
Rafael Antonio Comonfort Viveros A01324276
Luis Francisco Flores Romero A01328937

# Final Project

*27/11/18*

Computer Graphics
TC3022
*August 18 Group 1*

Professor: Dr. Daniel Pérez Rojas

# Index

# Running the project

Please note that due to CORS policies, loading models and textures can be restricted by the browser being used. In order to prevent this behavior, run a local web server with the HTML directory as root and access the index file from there.

# Implementation

Most of the files follow the same layout to create one type of object. They first have the declaration of the object's constructor. It defines an "object3D" object, which is the root of all the subobjects needed for that entity. Generally a prototype function is in charge of creating the meshes to be added. This approach was chosen so objects returned can have animate functions (for example, in the amusement park objects) or additional member variables (like materials, or dimensions). The drawback is that when handling or passing the object 3D to other THREE JS functions or attempting to transform it, you have to first access the object3D member.

The "ITESM.js" file is in charge of creating and positioning all buildings. The "createITESM" function takes a Scene object as parameter so they can be added automatically. Likewise, the AmusementPark file and its function create the objects in the amusement park.

## ITESM

The creation of the campus in a 3D environment required a lot of abstract thinking. This was for the purpose of creating all different sections of the institution by finding patterns and repeating structures. The lowest form of this action, was creating a classroom that can be used across the entire scene. In our final implementation, we can find the following repeating atomic structures:

- Stairs
- Classroom
- "Magna" classroom

This approach was needed due to the fact that neither of us, had knowledge of modeling programs and tools to manually create the campus and then import it to THREE JS.

In some files, we made use of an auxiliar library to facilitate development of the object in question. THREE.CSG (CSG standing for Constructive Solid Geometry) is

a JS library that, in junction with THREE JS, allows the union, subtraction and intersection of geometries to create new ones that otherwise would have to be constructed with triangles. It allows the conversion of CSG objects to THREE Meshes and vice versa.

However, this holds true only to the “Aulas 1” and “Aulas 2” sections. Both “Cafetería Borrego” and “Centro Estudiantil/ DAE” needed a more hand-crafted approach.

## Aulas

An important realization to begin with, is the fact that both Aulas 1 and Aulas 2 are essentially the same, meaning a mirror of each other. With the exception of Aulas 1 having some floors with glass panels. We should begin by mentioning the hierarchy of the sub objects that contributed to the construction of the Aula:

- Aula: a single classroom with fixed dimensions
- Magna: a magna classroom with fixed dimensions
- Story: represents a single level of a building, which can have aulas, one aula magna, glass cover, a front and back barrier and stairs to the previous (downwards) level. All depends on the passed parameters and what level (from 1 to n) it represents. For example, the first level does not have a security barrier, the top level does not have aulas and has a different ground color, etc. However, this structure is only good in one section of an Aulas Building. Each of the latter has two of this main sections.
- Stairs: a stairs with fixed step dimensions, that can be customized with a width, and a rise (height to ascend). The file Stairs.js contains also functions to build an U-Shaped stair, and a set of those with the given amount of levels within an enclosing half-cylinder.

An aula is composed of 6 box geometries for the walls, ground and ceiling. CSG was used to create the holes for windows (which were later filled with a different box geometry with glass material) and the door. An aula magna is composed of a series of cylinders (mostly open-ended) to create the circular walls in segments. A custom theta length was used on some of them to simulate the door opening.

Stairs are made of stacked up box geometries, with the dimensions of a step. A beginning or ending platform can also be added to act as floors.

Both of the aforementioned sections of an Aulas Building were very straightforward. We simply built X amount of levels, with Y aulas each and stack them on top of each other. The challenging area was that of “Fast Fruit”/”Banderas”. For starters, there is a set of four one-classroomed levels angled by approximately 45 degrees.

Therefore, the matching of said structure required lots of manual position adjustments and parameters modification to accommodate some ground and barriers into the 45 degree-corner.

Finally, the Fast Fruit area was one that needed a thorough consideration and analysis. Said arrangement was deconstructed in two subobjects: two sets of four levels with one or three aulas each, and a central area that is enclosed by the formers. CSG once again came to the rescue, when creating the central zone. We created a sphere that was subtracted from a box in order to achieve that characteristic shape. Later, another box was subtracted to make entrances to the set of levels. As previously, due to the angle of this section, manual position tuning was required to make it look like it was joint to the rest of the building. Fast Fruit is only created for Aulas 2, as there is only one of these sections in between both Aulas.

## Cafeteria El Borrego

This building required a very specific set of geometries to complete. All glass panels were made with an open-ended Cylinder Geometry, with a material to simulate the glass appearance and that also allows visualization of the mesh from both sides. This was important, as open-ended geometries can only be seen from one side otherwise. Circle geometries were used to create roofs. Their materials had to use a polygon offset, in order to prevent overlapping surfaces to flicker and instead draw the appropriate one. The metal arc roof was achieved with a ring geometry with only a given theta (angle) length and a wireframe white material to simulate the metal bars. Finally, the southern concrete section was a Cylinder Geometry with again, a custom theta length.

## Centro Estudiantil

This building is the simplest in our project. It is simply conformed by a main box structure as body, another one as top, and 3 additionals to create the side terrace that represents the Limoneto’s terrace. CSG was used to make a whole in the body’s center, which was then used to place a large window (a box Geometry with a glass-like material).
It took some tinkering around to find out the exact sizes, and placement positions of each object, but overall, was a quite straight-forward process.

# Car

The implementation of the 3D car put challenges in this project. This is because designing a car referencing an existing model would have consumed too much time. Instead of designing a car from scratch, it was decided to use an existing model[1], so that effort could be focused in the functionality of the car and not the design. The objects from this model were imported into the project using the MTLLoader and glTFLoader. Since the obtained car model didn't match all requirements, it was needed to break the model apart and isolate car components so they could be implemented into a functional unit. These components are:

- Wheel
- Door
- Hood
- Trunk
- Body

## Wheel

Wheels[2] are based on the Mitsubishi Evolution X wheels. The wheel was isolated in Blender and then exported to obj/MTL for THREE JS. Once imported into the project a single wheel was instantiated four times and attached to the car group. For steering direction of the car, the front wheels are rotated.

## Door

The door was isolated from the main model and textures were applied to it. For the window to be transparent, the original model had to be modified, since the renderer used was not appropriate and the front door window was attached to the rear door window. Since the surfaces in the original model only included normals for exterior faces, additional polygons had to be added, so the door had an internal face too.

Once isolated, the door was exported to obj/MTL for THREE JS. The door is imported into the project using the MTL Loader and instantiated twice so both front doors are present. These doors are joined to the car group by their front end and rotation for opening and closing occurs in that point.

---

[1] Model by HomiDiFerru from TurboSquid.
[2] From a different model, by Vladivpg.

## Hood

Similarly to the doors, the hood was isolated from the car model and applied textures into it. An additional polygon was also needed for this component for the same reason as for the door component. The hood was exported to obj/MTL and added to the project, where it is attached to the car group by the upper rear end, where rotation occurs.

Opening and closing the hood is implemented within the car object, using a openHood and closeHood function. For theses functions to behave correctly, the maximum and minimum hood angles where set.

## Trunk

The trunk component was isolated from the main model and applied textures to it. Due to the form factor of this component, its inner surface was duplicated from the original surface and made normal-reversed. This component was then exported to obj/MTL for THREE JS.

In the project, this component is attached to the car group by its upper front end, where it rotates to allow for opening and closing operations. This functionality is implemented exactly as in the hood component.

## Body

Most of the remaining parts of the model were kept for the body structure of the car. The process was very similar to the one used for other components: the already implemented car components were removed and the remaining parts were applied materials and textures. The windows were broken apart so they could be detached from the front doors windows. These parts were exported into a single obj/MTL file for THREE JS.

In the project, this body component was added to the car group and manipulated (resizing, translating, rotating) for it to fit the other components. For the car to be able to move, the forward direction angle is stored in the car object, along with the steering offset angle of the wheels. Both of these angles are used to compute the displacement vector on each step of the goForward behavior. Additionally, the car is rotated according to the steering angle so it simulates real-car movement.

## Drone

The drone is implemented based on two models[3] (one for the body, one for the propellers) joined together in THREE JS. The drone object is attached to the car object so they can move together. The propellers animation is triggered by the position of the drone relative to the car.

# Appendices

## Appendix A: Code

### styles.css

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*24.oct.2018*/

/*Set dark document background*/
html {
      padding: 0;
      border: none;
      margin: 0;

      background-color: rgb(50, 50, 50);
}

/*Make the canvas use the whole viewport*/
body {
      padding: 0;
      border: none;
      margin: 0;
}

canvas {
      padding: 0;
      border: none;
      margin: 0;

      width: 100%;
      height: 100%;
}
```

[3] By 3DHaupt and Dmytro Kotliar.

## index.html

```
<!-- A01324276    Rafael Antonio Comonfort Viveros -->
<!-- A01328937    Luis Francisco Flores Romero -->
<!-- 25.nov.2018 -->
<!DOCTYPE html />
<html>
     <head>
          <meta charset="UTF-8" />
          <link      rel="stylesheet"      type      ="text/css"
href="./styles.css" />
          <title>Final Project</title>
     </head>
     <body>
          <!--      Import libraries library -->
          <script src="three.min.js"></script>
          <script src="Three.CSG.js"></script>
          <!--      Import loaders -->
          <script src="GLTFLoader.js"></script>
          <script src="OBJLoader.js"></script>
          <script src="MTLLoader.js"></script>

          <!--      Import utility functions -->
          <script src="util.js"></script>
          <!--      Import materials -->
          <script src="materials.js"></script>


          <!--      Import models -->
          <script src="car.js"></script>
          <script src="drone.js"></script>

          <script src="Aula.js"></script>
          <script src="Magna.js"></script>
          <script src="Stairs.js"></script>
          <script src="DAE.js"></script>
          <script src="Cafeteria.js"></script>
          <script src="Story.js"></script>
          <script src="AulasBuilding.js"></script>
          <script src="FastFruitArea.js"></script>
          <script src="ITESM.js"></script>
          <script src="Carousel.js"></script>
          <script src="DropTower.js"></script>
          <script src="FerrisWheel.js"></script>
          <script src="ferris.js"></script>
```

