

objLoader.load( 'Texturas/basket.obj', function ( object ) {

obj = object;

object.traverse( function ( child )

{

if ( child instanceof THREE.Mesh )

{

child.material = redMaterial;

}

} );

baskets.push(obj);

}, onProgress, onError );

});

}

var numBaskets = 10;

var angleInc = 360/numBaskets;

var angleAct = 90;

var radius = 5;

for(var i = 0; i < numBaskets; i++){

loadBasket();

}

alert(baskets.length);

alert(baskets[0]);

for(var i = 0; i < numBaskets; i++){

var x = Math.cos(toRadians(angleAct)) \* radius;

var y = Math.sin(toRadians(angleAct)) \* radius;

baskets[i].position.set(x,y,0);

angleAct+=angleInc;

console.log("Im iterating");

}

var basketGroup = new THREE.Object3D();

for(var i = 0; i < numBaskets; i++){

basketGroup.add(baskets[i]);

}

scene.add(basketGroup);

var sphereGeometry = new THREE.SphereGeometry(1, 32, 32);

var sphere = new THREE.Mesh(sphereGeometry, redMaterial);

sphere.position.set(0,1,0);

var sphereGroup = new THREE.Object3D();

//sphereGroup.add(sphere);

sphereGroup.add(basketGroup);

sphereGroup.position.set(0,3,0);

scene.add(sphereGroup);

var armGeometry = new THREE.CylinderGeometry(0.3, 0.3, 10, 32);

var arm = new THREE.Mesh(armGeometry, redMaterial);

//scene.add(arm);

var armAngle = 100;

var actArm;

var armGroup = new THREE.Object3D();

for(var i = 0; i < numBaskets; i++){

actArm = arm.clone();

actArm.rotation.z += toRadians(armAngle);

armGroup.add(actArm);

armAngle+=angleInc;

}

armGroup.position.set(0,4,-1);

scene.add(armGroup);

var baseArmGeom = new THREE.CylinderGeometry(0.3, 0.3, 15, 32);

var baseArm = new THREE.Mesh(baseArmGeom,redMaterial);

baseArm.position.z = -1;

baseArm.position.y = -4;

scene.add(baseArm);

var carBody = new THREE.Mesh();

function loadCarBody(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'carBody.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'carBody.obj', function ( object ) {

carBody.add( object );

}, onProgress, onError );

} );

}

alert(baskets.length);

loadCarBody();

var carBodyGroup = new THREE.Object3D();

carBodyGroup.add(carBody);

scene.add(carBodyGroup);

//CAR Trunk ----------------------------------------------

var carTrunk = new THREE.Mesh();

function loadCarTrunk(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'carTrunk.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'carTrunk.obj', function ( object ) {

carTrunk.add( object );

}, onProgress, onError );

} );

}

loadCarTrunk();

var carTrunkGroup = new THREE.Object3D();

carTrunkGroup.add(carTrunk);

scene.add(carTrunkGroup);

//CAR Hood ----------------------------------------------

var carHood = new THREE.Mesh();

function loadCarHood(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'carHood.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'carHood.obj', function ( object ) {

carHood.add( object );

}, onProgress, onError );

} );

}

loadCarHood();

var carHoodGroup = new THREE.Object3D();

carHoodGroup.add(carHood);

scene.add(carHoodGroup);

//CAR Door Left ----------------------------------------------

var carLeftDoor = new THREE.Mesh();

function loadCarLeftDoor(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'carLeftDoor.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'carLeftDoor.obj', function ( object ) {

carLeftDoor.add( object );

}, onProgress, onError );

} );

}

loadCarLeftDoor();

var carLeftDoorGroup = new THREE.Object3D();

carLeftDoorGroup.add(carLeftDoor);

scene.add(carLeftDoorGroup);

//CAR Door Right ----------------------------------------------

var carRightDoor = new THREE.Mesh();

function loadCarRightDoor(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'carRightDoor.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'carRightDoor.obj', function ( object ) {

carRightDoor.add( object );

}, onProgress, onError );

} );

}

loadCarRightDoor();

var carRightDoorGroup = new THREE.Object3D();

carRightDoorGroup.add(carRightDoor);

scene.add(carRightDoorGroup);

//CAR Wheel ----------------------------------------------

var carWheelFrontLeft = new THREE.Mesh();

var carWheelFrontRight = new THREE.Mesh();

var carWheelBackLeft = new THREE.Mesh();

var carWheelBackRight = new THREE.Mesh();

function loadCarFrontLeftWheel(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'wheel.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'wheel.obj', function ( object ) {

carWheelFrontLeft.add(object);

}, onProgress, onError );

} );

}

function loadCarFrontRightWheel(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'wheel.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'wheel.obj', function ( object ) {

carWheelFrontRight.add(object);

}, onProgress, onError );

} );

}

function loadCarBackLeftWheel(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'wheel.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'wheel.obj', function ( object ) {

carWheelBackLeft.add(object);