```
            <script src="AmusementPark.js"></script>

            <!--       Import script -->
            <script src="custom.js"></script>
      </body>
</html>
```

## AmusementPark.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/

/**
 * Creates an amusement park in the given scene
 * @param {*} scene Scene from THREE JS that the amusement park
will be added to
 */
function AmusementPark(scene) {

    this.object3D = new THREE.Object3D();

    this.carousel = new Carousel();
    this.carousel.object3D.position.set(0, 0, -10);
    this.object3D.add(this.carousel.object3D);

    this.wheelSlim = new FerrisWheelSlim( 40, 24, 0.4, 4.5 );
    this.wheelSlim.mesh.position.set( -110, 0, -50 );
    this.wheelSlim.mesh.rotation.y += Math.PI / 5;
    this.object3D.add(this.wheelSlim.mesh);

    this.wheel = new FerrisWheel();
    this.wheel.object3D.position.set(30, 0, -25);
    this.wheel.object3D.rotation.y -= Math.PI / 5;
    this.object3D.add(this.wheel.object3D);

    this.dropTower = new DropTower();
    this.dropTower.object3D.position.set(-30, 0, -30);
    this.object3D.add(this.dropTower.object3D);

    this.object3D.position.set(-40, 0, -90);
    this.object3D.scale.set(.48, .48, .48);
    scene.add(this.object3D);

    this.animate = function() {
```

```
        this.wheelSlim.rotate(-0.2);
        this.wheel.animate();
        this.carousel.animate();
        this.dropTower.animate();
    }
}
```

## Aula.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
function Aula() {
    this.object3D = this.makeAula();
}

/**
 * Constructs an aula object
 * @returns An Object3D containing all Aula hierarchy
 */
Aula.prototype.makeAula = function() {

     //Materials
    var glassMat = new THREE.MeshPhongMaterial ({
        color: 0x19c1a8,
        specular: 0x555555,
        transparent: true,
        opacity: 0.3,
           shininess: 20,
           side: THREE.DoubleSide
     });
     var concreteMat = new THREE.MeshLambertMaterial ({
           color: 0xbabcbc,
           side: THREE.DoubleSide
    });

    //Measures
    let groundX = 8;
    let groundY = 0.3;
    let groundZ = 9;
    let latWallX = groundX;
    let latWallY = 2.75;
    let latWallZ = .25;
    let backWallX = latWallZ;
    let backWallY = latWallY;
```

```
    let backWallZ = groundZ;

    let bigWindowsCutX = latWallX - backWallX ;
    let bigWindowsCutY = latWallY * .6;
    let bigWindowsCutZ = latWallZ * 2;

    let windowsCutX = latWallX *.62;
    let windowsCutY = latWallY * .2;
    let windowsCutZ = latWallZ * 2;

    let doorCutX = latWallX * .19;
    let doorCutY = latWallY * .75;
    let doorCutZ = latWallZ * 2;

    let windowSmallX = windowsCutX;
    let windowSmallY = windowsCutY;
    let windowSmallZ = 0.05;
    let windowBigX = bigWindowsCutX;
    let windowBigY = bigWindowsCutY;
    let windowBigZ = 0.05;

    //Geometries
    let groundGeom = new THREE.BoxGeometry(groundX, groundY,
groundZ);
    let latWallGeom = new THREE.BoxGeometry(latWallX, latWallY,
latWallZ);
    let backWallGeom = new THREE.BoxGeometry(backWallX, backWallY,
backWallZ);

    let bigWindowsCutGeom = new THREE.BoxGeometry(bigWindowsCutX,
bigWindowsCutY, bigWindowsCutZ);
    let windowsCutGeom = new THREE.BoxGeometry(windowsCutX,
windowsCutY, windowsCutZ);
    let doorCutGeom = new THREE.BoxGeometry(doorCutX, doorCutY,
doorCutZ);
    let windowSmallGeom = new THREE.BoxGeometry(windowSmallX,
windowSmallY, windowSmallZ);
    let windowBigGeom = new THREE.BoxGeometry(windowBigX,
windowBigY, windowBigZ);

    //Meshes
    let ground = new THREE.Mesh(groundGeom);
    let ceiling = ground.clone();
    let backWall = new THREE.Mesh(backWallGeom);
    let frontWall = backWall.clone();
    let doorWall = new THREE.Mesh(latWallGeom);
```

```
    let windowsWall = doorWall.clone();

    let bigWindowsCut = new THREE.Mesh(bigWindowsCutGeom);
    let windowsCut = new THREE.Mesh(windowsCutGeom);
    let doorCut = new THREE.Mesh(doorCutGeom);
    let windowSmall = new THREE.Mesh(windowSmallGeom, glassMat);
    let windowBig = new THREE.Mesh(windowBigGeom, glassMat);

    //Offsets
    ground.position.y -= groundY/2;
    backWall.position.y = frontWall.position.y =
doorWall.position.y = windowsWall.position.y = backWallY / 2;
    ceiling.position.y += backWallY + groundY / 2;
    backWall.position.x = groundX / 2 - backWallX / 2;
    frontWall.position.x -= groundX / 2 - backWallX / 2;
    doorWall.position.z += groundZ / 2 - latWallZ / 2;
    windowsWall.position.z -= groundZ / 2 - latWallZ / 2;

    bigWindowsCut.position.z =  windowsWall.position.z;
    bigWindowsCut.position.y =  latWallY *.37 + bigWindowsCutY /
2;
    windowsCut.position.z = doorWall.position.z;
    windowsCut.position.y = latWallY * .9 - windowsCutY / 2;
    windowsCut.position.x = -latWallX / 2 + backWallX * 1.6 +
windowsCutX / 2;
    doorCut.position.z = doorWall.position.z;
    doorCut.position.y = doorCutY / 2;
    doorCut.position.x = latWallX / 2 - backWallX * 1.6 - doorCutX
/ 2;
    windowSmall.position.copy(windowsCut.position);
    windowBig.position.copy(bigWindowsCut.position);

    //Create aula using CSG
    var classroomCSG = THREE.CSG.fromMesh (ground);
    classroomCSG = classroomCSG.union(THREE.CSG.fromMesh
(ceiling));
    classroomCSG = classroomCSG.union(THREE.CSG.fromMesh
(backWall));
    classroomCSG = classroomCSG.union(THREE.CSG.fromMesh
(frontWall));
    classroomCSG = classroomCSG.union(THREE.CSG.fromMesh
(doorWall));
    classroomCSG = classroomCSG.union(THREE.CSG.fromMesh
(windowsWall));
```

```
    classroomCSG = classroomCSG.subtract(THREE.CSG.fromMesh
(bigWindowsCut));
    classroomCSG = classroomCSG.subtract(THREE.CSG.fromMesh
(windowsCut));
    classroomCSG = classroomCSG.subtract(THREE.CSG.fromMesh
(doorCut));

    let classroom = THREE.CSG.toMesh(classroomCSG, concreteMat);

    var aula = new THREE.Object3D();
    aula.add(classroom, windowSmall, windowBig);

    for (var obj of aula.children) {
        obj.position.z -= groundZ / 2;
    }

    return aula;
}
```

## AulasBuilding.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
function AulasBuilding(aulasBuildingNum) {
    this.rightSideLevels = [];
    this.leftSideLevels = [];
    this.orphanLevels = [];
    this.object3D = this.makeAulasBuilding(aulasBuildingNum);
}

/**
 * Constructs an aula building object
 * @param {Number} aulasBuildingNum 1 or 2, depending on which
aula we want
 * @returns An Object3D containing all hierarchy
 */
AulasBuilding.prototype.makeAulasBuilding =
function(aulasBuildingNum) {

    var aulasBuilding = new THREE.Object3D();

    let stairsDiameter = 8;
    let middleRotationAngle = - Math.PI / 6;
```

```
    let aulaLengthX = (new Story(1, false, false, 1,
false).levelLength) * Math.cos(middleRotationAngle);
    let leftSideZ = - 2 * 14;

    //Checkpoints (where main sections of AulasBuilding should be)
    let rightMainStairsX = 8;
    let rightSideX = rightMainStairsX + (new Story(1, false, true,
4, true).levelLength) / 2 + 11.5;
    let orphanAulasX = -rightMainStairsX - aulaLengthX / 2;
    let leftMainStairsX = orphanAulasX - aulaLengthX;
    let leftSideX = leftMainStairsX - (new Story(1, false, true,
4, true).levelLength) / 2 - .5;

    //Create rightSide (where the labs were) of Aulas2
    for (var i = 0; i < 4; i++) {
        var level = new Story(i + 1, false, i == 3, 4, true, true,
true);
        this.rightSideLevels.push(level);

        level.object3D.position.y = level.levelHeight * i;
        level.object3D.position.x = rightSideX;
        level.object3D.position.z = -8;
        aulasBuilding.add(level.object3D);
    }

    //Create middle orphan levels (that do not belong to the left
or right sections of an Aulas Building)
    for (var i = 0; i < 5; i++) {
        var level = new Story(i + 1, aulasBuildingNum == 1, false,
1, false);
        this.orphanLevels.push(level);

        level.object3D.position.y = level.levelHeight * i;
        level.object3D.position.x = orphanAulasX;
        level.object3D.position.z = leftSideZ + 2.5;
        level.object3D.rotation.y += middleRotationAngle;
        aulasBuilding.add(level.object3D);
    }

    //Create fast fruit/banderas only once
    if (aulasBuildingNum == 2)
    {
        var ffArea = makeFastFruitArea();
        ffArea.rotation.y += middleRotationAngle;
        ffArea.position.y = 3.25 * 5/2 + .1;
        ffArea.position.x -= 4.28;
```

```
        ffArea.position.z += 2.18;
        aulasBuilding.add(ffArea);
    }

    //Create left stairs
    let leftMainStairs = makeAulaStairs(5);
    leftMainStairs.position.z = leftSideZ;
    leftMainStairs.position.x = leftMainStairsX;
    aulasBuilding.add(leftMainStairs);

    //Create leftSide (parallel to OXXO?)
    for (var i = 0; i < 6; i++) {
        var level = new Story(i + 1, aulasBuildingNum == 1, i ==
5, 4, true, true, true);
        this.leftSideLevels.push(level);

        level.object3D.position.y = level.levelHeight * i;
        level.object3D.position.x = leftSideX;
        level.object3D.position.z = leftSideZ;
        level.object3D.scale.x *= -1;
        aulasBuilding.add(level.object3D);
    }

    return aulasBuilding;
}
```

## Cafeteria.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/

/**
 * Constructor for a new cafeteria
 */
function Cafeteria() {
     this.object3D = this.makeCafeteria();
}

/**
 * Constructs a cafeteria building object
 * @returns An Object3D containing all hierarchy
 */
Cafeteria.prototype.makeCafeteria = function() {
```

```
//Materials
var glassMat = new THREE.MeshPhongMaterial ({
    color: 0x19c1a8,
    specular: 0x555555,
    transparent: true,
    opacity: 0.3,
       shininess: 20,
       side: THREE.DoubleSide
 });
 var concreteMat = new THREE.MeshLambertMaterial ({
       color: 0xbabcbc,
       side: THREE.DoubleSide,
       polygonOffset: true,
    polygonOffsetFactor: 0.05
 });
 var waterProofMat = new THREE.MeshLambertMaterial ({
       color: 0xaa2001,
       side: THREE.DoubleSide,
       polygonOffset: true,
    polygonOffsetFactor: -0.01
 });
 var ringMat = new THREE.MeshLambertMaterial ({
       color: 0xffffff,
       side: THREE.DoubleSide,
       wireframe: true
 });
 var floorMat = new THREE.MeshLambertMaterial ({
       color: 0xd16f00,
       side: THREE.DoubleSide
 });


 //Measures
 let cafeteriaRadius = 20;
 let cafeteriaHeight = 5;
 let outerRadius = cafeteriaRadius * 1.2;
 let thetaLength = THREE.Math.degToRad(230);
 let middleRadius = cafeteriaRadius / 2;
 let middleHeight = cafeteriaHeight;
 let topRadius = middleRadius / 3;
 let topHeight = middleHeight / 2;

 let complementTheta = 2 * Math.PI - thetaLength;
 let wallY = cafeteriaHeight * 2;
 let wallX = 0.005;
 let wallZ = outerRadius - cafeteriaRadius;
```

```
      //Geometries
      let firstFloorGeom = new
THREE.CylinderGeometry(cafeteriaRadius, cafeteriaRadius,
cafeteriaHeight, 16, 1, true, 0, thetaLength);
      let middleGeom = new THREE.CylinderGeometry(middleRadius,
middleRadius, middleHeight, 16, 1, true);
      let topGeom = new THREE.CylinderGeometry(topRadius,
topRadius, topHeight, 16, 1, true);
      let ringGeom = new THREE.RingGeometry(cafeteriaRadius,
outerRadius, 10, 6, -complementTheta - .17, thetaLength);
      let floorGeom = new THREE.CircleGeometry(outerRadius, 32);

      let mainHoodGeom = new THREE.CircleGeometry(cafeteriaRadius,
32);
      let middleHoodGeom = new THREE.CircleGeometry(middleRadius,
32);
      let topHoodGeom = new THREE.CircleGeometry(topRadius, 32);
      let complementGeom = new THREE.CylinderGeometry(outerRadius,
outerRadius, 3 * cafeteriaHeight, 32, 4, false, - complementTheta,
complementTheta);
      let wallGeom = new THREE.BoxGeometry(wallX, wallY, wallZ);

      //Meshes
      let firstFloorGlass = new THREE.Mesh(firstFloorGeom,
glassMat);
      let firstFloorConcrete = new THREE.Mesh(firstFloorGeom,
concreteMat);
      let middleFloor = new THREE.Mesh(middleGeom, glassMat);
      let topFloor = new THREE.Mesh(topGeom, glassMat);
      let ring = new THREE.Mesh(ringGeom, ringMat);
      let floor = new THREE.Mesh(floorGeom, floorMat);

      let mainHood = new THREE.Mesh(mainHoodGeom, waterProofMat);
      let middleHood = new THREE.Mesh(middleHoodGeom, concreteMat);
      let topHood = new THREE.Mesh(topHoodGeom, concreteMat);
      let complement = new THREE.Mesh(complementGeom, concreteMat);
      let wall1 = new THREE.Mesh(wallGeom, concreteMat);
      let wall2 = new THREE.Mesh(wallGeom, concreteMat);


      //Placement
      var cafeteria = new THREE.Object3D();
      cafeteria.add(firstFloorGlass, firstFloorConcrete,
middleFloor, topFloor, ring, complement, floor, mainHood,
middleHood, topHood, wall1, wall2);
```

```
	firstFloorGlass.position.y = cafeteriaHeight / 2;
	firstFloorConcrete.position.y = firstFloorGlass.position.y +
cafeteriaHeight;
	middleFloor.position.y = firstFloorConcrete.position.y +
cafeteriaHeight / 2 + middleHeight / 2;
	topFloor.position.y = middleFloor.position.y + middleHeight /
2 + topHeight / 2;
	ring.position.y = cafeteriaHeight;
	ring.rotation.x += Math.PI / 2;
	floor.rotation.x += Math.PI / 2;

	mainHood.position.y = firstFloorConcrete.position.y +
cafeteriaHeight / 2;
	middleHood.position.y = middleFloor.position.y + middleHeight
/ 2;
	topHood.position.y = topFloor.position.y + topHeight / 2;
	mainHood.rotation.x += Math.PI / 2;
	middleHood.rotation.x += Math.PI / 2;
	topHood.rotation.x += Math.PI / 2;
	complement.position.y = (3 *cafeteriaHeight /2) -
cafeteriaHeight;
	wall1.position.y = wall2.position.y = cafeteriaHeight;
	wall1.position.z += cafeteriaRadius + wallZ / 2;
	wall2.rotation.y += THREE.Math.degToRad(50);
	wall2.translateZ(-(cafeteriaRadius + wallZ / 2));

	return cafeteria;
}
```

## car.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
const loader = new THREE.GLTFLoader();
const mtlLoader = new THREE.MTLLoader();

//   Import door and scale
//   passenger indicades whether the door will be the passenger's
door (-1/1)
function Door( passenger ) {
	this.mesh = new THREE.Object3D();
	this.group = new THREE.Object3D();
```

```
	this.init = () => {
		mtlLoader.setTexturePath('./objects/');
		mtlLoader.setPath('./objects/');
		mtlLoader.load('door.mtl', (materials) => {
			materials.preload();
			let objLoader = new THREE.OBJLoader();
			objLoader.setMaterials(materials);
			objLoader.setPath('./objects/');
			objLoader.load('door.obj', (object) => {
				this.group.add( object );
			});
		});
	};
	this.group.scale.set( 0.7, 0.7, 0.7 );
	this.group.rotation.y = toRad(90);

	this.mesh.add( this.group );
	this.mesh.position.x = 0.76;
	this.mesh.position.y = 0.43;
	this.mesh.position.z = 0.83 * passenger;

	this.init();
}

//	Import hood and scale
function Hood() {
	this.mesh = new THREE.Object3D();
	this.group = new THREE.Object3D();

	this.init = () => {
		mtlLoader.setTexturePath('./objects/');
		mtlLoader.setPath('./objects/');
		mtlLoader.load('hood.mtl', (materials) => {
			materials.preload();
			let objLoader = new THREE.OBJLoader();
			objLoader.setMaterials(materials);
			objLoader.setPath('./objects/');
			objLoader.load('hood.obj', (object) => {
				this.group.add( object );
			});
		});

		this.group.rotation.y = toRad(90);
		this.group.scale.set( 0.7, 0.7, 0.7 );

		this.mesh.add( this.group );
```

```
            this.mesh.position.x = 1.05;
            this.mesh.position.y = 0.65;
      };

      this.init();
}

//    Import trunk and scale
function Trunk() {
      this.mesh = new THREE.Object3D();
      this.group = new THREE.Object3D();

      this.init = () => {
            mtlLoader.setTexturePath('./objects/');
            mtlLoader.setPath('./objects/');
            mtlLoader.load('trunk.mtl', (materials) => {
                  materials.preload();
                  let objLoader = new THREE.OBJLoader();
                  objLoader.setMaterials(materials);
                  objLoader.setPath('./objects/');
                  objLoader.load('trunk.obj', (object) => {
                        this.group.add( object );
                  });
            });
            this.group.rotation.y = toRad(90);
            this.group.scale.set( 0.7, 0.7, 0.7 );

            this.mesh.add( this.group );
            this.mesh.position.x = -1.63;
            this.mesh.position.y = 0.76;
      };

      this.init();
}

//    Import wheel
function Wheel() {
      this.mesh = new THREE.Object3D();

      this.init = () => {
            loader.load( './wheel.glb', ( gltf ) => {
                  this.mesh.add( gltf.scene );
            }, undefined, function ( error ) {
                  console.error( error );
            } );
      };
```

```
	this.init();
}

//	Import car body and scale
function Body() {
	this.mesh = new THREE.Object3D();

	this.init = () => {
		mtlLoader.setTexturePath('./objects/');
		mtlLoader.setPath('./objects/');
		mtlLoader.load('body.mtl', (materials) => {
			materials.preload();
			let objLoader = new THREE.OBJLoader();
			objLoader.setMaterials(materials);
			objLoader.setPath('./objects/');
			objLoader.load('body.obj', (object) => {
				this.mesh.add( object );
			});
		});

		this.mesh.rotation.y = toRad(90);
		this.mesh.scale.set( 0.7, 0.7, 0.7 );
		this.mesh.position.x = -0.33;
		this.mesh.position.y = -0.27;
	};

	this.init();
}

//	Assemble car object with components
//	This car saves forward direction as an angle.
//	Wheels direction is saved as an offset angle from the forward
direction.
function Car() {
	this.wheel_fr;
	this.wheel_fl;
	this.wheel_rr;
	this.wheel_rl;

	//	Actionable components
	this.doorR;
	this.doorL;
	this.hood;
	this.trunk;
	this.drone;
```

```
//   Control flags
this.doorsClosed = true;
this.doorsOpen = false;
this.hoodClosed = true;
this.hoodOpen = false;
this.trunkClosed = true;
this.trunkOpen = false;

this.mesh = new THREE.Object3D();
this.forwardAngle;   // Degrees
this.steeringOffset; // Degrees
this.maxSteeringDelta;     // Degrees

this.init = () => {
      let axisToAxis = 2.53;
      let wheelToWheel = 1.4;

      //   Attach wheels
      this.wheel_fr = new Wheel();
      this.mesh.add( this.wheel_fr.mesh );
      this.wheel_fr.mesh.rotation.y = toRad(180);
      this.wheel_fr.mesh.position.x = axisToAxis / 2;
      this.wheel_fr.mesh.position.z = wheelToWheel / 2;

      this.wheel_fl = new Wheel();
      this.mesh.add( this.wheel_fl.mesh );
      this.wheel_fl.mesh.position.x = axisToAxis / 2;
      this.wheel_fl.mesh.position.z = -wheelToWheel / 2;

      this.wheel_rr = new Wheel();
      this.mesh.add( this.wheel_rr.mesh );
      this.wheel_rr.mesh.rotation.y = toRad(180);
      this.wheel_rr.mesh.position.x = -axisToAxis / 2;
      this.wheel_rr.mesh.position.z = wheelToWheel / 2;

      this.wheel_rl = new Wheel();
      this.mesh.add( this.wheel_rl.mesh );
      this.wheel_rl.mesh.position.x = -axisToAxis / 2;
      this.wheel_rl.mesh.position.z = -wheelToWheel / 2;

      //   Attach body
      let b = new Body();
      this.mesh.add( b.mesh );

      //   Attach hood
```

```
            this.hood = new Hood();
            this.mesh.add( this.hood.mesh );

            //    Attach trunk
            this.trunk = new Trunk();
            this.mesh.add( this.trunk.mesh );

            //    Attach doors
            this.doorR = new Door( 1 );
            this.mesh.add( this.doorR.mesh );
            this.doorR.mesh.scale.set( 1, 1, -1 );

            this.doorL = new Door( -1 );
            this.mesh.add( this.doorL.mesh );

            //    Attach drone
            this.drone = new Drone();
            this.mesh.add( this.drone.mesh );

            //    Initialize car parameters
            this.mesh.position.y = 0.33;
            this.forwardAngle = 0;
            this.steeringOffset = 0;
            this.maxSteeringDelta = 30;
      }

      //    CONTROL FUNCTIONS

      //    Make the car go forward in the direction of the wheels
      this.goForward = (step) => {
            //    Compute wheels absolute angle
            let wheelsDir = this.forwardAngle + this.steeringOffset;
            //    Compute object translation
            let dx = step * Math.cos(toRad(wheelsDir));
            let dz = step * Math.sin(toRad(wheelsDir));
            //    Translate the object
            this.mesh.position.x += dx;
            this.mesh.position.z -= dz;

            //    Compute new object rotation
            let newRotation = toRad(this.steeringOffset * 0.5) *
step;
            //    Rotate the object
            this.mesh.rotation.y += newRotation;
            //    Update forward direction angle
            this.forwardAngle = toDeg(this.mesh.rotation.y);
```

```
            //   Rotate wheels to simulate car movement
            this.wheel_fr.mesh.rotation.z += toRad(step * 60);
            this.wheel_fl.mesh.rotation.z -= toRad(step * 60);
            this.wheel_rr.mesh.rotation.z += toRad(step * 60);
            this.wheel_rl.mesh.rotation.z -= toRad(step * 60);
      };

      //   Make the car turn its front wheels
      this.steer = (angle) => {
            if(Math.abs(this.steeringOffset + angle) <=
                  Math.abs( this.maxSteeringDelta )) {
                        this.steeringOffset = this.steeringOffset +
angle;
                        this.wheel_fr.mesh.rotation.y +=
toRad(angle);
                        this.wheel_fl.mesh.rotation.y +=
toRad(angle);
            }
      };

      //   Open the car doors one step
      this.openDoors = () => {
            if(!this.doorsOpen) {
                  this.doorL.mesh.rotation.y -= 0.1;    // Left door
is mirrored
                  this.doorR.mesh.rotation.y += 0.1;
                  this.doorsClosed = false;
            }

            if(this.doorL.mesh.rotation.y <= -1 ||
this.doorR.mesh.rotation.y >= 1) {
                  this.doorsOpen = true;
            }
      };

      //   Close the car doors one step
      this.closeDoors = () => {
            if(!this.doorsClosed) {
                  this.doorL.mesh.rotation.y += 0.1;
                  this.doorR.mesh.rotation.y -= 0.1;
                  this.doorsOpen = false;
            }

            if(this.doorL.mesh.rotation.y >= 0.05 ||
this.doorR.mesh.rotation.y <= 0.05) {
```

```
            this.doorsClosed = true;
      }
};

//    Open the car hood one step
this.openHood = () => {
      if(!this.hoodOpen) {
            this.hood.mesh.rotation.z += 0.1;
            this.hoodClosed = false;
      }

      if(this.hood.mesh.rotation.z >= 1)
            this.hoodOpen = true;
};

//    Close the car hood one step
this.closeHood = () => {
      if(!this.hoodClosed) {
            this.hood.mesh.rotation.z -= 0.1;
            this.hoodOpen = false;
      }

      if(this.hood.mesh.rotation.z <= 0.05)
            this.hoodClosed = true;
};

//    Open the car trunk one step
this.openTrunk = () => {
      if(!this.trunkOpen) {
            this.trunk.mesh.rotation.z -= 0.1;
            this.trunkClosed = false;
      }

      if(this.trunk.mesh.rotation.z <= -1)
            this.trunkOpen = true;
};

//    Close the car trunk one step
this.closeTrunk = () => {
      if(!this.trunkClosed) {
            this.trunk.mesh.rotation.z += 0.1;
            this.trunkOpen = false;
      }

      if(this.trunk.mesh.rotation.z >= -0.05)
            this.trunkClosed = true;
```

```
	};

	//	Animate the drone object
	this.animateDrone = () => {
		this.drone.animate();
	}

	//	Release/attach the drone
	this.toggleDrone = () => {
		if(this.droneActivated)
			this.drone.settle();
		else
			this.drone.activate();

		this.droneActivated = !this.droneActivated;
	};

	this.init();
}
```