}, onProgress, onError );

} );

}

function loadCarBackRightWheel(){

var onProgress = function ( xhr ) {

if ( xhr.lengthComputable ) {

var percentComplete = xhr.loaded / xhr.total \* 100;

console.log( Math.round( percentComplete, 2 ) + '% downloaded' );

}

};

var onError = function ( xhr ) { };

THREE.Loader.Handlers.add( /\.dds$/i, new THREE.DDSLoader() );

new THREE.MTLLoader()

.setPath( 'Texturas/' )

.load( 'wheel.mtl', function ( materials ) {

materials.preload();

new THREE.OBJLoader()

.setMaterials( materials )

.setPath( 'Texturas/' )

.load( 'wheel.obj', function ( object ) {

carWheelBackRight.add(object);

}, onProgress, onError );

} );

}

loadCarFrontLeftWheel();

loadCarFrontRightWheel();

loadCarBackLeftWheel();

loadCarBackRightWheel();

var carWheelFrontLeftGroup = new THREE.Object3D();

var carWheelFrontRightGroup = new THREE.Object3D();

var carWheelBackLeftGroup = new THREE.Object3D();

var carWheelBackRightGroup = new THREE.Object3D();

carWheelFrontLeftGroup.add(carWheelFrontLeft);

carWheelFrontRightGroup.add(carWheelFrontRight);

carWheelBackLeftGroup.add(carWheelBackLeft);

carWheelBackRightGroup.add(carWheelBackRight);

carWheelFrontRightGroup.position.set(-1,-0.5,1.3);

carWheelFrontRightGroup.rotation.y += toRadians(180);

carWheelFrontLeftGroup.position.set(1,-0.5,1.3);

carWheelBackLeftGroup.position.set(1,-0.5,-1.3);

carWheelBackRightGroup.position.set(-1,-0.5,-1.3);

carWheelBackRightGroup.rotation.y += toRadians(180);

scene.add(carWheelFrontLeftGroup);

scene.add(carWheelFrontRightGroup);

scene.add(carWheelBackLeftGroup);

scene.add(carWheelBackRightGroup);

var carWheelGroup = new THREE.Object3D();

carWheelGroup.add(carWheelFrontLeftGroup);

carWheelGroup.add(carWheelFrontRightGroup);

carWheelGroup.add(carWheelBackLeftGroup);

carWheelGroup.add(carWheelBackRightGroup);

scene.add(carWheelGroup);

carBodyGroup.position.set(0, 1,0);

carRightDoorGroup.position.set(-1.1, 0.4, 0.8);

carLeftDoorGroup.position.set(1.1, 0.4, 0.8);

carTrunkGroup.position.set(0.1, 0.75, 1.3);

carHoodGroup.position.set(0, 0.8, -1);

var carGroup = new THREE.Object3D();

carGroup.add(carBodyGroup);

carGroup.add(carTrunkGroup);

carGroup.add(carHoodGroup);

carGroup.add(carLeftDoorGroup);

carGroup.add(carRightDoorGroup);

carGroup.add(carWheelGroup);

carGroup.add(Drone);

Drone.position.z= 0;

Drone.position.y =0;

scene.add(carGroup);

carGroup.position.y = -8;

carGroup.position.z = 6;

var render = function () {

renderer.render(scene, camera);

requestAnimationFrame(render);

CarrouselGroup.rotation.y-=(0.35\*Math.PI/180);

armGroup.rotation.z += toRadians(0.5);

sphereGroup.rotation.z += toRadians(0.5);

for(var i = 0; i < numBaskets; i++){

baskets[i].rotation.z-= toRadians(0.5);

}

//Wing rotation

WingGroup[0].rotation.z-=(100\*Math.PI/180);

WingGroup[1].rotation.z-=(100\*Math.PI/180);

WingGroup[2].rotation.z-=(100\*Math.PI/180);

WingGroup[3].rotation.z-=(100\*Math.PI/180);

//horse movement

for(var i = 0; i < (15/2); i++){

if(abajo == 1){

HorseUpGroup[i].position.y-=speed;

HorseDownGroup[i].position.y-=speed;

limites -= speed;

if(limites <0){

abajo = 0;

}

}else{

if(limites < 7){

HorseUpGroup[i].position.y+=speed;

HorseDownGroup[i].position.y+=speed;

limites += speed;

if (limites > 6){

abajo = 1;

}

}

}

}

//Drone deployment

if(deployDrone == 1){

if(Drone.position.y<=6){

Drone.position.y += 0.25;

}

}

else{

if(deployDrone == 0){

if(Drone.position.y >0 ){

Drone.position.y -= 0.25;

}

}

}

//Gyro tower movement

if (sube) {

torus.position.y += 0.25;

torusPos+=0.25;

if (torusPos==30) {

sube=false

}

}

else{

torus.position.y -= 1.0;

torusPos--;

if (torusPos==0) {

sube=true

}

}

};

render();

</script>

</body>

</html>