## Carousel.js

```
/*A01328937	Luis Francisco Flores Romero*/
/*A01324276	Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
function Carousel() {

	this.object3D = this.makeCarousel();
	this.angularVelocity = 0.005;

	this.animate = function () {
		this.object3D.children[1].rotation.y +=
this.angularVelocity;

		var animals =
this.object3D.children[1].children[2].children;

		let middlePoint =
this.object3D.children[1].children[1].children[0].position.y /
2.4;
		for (var i = 0; i < animals.length; i++) {
			var moveDir;
			if (i % 2 == 0)
				moveDir = 1;
			else
```

```
                    moveDir = -1;

                animals[i].position.y = middlePoint + moveDir *
Math.sin(this.object3D.children[1].rotation.y * 3);
            }
        }
}

/**
 *    Carousel (object3D)
 *      fixedG
 *              base
 *              column
 *              top
 *         movingG
 *              hidingRing
 *              polesG
 *              figuresG
 */
Carousel.prototype.makeCarousel = function () {

    //Materials
    let redMat = new THREE.MeshLambertMaterial({color:
0xaa0000});
    let greyMat = new THREE.MeshLambertMaterial({color:
0x595e5d});
    let blueMat = new THREE.MeshLambertMaterial({color:
0x012256});

    //Measures
    let baseRadius = 15;
    let baseHeight = .5;
    let columnRadius = 3;
    let columnHeight = 13;
    let topRadius = baseRadius * 1.1;
    let hidingRingHeight = 2;
    let hidingRingRadius = baseRadius * .341;
    let topHeight = 4.5;
    let dogScale = 1;

    let circles = 2;
    let animalsPerCircle = 8;

    //groups
    let carousel = new THREE.Object3D();
    let fixedG = new THREE.Object3D();
```

```
      let polesG = new THREE.Object3D();
      let movingG = new THREE.Object3D();
      let dogsG = new THREE.Object3D();

      //Objects
      let baseGeom = new THREE.CylinderGeometry(baseRadius,
baseRadius, baseHeight, 32);
      let base = new THREE.Mesh(baseGeom, blueMat);

      let columnGeom = new THREE.CylinderGeometry(columnRadius,
columnRadius, columnHeight, 32);
      let column = new THREE.Mesh(columnGeom, blueMat);

      var top = createUmbrellaObject(topRadius, topHeight, redMat);

      let hidingRingGeom = new
THREE.CylinderGeometry(hidingRingRadius, hidingRingRadius,
hidingRingHeight, 32);
      let hidingRing = new THREE.Mesh(hidingRingGeom, greyMat);

      //Make the circunferences of animals
      let poleGeom = new THREE.CylinderGeometry(0.07, 0.07,
columnHeight, 8);
      for (var i = 0; i < circles; i++) {

            var angleStep = Math.PI * 2.0 / animalsPerCircle;
            var startAngle = i * angleStep / 2.0;
            var targetRadius = columnRadius + (i + 1) * (baseRadius
- columnRadius) / (circles + 1);

            for (var j = 0; j < animalsPerCircle; j++) {
                  var angle = startAngle + j * angleStep;

                  //Create animal i's pole
                  let pole = new THREE.Mesh(poleGeom, greyMat);
                  polesG.add(pole);
                  pole.position.set(targetRadius * Math.cos(angle),
columnHeight / 2.0 - (i * .8125 + .8125), targetRadius *
Math.sin(angle));

                  //Create animal i
                  let dog = createDogObject();
                  dogsG.add(dog);
                  dog.scale.set(dogScale, dogScale, dogScale);
                  dog.position.set(targetRadius * Math.cos(angle),
pole.position.y / 2, targetRadius * Math.sin(angle));
```

```
                dog.rotation.y = Math.PI - angle;
            }
        }
        //Create hierarchy
        carousel.add(fixedG, movingG);
        fixedG.add(base, column, top);
        movingG.add(hidingRing, polesG, dogsG);

        //Offset objects
        base.position.y = baseHeight / 2.0;
        column.position.y = baseHeight + columnHeight / 2.0;
        top.position.y = baseHeight - 2.5;
        hidingRing.position.y = column.position.y * 2 -
hidingRingHeight;

        return carousel;
}

//For the top of the Carousel
function createUmbrellaObject(radius, height, material) {

        let poleFactor = 3.5;

        material.side = THREE.DoubleSide;

        let mainGeom = new THREE.ConeGeometry(radius, height, 16, 4,
true);
        let main = new THREE.Mesh(mainGeom, material);

        let poleGeom = new THREE.CylinderGeometry(radius / 14, radius
/ 14, height * poleFactor, 32);
        let pole = new THREE.Mesh(poleGeom, material);

        var umbrella = new THREE.Object3D();
        umbrella.add(main, pole);

        pole.position.y = height * poleFactor / 2;
        main.position.y = height * poleFactor * 0.9;

        return umbrella;
}

//Animal that can be ridden
function createDogObject() {
```

```
      let color1 = new THREE.MeshLambertMaterial({color:
Math.random() * 0xffffff});
      let color2 = new THREE.MeshLambertMaterial({color:
Math.random() * 0xffffff});


      var legGeom = new THREE.CubeGeometry(.25, .9, .25);
      var leftFrontLeg = new THREE.Mesh(legGeom, color1);
      leftFrontLeg.position.set(-0.2, 0, 0);

      var rightFrontLeg = new THREE.Mesh(legGeom, color1);
      rightFrontLeg.position.set(0.2, 0, 0);

      var leftBackLeg = new THREE.Mesh(legGeom, color1);
      leftBackLeg.position.set(-0.2, 0, -1.5);
      var rightBackLeg = new THREE.Mesh(legGeom, color1);
      rightBackLeg.position.set(0.2, 0, -1.5);

      var bodyGeom = new THREE.CubeGeometry(.75, .75, 1.95);
      var body = new THREE.Mesh(bodyGeom, color2);
      body.position.set(0, .75, -.73);

      var maneGeom = new THREE.CubeGeometry(.8, 0.8, 0.6);
      var mane = new THREE.Mesh(maneGeom, color1);
      mane.position.set(0, .8, 0);

      var headGeom = new THREE.CubeGeometry(.5, .5, .5);
      var head = new THREE.Mesh(headGeom, color2);
      head.position.set(0, .85, .35);

      var muzzleGeom = new THREE.CubeGeometry(.2, .17, .3);
      var muzzle = new THREE.Mesh(muzzleGeom, color2);
      muzzle.position.set(0, .73, .7);

      var earGeom = new THREE.CubeGeometry(.17, .17, .05);
      var ear = new THREE.Mesh(earGeom, color2);
      ear.position.set(-.15, 1.2, .35);

      var ear2 = new THREE.Mesh(earGeom, color2);
      ear2.position.set(.15, 1.2, .35);

      var dogGroup = new THREE.Object3D();
      dogGroup.add(leftFrontLeg);
      dogGroup.add(rightFrontLeg);
      dogGroup.add(leftBackLeg);
      dogGroup.add(rightBackLeg);
```

```
      dogGroup.add(body);
      dogGroup.add(mane);
      dogGroup.add(head);
      dogGroup.add(muzzle);
      dogGroup.add(ear);
      dogGroup.add(ear2);

      for (var i = 0; i < dogGroup.children.length; i++) {
            dogGroup.children[i].position.z += 0.5;
      }

      return dogGroup;
}
```

## custom.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/

//    ************************************
//    GLOBAL VARIABLES
//    ************************************
let scene, camera, renderer;
let car, park;

//    ************************************
//    WORLD CREATION
//    ************************************

//    Add custom objects to the world
let addWorldObjects = () => {
      car = new Car();
      car.mesh.scale.set( 1, 1, 1 );
      car.mesh.position.set( 70, 0, 5 );
      car.mesh.rotation.y = toRad(-160);
      scene.add( car.mesh );

      createITESM(scene);
      park = new AmusementPark(scene);
}

//    Add light points to the scene
let addLights = () => {
      var front = new THREE.DirectionalLight( 0xffffff, .05 );
```

```
      front.position.set(-10, 1000, 1500);
      front.rotation.x += THREE.Math.degToRad(30);
      front.rotation.y += THREE.Math.degToRad(35);
      scene.add(front);
      front.castShadow = true;
      front.shadow.camera.near = 1.0;

      var right = new THREE.DirectionalLight( 0xffffff, .8 );
      right.position.set(-1030, 1000, 1500);
      right.rotation.y += THREE.Math.degToRad(90);
      scene.add(right);

      var left = new THREE.DirectionalLight( 0xffffff, .7 );
      left.position.set(1030, 1000, 1500);
      left.rotation.y += THREE.Math.degToRad(90);
      scene.add(left);

      var back = new THREE.DirectionalLight( 0xffffff,.75  );
      back.position.set(-10, 1000, -1500);
      back.rotation.x -= THREE.Math.degToRad(30);
      back.rotation.y -= THREE.Math.degToRad(35);
      scene.add(back);
}

//    Initialize the scene
let init = () => {
      scene = new THREE.Scene();

      camera = new THREE.PerspectiveCamera( 45, window.innerWidth /
window.innerHeight, 0.5, 500 );
      camera.position.x = 105;
      camera.position.y = 17;
      camera.position.z = 41;
      camera.rotation.y = 0.79;

      addWorldObjects();
      addLights();

      renderer = new THREE.WebGLRenderer();
      renderer.setSize( window.innerWidth, window.innerHeight );
      renderer.setClearColor(0x2a313d, 1); // set the background
color
      document.body.appendChild( renderer.domElement );
      window.addEventListener( 'keydown', keyDownController, true
);
```

```
     renderer.shadowMap.enabled = true;
};

//   Animate the scene
let animate = () => {
     requestAnimationFrame( animate );
     car.animateDrone();
     park.animate();
     renderer.render( scene, camera );
};

//   ************************************
//   CONTROLLER
//   ************************************

//   Controller for key events
let keyDownController = (evt) => {
     switch ( evt.keyCode ) {
          case 38:   // Up arrow
               goForward(1);
               break;
          case 40:   // Down arrow
               goForward(-1);
               break;
          case 37:   // Left arrow
               rotateCamera(5);
               break;
          case 39:   // Right arrow
               rotateCamera(-5);
               break;
          case 87:   // w
               elevateCamera(0.5);
               break;
          case 83:   // s
               elevateCamera(-0.5);
               break;
          case 65:   // a
               moveHorizontally(0.5);
               break;
          case 68:   // d
               moveHorizontally(-0.5);
               break;

          case 84:   // t
               car.goForward(0.5);
               break;
```

```
            case 71:   // g
                 car.goForward(-0.2);
                 break;
            case 72:   // h
                 car.steer(5);
                 break;
            case 70:   // f
                 car.steer(-5);
                 break;
            case 85:   // u
                 car.openDoors();
                 break;
            case 74:   // j
                 car.closeDoors();
                 break;
            case 89:   // y
                 car.openHood();
                 break;
            case 82:   // r
                 car.closeHood();
                 break;
            case 86:   // v
                 car.openTrunk();
                 break;
            case 66:   // b
                 car.closeTrunk();
                 break;
            case 88:   // x
                 car.toggleDrone();
                 break;
            default:
//               console.log(`Pressed key ${evt.keyCode}`);
                 break;
      }
};

// Rotate the camera on y axis
let rotateCamera = (degrees) => {
      camera.rotation.y += toRad(degrees);
};

// Make the camera go forward
let goForward = (distance) => {
      let alpha = camera.rotation.y;
      let sinA = Math.sin(alpha);
      let cosA = Math.cos(alpha);
```

```
      camera.position.x -= sinA * distance;
      camera.position.z -= cosA * distance;
};

//    Move the camera horizontally from its viewing direction
let moveHorizontally = (distance) => {
      let alpha = toRad(90) - camera.rotation.y;
      let sinA = Math.sin(alpha);
      let cosA = Math.cos(alpha);

      camera.position.x -= sinA * distance;
      camera.position.z += cosA * distance;
}

//    Vertical translation of the camera
let elevateCamera = (elevation) => {
      camera.position.y += elevation;
};

//    ************************************
//    BOOTSTRAP
//    ************************************

//    Bootstrap the element
init();
animate();
```

## DAE.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
function DAE() {
    this.object3D = this.makeDae();
}

/**
 * Constructs a Centro Estudiantil building object
 * @returns An Object3D containing all hierarchy
 */
DAE.prototype.makeDae = function(){

    //Mats
    var glassMat = new THREE.MeshPhongMaterial ({
```

```
        color: 0x01AA83,
        specular: 0x555555,
        transparent: true,
        opacity: 0.6,
           shininess: 20,
           side: THREE.DoubleSide
     });
     var concreteMat = new THREE.MeshLambertMaterial ({
           color: 0xbabcbc,
           side: THREE.DoubleSide
    });

    //Measures
    let daeX = 60;
    let daeY = 12;
    let daeZ = 50;

    let daeRoofX = daeX;
    let daeRoofY = 2/3*daeY;
    let daeRoofZ = daeZ;

    let daeCutX = daeX / 2;
    let daeCutY = daeY;
    let daeCutZ = 40;

    let blockX = daeX/4 * 2/3;
    let blockY = daeY * 1.23;
    let blockZ = 6.5;

    let terraceX = blockX * 1.12;
    let terraceY = 2;
    let terraceZ = 19;

    //Geometries
    let daeGeom = new THREE.BoxGeometry(daeX, daeY, daeZ);
    let daeRoofGeom = new THREE.BoxGeometry(daeRoofX, daeRoofY,
daeRoofZ);
    let daeCutGeom = new THREE.BoxGeometry(daeCutX, daeCutY,
daeCutZ);
    let glassGeom = new THREE.BoxGeometry(daeCutX, daeCutY, 1);
    let blockGeom = new THREE.BoxGeometry(blockX, blockY, blockZ);
    let block2Geom = new THREE.BoxGeometry(blockX, terraceZ,
blockZ);
    let terraceGeom = new THREE.BoxGeometry(terraceX, terraceY,
terraceZ);
```

```
    //Meshes
    let dae = new THREE.Mesh(daeGeom);
    let daeRoof = new THREE.Mesh(daeRoofGeom);
    let daeCut = new THREE.Mesh(daeCutGeom);
    let block = new THREE.Mesh(blockGeom, concreteMat);
    let block2 = new THREE.Mesh(block2Geom, concreteMat);
    let terrace = new THREE.Mesh(terraceGeom, concreteMat);
    let glass = new THREE.Mesh(glassGeom, glassMat);

    //Offsets
    dae.position.y = daeY / 2;
    daeRoof.position.y = daeY + daeRoofY / 2;
    daeCut.position.y = dae.position.y;
    daeCut.position.z = daeZ / 2;
    glass.position.y = daeCutY / 2;
    glass.position.z = daeZ / 2 - 2.5;

    block.position.x = -daeX / 2 - blockX/2 + 1;
    block.position.y = dae.position.y * .365;
    block.position.z = daeZ / 2 - 3*blockZ/2;

    block2.position.copy(block.position);
    block2.rotation.x += Math.PI / 2;
    block2.rotation.y += Math.PI / 2;
    block2.position.z = terrace.position.z;
    block2.position.x += blockX / 3;
    block2.position.y += 3.5;

    terrace.position.x = -daeX / 2 - terraceX /2;

    //CSG
    var daeCSG = THREE.CSG.fromMesh(dae);
    daeCSG = daeCSG.union(THREE.CSG.fromMesh (daeRoof));
    daeCSG = daeCSG.subtract(THREE.CSG.fromMesh (daeCut));

    let daeObj = THREE.CSG.toMesh(daeCSG, concreteMat);

    var daeGroup = new THREE.Object3D();
    daeGroup.add(daeObj, glass, terrace, block, block2);

    return daeGroup;
}
```

## drone.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
const loader2 = new THREE.GLTFLoader();
const mtlLoader2 = new THREE.MTLLoader();

//    Import propeller and scale
function Propeller() {
      this.mesh = new THREE.Object3D();
      this.group = new THREE.Object3D();

      this.init = () => {
            mtlLoader2.setTexturePath('./objects/');
            mtlLoader2.setPath('./objects/');
            mtlLoader2.load('propeller.mtl', (materials) => {
                  materials.preload();
                  let objLoader = new THREE.OBJLoader();
                  objLoader.setMaterials(materials);
                  objLoader.setPath('./objects/');
                  objLoader.load('propeller.obj', (object) => {
                        this.group.add( object );
                  });
            });

            this.group.rotation.z = toRad(-90);
            this.group.scale.set( 0.1, 0.1, 0.1);
            this.mesh.add( this.group );
      };

      this.init();
}

//    Import drone body and attach propellers
function Drone() {
      this.mesh = new THREE.Object3D();
      this.group = new THREE.Object3D();
      this.activated; // Control flag
      this.onBottom = true;// Control flag

      this.propL;
      this.propR;

      this.init = () => {
```

```
	loader2.load( './drone.glb', ( gltf ) => {
		this.group.add( gltf.scene );
	}, undefined, function ( error ) {
		console.error( error );
	} );

	this.group.rotation.y = toRad(90);
	this.group.scale.set( 0.8, 0.8, 0.8 );
	this.mesh.add( this.group );
	this.mesh.position.y = 1.15;

	this.propL = new Propeller();
	this.mesh.add( this.propL.mesh );
	this.propL.mesh.position.z -= 0.2;

	this.propR = new Propeller();
	this.mesh.add( this.propR.mesh );
	this.propR.mesh.position.z += 0.2;
};

//	Animate propellers if required,
//	update height if required
this.animate = () => {
	if(this.activated && !this.onTop) {
		this.mesh.position.y += 0.02;

		if(this.mesh.position.y >= 3)
		this.onTop = true;
	}

	if(!this.activated && this.onTop) {
		this.mesh.position.y -= 0.02;

		if(this.mesh.position.y <= 1.15)
			this.onTop = false;
	}

	if(this.mesh.position.y <= 1.15)
		this.onBottom = true;
	else
		this.onBottom = false;

	if(!this.onBottom) {
		this.propL.mesh.rotation.y += toRad(15);
		this.propR.mesh.rotation.y += toRad(15);
	}
```

```
	};

	//	Activate the drone
	this.activate = () => {
		this.activated = true;
	};

	//	Re-attach the drone
	this.settle = () => {
		this.activated = false;
	};

	this.init();
}
```

## DropTower.js

```
/*A01328937	Luis Francisco Flores Romero*/
/*A01324276	Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
function DropTower() {

	//For seats' movement
	this.yDelta;
	this.startPosition;

	this.object3D = this.makeDropTower();

	this.animate = function() {
		this.object3D.children[2].position.y =
this.startPosition + this.yDelta * Math.sin(Date.now() / 1000);
		//console.log("yDelta: " + this.yDelta + ", startPos: "
+ this.startPosition);
	}
}

/**
 *	object3D
 *		tower
 *		dome
 *		seats
 *			body
 *			seatsG
 */
```

```
DropTower.prototype.makeDropTower = function() {

      //Measures
      let towerHeight = 75;
      let towerRadius = 6;
      let domeX = 16;
      let domeY = 6;
      let domeZ = domeX;
      let seatsRadius = towerRadius * 1.45;
      let seatsWidth = towerRadius / 2;

      let domeCRadius = (domeX * Math.sqrt(2)) / 2;
      let domeCHeight = domeY;

      //Materials
      let yellow = new THREE.MeshLambertMaterial({color:
0xfff30f});
      let purple = new THREE.MeshLambertMaterial({color:
0xc904e2});

      let towerGeom = new THREE.CylinderGeometry(towerRadius,
towerRadius, towerHeight, 32);
      let tower = new THREE.Mesh(towerGeom, yellow);

      let domeGeom = new THREE.BoxGeometry(domeX, domeY, domeZ);
      let dome = new THREE.Mesh(domeGeom, purple);

      let domeCGeom = new THREE.CylinderGeometry(domeCRadius,
domeCRadius, domeCHeight, 32, 4, true);
      let domeC = new THREE.Mesh(domeCGeom, purple.clone());
      domeC.material.wireframe = true;

      let seats = makeSeats(seatsRadius, seatsWidth, yellow, 32);

      //Groups
      var dropTower = new THREE.Object3D();
      dropTower.add(tower,dome, seats, domeC);

      tower.position.y = towerHeight / 2.0;
      dome.position.y = towerHeight - domeY / 2.0;
      seats.position.y = tower.position.y;
      domeC.position.y = dome.position.y;

      this.startPosition = seats.position.y;
      this.yDelta = (towerHeight - domeY - seatsWidth / 2.0) -
this.startPosition;
```

```
	return dropTower;
}

function makeSeats(seatsBodyRadius, seatsBodyWidth, yellowMat,
seatCount) {

	let bodyGeom = new THREE.TorusGeometry(seatsBodyRadius,
seatsBodyWidth, 32, 32);
	let body = new THREE.Mesh(bodyGeom, yellowMat);

	let seats = new THREE.Object3D();
	let seatsG = new THREE.Object3D();
	seats.add(body, seatsG);

	for (var i = 0; i < seatCount; i++) {
		let seat = makeSeat();
		var angle = i * ((2 * Math.PI) / seatCount);

		seatsG.add(seat);
		seat.position.set((seatsBodyRadius + seatsBodyWidth *
1.25) * Math.cos(angle), 0, (seatsBodyRadius + seatsBodyWidth *
1.25) * Math.sin(angle));
		seat.rotation.y = -angle;
	}

	body.rotation.x = Math.PI /2;

	return seats;
}

function makeSeat() {

	let mat = new THREE.MeshLambertMaterial({color: 0x111111});

	let initialScale = 2;
	let seatWidth = .9 * initialScale;
	let seatHeight = .25 * initialScale;
	let seatGirth = .7 * initialScale;
	let backrestHeight = seatWidth * 1.5;

	let seatGeom = new THREE.BoxGeometry(seatWidth, seatHeight,
seatGirth);
	let seat = new THREE.Mesh(seatGeom, mat);

	let backRestGeom = new THREE.BoxGeometry(backrestHeight,
seatHeight, seatGirth);
```

```
      let backRest = new THREE.Mesh(backRestGeom, mat);

      let seatObj = new THREE.Object3D();
      seatObj.add(seat, backRest);

      backRest.rotation.z -= THREE.Math.degToRad(75);
      backRest.position.x -= backrestHeight / 3;
      seat.position.y -= seatHeight * 2;

      return seatObj;
}
```

## FastFruitArea.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
function makeFastFruitArea() {
    //Materials
    let concreteMat = new THREE.MeshLambertMaterial ({
          color: 0xbabcbc,
        side: THREE.DoubleSide,
        polygonOffset: true,
        polygonOffsetFactor: -0.08
    });
    let fakeGrassMat = new THREE.MeshLambertMaterial ({
          color: 0x009336,
        side: THREE.DoubleSide,
        polygonOffset: true,
        polygonOffsetFactor: -0.09
    });
    let glassMat = new THREE.MeshPhongMaterial ({
        color: 0x003f3f,
        specular: 0x222222,
        transparent: true,
        opacity: 0.4,
          shininess: 50,
        side: THREE.DoubleSide,
        polygonOffset: true,
        polygonOffsetFactor: -0.1
     });

    //Measures
    let sideLength = 32;
    let bodyHeight = 3.55;
```

```
    let centerRadius = sideLength / 3;

    //Geometries
    let mainBodyGeom = new THREE.BoxGeometry(sideLength,
bodyHeight, sideLength);
    let centerGeom = new THREE.SphereGeometry(centerRadius, 64,
64);
    let exitGeom = new THREE.BoxGeometry(sideLength - 1.2,
bodyHeight - .6, sideLength + 1);
    let glassGeom = new THREE.CylinderGeometry(centerRadius,
centerRadius, bodyHeight - .3, 32, 1, true);
    let grassGeom = new THREE.BoxGeometry(sideLength, sideLength,
0.05);

    //Meshes
    let mainBody = new THREE.Mesh(mainBodyGeom);
    let center = new THREE.Mesh(centerGeom);
    let exit = new THREE.Mesh(exitGeom);
    let glass = new THREE.Mesh(glassGeom, glassMat);
    let grass = new THREE.Mesh(grassGeom);
    grass.rotation.x += Math.PI/2;

    let southCorridor = makeAdjacentToFF();
    southCorridor.position.z -= sideLength / 2 + 4.70;
    southCorridor.position.y -= 8.15;
    southCorridor.position.x += 4;

    let northCorridor = makeAdjacentToFF(true);
    northCorridor.scale.x *= -1;
    northCorridor.scale.z *= -1;
    northCorridor.position.z = sideLength / 2 + 4.75;
    northCorridor.position.y = -8.15 + 3.25;
    northCorridor.position.x = -4

    //CSG
    var mainBodyCSG = THREE.CSG.fromMesh(mainBody);
    mainBodyCSG =
mainBodyCSG.subtract(THREE.CSG.fromMesh(center));
    mainBodyCSG = mainBodyCSG.subtract(THREE.CSG.fromMesh(exit));

    exit.rotation.y = Math.PI / 2;
    mainBodyCSG = mainBodyCSG.subtract(THREE.CSG.fromMesh(exit));

    var topCSG = THREE.CSG.fromMesh(grass);
    topCSG = topCSG.subtract(THREE.CSG.fromMesh(new
THREE.Mesh(centerGeom)));
```

```
    //Hierarchy
    var FFarea = new THREE.Object3D();
    var grassTop = THREE.CSG.toMesh(topCSG, fakeGrassMat);
    grassTop.position.y += bodyHeight / 2;
    FFarea.add(THREE.CSG.toMesh(mainBodyCSG, concreteMat), glass,
grassTop, southCorridor, northCorridor);

    return FFarea;
}

/**
 * Constructs a hallway adjacent to fast fruit area
 * @param {boolean} isForNorthSide Whether the hallway is north
(towards fountain) or south (towards library). Used mainly to add
glass to one of the hallways if in north side
 * @returns An Object3D containing all hierarchy
 */
function makeAdjacentToFF(isForNorthSide = false) {
    var object = new THREE.Object3D();

    //only 1 aula
    let level1 = new Story(1, false, false, 1, false);
    let level2 = new Story(2, false, false, 1, false);
    level2.object3D.position.y = level2.levelHeight;

    //3 aulas
    let level3 = new Story(3, false, false, 3, false, false);
    level3.object3D.position.y = level3.levelHeight * 2;
    level3.object3D.position.x -= 8;

    //none
    let level4 = new Story(4, false, true, 3, false,
isForNorthSide);
    level4.object3D.position.y = level4.levelHeight * 3;
    level4.object3D.position.x -= 8;

    let stairs = makeAulaStairs(3);
    stairs.position.x += 8; //aula and stairs

    object.add(level1.object3D, level2.object3D, level3.object3D,
level4.object3D, stairs);

    return object;
}
```

## ferris.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/

//    Create capsule
//    baseRadius Radius of the capsule
//    jointRadius     Radius of the capsule joint ring
//    material   Material to apply to the capsule
function Capsule(baseRadius, jointRadius, material) {
      this.mesh = new THREE.Object3D();

      this.jointRadius = jointRadius;
      this.baseRadius = baseRadius;
      this.material = material ? material : whiteMetallicSheet;

      //    Initialization
      let geom = new THREE.TorusGeometry(
            jointRadius * (5 / 4),
            jointRadius / 4,
            8,
            24
      );
      let joint = new THREE.Mesh( geom, aluminium );

      geom = new THREE.TorusGeometry(
            baseRadius * (15 / 16),
            baseRadius / 16,
            8,
            24,
            Math.PI
      );
      let handle = new THREE.Mesh( geom, aluminium );
      handle.rotation.y = Math.PI / 2;
      handle.position.y -= baseRadius;
      handle.position.y -= jointRadius * (5 / 4);

      geom = new THREE.SphereGeometry(
            baseRadius,
            16,
            16,
            0,
            2 * Math.PI,
```

```
            Math.PI / 2,
            Math.PI / 2
      );
      let base = new THREE.Mesh( geom, this.material );
      base.position.y -= baseRadius;
      base.position.y -= jointRadius * (5 / 4);

      this.mesh.add( joint );
      this.mesh.add( handle );
      this.mesh.add( base );
}

//    Create the ferris wheel
//    radius          Wheel radius
//    capsules   Amount of capsules in the wheel
//    tubeRadius Tube radius of the wheel structure
//    ringToRing Distance in between wheel rings
function FerrisWheelSlim(radius, capsules, tubeRadius, ringToRing)
{
      this.mesh = new THREE.Object3D();
      this.wheel;
      this.capsulesArr;
      this.ridableGroup;
      this.base;
      this.baseHeight;

      this.radius = radius;
      this.capsules = capsules;
      this.tubeRadius = tubeRadius;
      this.ringToRing = ringToRing;
      this.supportLength = ringToRing * 0.2;

      this.createFixingTube = (angle) => {
            geom = new THREE.CylinderGeometry(
                  tubeRadius,
                  tubeRadius,
                  ringToRing,
                  16,
                  8
            );
            let fixingTube = new THREE.Mesh( geom,
whiteMetallicSheet );
            fixingTube.rotation.x = Math.PI / 2;
            fixingTube.position.x = radius * Math.cos(toRad(angle));
            fixingTube.position.y = radius * Math.sin(toRad(angle));
```

```
            return fixingTube;
      };

      this.createRadialTube = (angle, material) => {
            let joinableGroup = new THREE.Object3D();
            geom = new THREE.CylinderGeometry(
                  tubeRadius / 2,
                  tubeRadius / 3,
                  radius,
                  16,
                  8
            );
            let radialTube = new THREE.Mesh( geom, material );
            radialTube.rotation.z = Math.PI / 2;
            radialTube.position.x = radius / 2;
            joinableGroup.add( radialTube );
            joinableGroup.rotation.z = toRad( angle );

            return joinableGroup;
      };

      this.createWheel = () => {
            let ringGroup = new THREE.Object3D();
            let wheelGroup = new THREE.Object3D();

            let geom = new THREE.TorusGeometry(
                  radius,
                  tubeRadius,
                  8,
                  capsules * 3
            );
            let ring = new THREE.Mesh( geom, whiteMetallicSheet );

            ringGroup.add( ring );
            for(let i = 0; i < 2 * capsules; i++) {
                  let angle = (i * 360) / (2 * capsules);
                  ringGroup.add( this.createRadialTube(angle,
psycho) );
            }

            let rearRingGroup = ringGroup.clone();
            ringGroup.position.z = ringToRing / 2;
            rearRingGroup.position.z = - ringToRing / 2;

            geom = new THREE.CylinderGeometry(
                  tubeRadius,
```

```
                tubeRadius,
                ringToRing + 2 * this.supportLength,
                16,
                8
            );
            let centralAxis = new THREE.Mesh( geom, aviationAl );
            centralAxis.rotation.x = Math.PI / 2;

            wheelGroup.add( ringGroup );
            wheelGroup.add( rearRingGroup );
            wheelGroup.add( centralAxis );
            for(let i = 0; i < capsules; i++) {
                wheelGroup.add( this.createFixingTube(i * 360 /
capsules) );
            }

            return wheelGroup;
        };

        this.createBase = () => {
            let baseGroup = new THREE.Object3D();
            let supportGroup = new THREE.Object3D();
            let tubeLength = this.radius * 1.5;
            this.baseHeight = tubeLength * Math.sin(toRad(60));

            let geom = new THREE.CylinderGeometry(
                this.tubeRadius * 1.2,
                this.tubeRadius * 1.2,
                tubeLength,
                16,
                8
            );
            let tube = new THREE.Mesh( geom, whiteMetallicSheet );
            tube.position.y = -tubeLength / 2;
            supportGroup.add( tube );
            let supportGroup2 = supportGroup.clone();
            let supportGroup3 = supportGroup.clone();
            let supportGroup4 = supportGroup.clone();
            supportGroup.rotation.z = toRad(30);
            supportGroup.rotation.y = toRad(-30);
            supportGroup.position.z += this.ringToRing / 2;
            supportGroup.position.z += this.tubeRadius * 1.5;
            supportGroup.position.y = this.baseHeight;

            supportGroup2.rotation.z = toRad(-30);
            supportGroup2.rotation.y = toRad(30);
```

```
            supportGroup2.position.z += this.ringToRing / 2;
            supportGroup2.position.z += this.tubeRadius * 1.5;
            supportGroup2.position.y = this.baseHeight;

            supportGroup3.rotation.z = toRad(30);
            supportGroup3.rotation.y = toRad(30);
            supportGroup3.position.z -= this.ringToRing / 2;
            supportGroup3.position.z -= this.tubeRadius * 1.5;
            supportGroup3.position.y = this.baseHeight;

            supportGroup4.rotation.z = toRad(-30);
            supportGroup4.rotation.y = toRad(-30);
            supportGroup4.position.z -= this.ringToRing / 2;
            supportGroup4.position.z -= this.tubeRadius * 1.5;
            supportGroup4.position.y = this.baseHeight;

            baseGroup.add( supportGroup );
            baseGroup.add( supportGroup2 );
            baseGroup.add( supportGroup3 );
            baseGroup.add( supportGroup4 );
            return baseGroup;
      };

      this.rotate = (angle) => {
            this.ridableGroup.rotation.z += toRad(angle);
            this.capsulesArr.forEach((element) => {
                  element.rotation.z -= toRad(angle);
            });
      };

      this.wheel = this.createWheel();
      this.ridableGroup = new THREE.Object3D();
      this.ridableGroup.add( this.wheel );

      this.capsulesArr = [];
      for(let i = 0; i < capsules; i++) {
            this.capsulesArr.push(new Capsule(ringToRing * 0.4,
tubeRadius,
                  redSteel).mesh);
            this.ridableGroup.add( this.capsulesArr[i] );

            let angle = i * 360 / capsules;
            this.capsulesArr[i].position.x = radius *
Math.cos(toRad(angle));
            this.capsulesArr[i].position.y = radius *
Math.sin(toRad(angle));
```

```
      }

      let base = this.createBase();
      this.ridableGroup.position.y = this.baseHeight;

      this.mesh.add( this.ridableGroup );
      this.mesh.add( base );
}
```

## FerrisWheel.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
function FerrisWheel() {
    this.object3D = makeFerrisWheel();
    this.object3D.scale.set(2.5, 2.5, 2.5);

      this.animate = function() {
           this.object3D.children[0].rotation.z += 0.001;
//rotationG

           var pods =
this.object3D.children[0].children[2].children; //pods
           for (var i = 0; i < pods.length; i++) {
                pods[i].rotation.z -= 0.001;
           }
      }
}
/**
 * FerrisWheel
 *         rotationG
 *              body
 *         center
 *              podsG
 *         anchor
 */
function makeFerrisWheel() {

    //Materials
    let whiteWireMat = new THREE.MeshLambertMaterial({color:
0x868781, wireframe: true});
    let blackMat = new THREE.MeshLambertMaterial({color: 0x222222,
side: THREE.DoubleSide});
```

```
    //Measures and constants
    let wheelRadius = 10;
    let wheelHeight = 3;
    let centerRadius = 1;
    let anchorRadius = 3;
    let anchorHeightScaleFactor = 2;
    let numPods = 16;

     //groups
     var ferrisWheel = new THREE.Object3D();
     var podG = new THREE.Object3D();
     var rotationG = new THREE.Object3D();

     //objects
    let bodyGeom = new THREE.CylinderGeometry(wheelRadius,
wheelRadius, wheelHeight, 32);
    let body = new THREE.Mesh(bodyGeom, whiteWireMat);

    let centerGeom = new THREE.CylinderGeometry(centerRadius,
centerRadius, wheelHeight, 32);
    let center = new THREE.Mesh(centerGeom, blackMat);

    let anchorGeom = new THREE.ConeGeometry(anchorRadius,
wheelRadius * anchorHeightScaleFactor, 3, 1, false,  -Math.PI / 2,
Math.PI);
    let anchor = new THREE.Mesh(anchorGeom, blackMat);

     //Add objs to hierarchy
     rotationG.add(body, center, podG)
    ferrisWheel.add(rotationG, anchor);

    //Create pods
    for (var i = 0; i < numPods; i++) {
        var pod = createPod();
        var angle = i * (Math.PI * 2) / numPods;
        podG.add(pod);

        pod.position.set(wheelRadius * Math.cos(angle),
wheelRadius * Math.sin(angle), 0);
    }

    anchor.position.y = wheelRadius * anchorHeightScaleFactor /
2.0;
    anchor.position.z += anchorRadius / 2.0 + .3;

     body.rotation.x += Math.PI / 2.0;
```

```
     center.rotation.x += Math.PI / 2.0;
    rotationG.position.y += anchor.position.y * 1.2;

    return ferrisWheel;
}

/**
 *    Pod
 *         floor
 *         podBody
 *         umbrella
 *
 */
function createPod() {

    let radius = 1.25;
    let height = radius * .75;

    let material = new THREE.MeshLambertMaterial({color:
0xefeb0e});
    let blackMat = new THREE.MeshLambertMaterial({color: 0x222222,
side: THREE.DoubleSide});

    var pod = new THREE.Object3D();

    let umbrella = createUmbrellaObject(radius, height / 2,
material);

    let podGeom = new THREE.CylinderGeometry(radius, radius,
height, 6, 1, true);
    let podBody = new THREE.Mesh(podGeom, blackMat);

    let floorGeom = new THREE.CylinderGeometry(radius, radius,
0.05, 6, 1, false);
    let floor = new THREE.Mesh(floorGeom, blackMat);

     pod.add(floor, podBody, umbrella);
     floor.position.y -= height / 2.0;

    return pod;
}
```

## ITESM.js

```
/*A01328937     Luis Francisco Flores Romero*/
```

```
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/

/**
 * Creates ITESM Campus Puebla in the given scene
 * @param {*} scene Scene from THREE JS that the ITESM will be
added to
 */
function createITESM(scene) {
    var aulas2 = new AulasBuilding(2).object3D;
    scene.add(aulas2);
    aulas2.castShadow = true;
    aulas2.receiveShadow = true;

    var aulas1 = new AulasBuilding(1).object3D;
    aulas1.scale.z *= -1;
    aulas1.scale.x *= -1;
    aulas1.position.x = -8.9;
    aulas1.position.z = 4.3;
    aulas1.position.y = 3.3;
    scene.add(aulas1);
    aulas1.castShadow = true;
    aulas1.receiveShadow = true;

    var cafeteria = new Cafeteria().object3D;
    cafeteria.position.y = aulas1.position.y;
    cafeteria.position.x = 100;
    cafeteria.scale.set(.5,.5,.5);
    cafeteria.rotation.y -= Math.PI;
    cafeteria.position.z = 25;
    scene.add(cafeteria);

    var dae = new DAE().object3D;
    dae.position.x = -125;
    dae.position.z = -85;
    dae.scale.set(.65,.65,.65);

    scene.add(dae);
}
```

## Magna.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
```

```
function Magna() {
    this.object3D = this.makeMagna();
}

/**
 * Constructs an Aula Magna
 * @returns An Object3D containing all hierarchy
 */
Magna.prototype.makeMagna = function() {

    //Materials
    var glassMat = new THREE.MeshPhongMaterial ({
        color: 0x003f3f,
        specular: 0x222222,
        transparent: true,
        opacity: 0.8,
           shininess: 50,
           side: THREE.DoubleSide
     });
     var concreteMat = new THREE.MeshLambertMaterial ({
          color: 0xbabcbc,
        side: THREE.DoubleSide,
    });

    //Measures
    let magnaRadius = 7;
    let magnaHeight = 3.15;
    let firstBaseHeight = magnaHeight * .25;
    let windowHeight = magnaHeight * .12;
    let inBetweenBaseHeight = magnaHeight * .2;
    let topBaseHeight = magnaHeight - (firstBaseHeight +
windowHeight * 2 + inBetweenBaseHeight);
    let groundHeight = .30;
    let windowRadius = magnaRadius * 1;
    let thethaLength = Math.PI * 2 - THREE.Math.degToRad(17);
    let thetaStart = THREE.Math.degToRad(-50);

    //Geometries
    let firstBaseGeom = new THREE.CylinderGeometry(magnaRadius,
magnaRadius, firstBaseHeight, 128, 4, true, thetaStart,
thethaLength);
    let windowGeom = new THREE.CylinderGeometry(windowRadius,
windowRadius, windowHeight, 128, 4, true, thetaStart,
thethaLength);
```

```
    let inBetweenBaseGeom = new
THREE.CylinderGeometry(magnaRadius, magnaRadius,
inBetweenBaseHeight, 128, 4, true, thetaStart, thethaLength);
    let topBaseGeom = new THREE.CylinderGeometry(magnaRadius,
magnaRadius, topBaseHeight, 128, 4, true);

    let groundGeom = new THREE.CylinderGeometry(magnaRadius,
magnaRadius, groundHeight, 128, 4, false);

    //Meshes
    let firstBase = new THREE.Mesh(firstBaseGeom, concreteMat);
    let window1 = new THREE.Mesh(windowGeom, glassMat);
    let inBetweenBase = new THREE.Mesh(inBetweenBaseGeom,
concreteMat);
    let window2 = window1.clone();
    let topBase = new THREE.Mesh(topBaseGeom, concreteMat);

    let ground = new THREE.Mesh(groundGeom, concreteMat);
    let ceiling = ground.clone();

    //offsets
    firstBase.position.y = firstBaseHeight / 2;
    window1.position.y = firstBaseHeight + windowHeight / 2;
    inBetweenBase.position.y = firstBaseHeight + windowHeight +
inBetweenBaseHeight / 2;
    window2.position.y = firstBaseHeight + windowHeight +
inBetweenBaseHeight + windowHeight /2;
    topBase.position.y = firstBaseHeight + windowHeight +
inBetweenBaseHeight + windowHeight + topBaseHeight / 2;

    ground.position.y = -groundHeight/2;
    ceiling.position.y = magnaHeight + groundHeight / 2;

    //
    var magna = new THREE.Object3D();
    magna.add(ground, ceiling, firstBase, window1, window2,
inBetweenBase, topBase);

    return magna;

}
```

## materials.js

```
/*A01328937     Luis Francisco Flores Romero*/
```

```
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*16.nov.2018*/

//    ************************************
//    BASIC MATERIALS
//    ************************************
let wire = new THREE.MeshBasicMaterial( {
     color: 0xffff00,
     wireframe: true
} );

let basic = new THREE.MeshBasicMaterial( {
     color: 0xaa0000
} );

let lambert = new THREE.MeshLambertMaterial( {
     color:  0x00aacc
} );

//    ************************************
//    CUSTOM MATERIALS
//    ************************************
let aviationAl = new THREE.MeshLambertMaterial( {
     color: 0x404040
} );
let aluminium = new THREE.MeshLambertMaterial( {
     color: 0x828282
} );
let steel = new THREE.MeshPhongMaterial( {
     color: 0xa5a5a5
} );
let whiteMetallicSheet = new THREE.MeshLambertMaterial( {
     color: 0xeeeeee
} );
let redSteel = new THREE.MeshLambertMaterial( {
     color: 0xe51e14,
     side: THREE.DoubleSide
} );
let wood = new THREE.MeshLambertMaterial( {
     color: 0x513902
} );
let psycho = new THREE.MeshNormalMaterial();
let glass = new THREE.MeshPhongMaterial( {
     color: 0x3fd2ff,
     transparent: true,
     opacity: 0.8
```

```
} );
```

## Stairs.js

```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/
function Stairs(width, rise, makeBeginPlatform = false) {
    this.stepHeight = 0.18;
    this.stepCount = rise / this.stepHeight;
    this.stepRun =  0.45;
    this.run = this.stepCount * this.stepRun;
    this.object3D = this.makeStairs(width, rise,
makeBeginPlatform);
}

/**
 * Constructs a set of standard stairs with an ending floor.
 * @param {Number} width The X dimensions of the stairs
 * @param {Number} rise The height (Y dimension) the stairs will
have
 * @param {boolean} makeBeginPlatform Whether or not to add a
platform to the beginning of the stair
 * @returns An Object3D containing all hierarchy
 */
Stairs.prototype.makeStairs = function(width, rise,
makeBeginPlatform) {

    let stepMat = new THREE.MeshLambertMaterial ({
          color: 0x77797c,
        side: THREE.DoubleSide,
    });

    let endPlatformDepth = width;

    var nextZ = -this.stepRun/2;
    var nextY = this.stepHeight / 2;
    let stepGeom = new THREE.BoxGeometry(width, this.stepHeight,
this.stepRun);
    var stairs = new THREE.Object3D();
    for (var i = 0; i < this.stepCount - 1; i++) {

        let step = new THREE.Mesh(stepGeom, stepMat);

        stairs.add(step);
```

```
        step.position.set(0, nextY, nextZ);

        nextZ -= this.stepRun;
        nextY += this.stepHeight;
    }
    let endPlatformGeom = new THREE.BoxGeometry(width,
this.stepHeight, endPlatformDepth);
    let endPlatform = new THREE.Mesh(endPlatformGeom, stepMat);
    endPlatform.position.set(0, nextY, nextZ + this.stepRun/2 -
endPlatformDepth / 2);
    stairs.add(endPlatform);

    if (makeBeginPlatform) {
        let beginPlatform = endPlatform.clone();
        beginPlatform.position.set(0, this.stepHeight / 2,
endPlatformDepth / 2);

        stairs.add(beginPlatform);
    }

    return stairs;
}

/**
 * Constructs a U-Shaped stair
 * @param {Number} width The X dimensions of the stairs together
 * @param {Number} rise The height (Y dimension) the 2 stairs will
have
 * @param {boolean} makeBeginPlatform Whether or not to add a
platform to the beginning of the stair
 * @returns An Object3D containing all hierarchy
 */
function makeUShapedStair(width, rise, makeBeginPlatform = false)
{

    var stair = new Stairs(width / 2, rise / 2,
makeBeginPlatform);
    let stepHeight = stair.stepHeight;

    var firstHalf = stair.object3D;
    var secondHalf = firstHalf.clone();

    firstHalf.position.x = width / 4;
    firstHalf.position.y -= stepHeight/2;
    secondHalf.rotation.z += Math.PI;
    secondHalf.position.y = rise;
```

```
    secondHalf.position.x = -width/4;

    var stairSet = new THREE.Object3D();
    stairSet.add(firstHalf, secondHalf);

    return stairSet;
}

/**
 * Constructs a U-Shaped stair
 * @param {Number} levels The amount of stairs to stack on top of
each other
 * @param {boolean} makeBeginPlatform Whether or not to add a
platform to the beginning of the stair
 * @returns An Object3D containing all hierarchy
 */
function makeAulaStairs (levels, makeBeginPlatform = false) {

    //Materials
    let coverMat = new THREE.MeshPhongMaterial({
        color: 0xb7b7b7,
        shininess: 0.6,
        reflectivity: 0.8,
        side: THREE.DoubleSide
    });
    let stepMat = new THREE.MeshLambertMaterial ({
          color: 0x77797c,
        side: THREE.DoubleSide,
    });

    //Measures
    let levelHeight = (new THREE.Box3().setFromObject(new
Aula().object3D)).getSize().y;
    let coverRadius = 4;
    let thetaLength = Math.PI;
    let thetaStart = Math.PI / 2;
    let coverHeight = levelHeight * (levels + 1);
    let latWallX = 0.1;
    let latWallY = coverHeight;
    let latWallZ = coverRadius * 1.5;
    let floorZ = 5;

    //Geometries
    let coverGeom = new THREE.CylinderGeometry(coverRadius,
coverRadius, coverHeight, 64, levels, false, thetaStart,
thetaLength);
```

```
    let latWallGeom = new THREE.BoxGeometry(latWallX, latWallY,
latWallZ);
    let floorGeom = new THREE.BoxGeometry(coverRadius * 2, .3,
floorZ);

    //Object
    let aulaStairs = new THREE.Object3D();

    //Meshes
    let cover = new THREE.Mesh(coverGeom, coverMat);
    let wall1 = new THREE.Mesh(latWallGeom, coverMat);
    let wall2 = wall1.clone();

    aulaStairs.add(cover, wall1, wall2);
    //Stairs
    for (var i = 0; i <= levels; i++) {

        if (i < levels) {
            var stairs = makeUShapedStair(coverRadius * 2,
levelHeight, makeBeginPlatform);
            stairs.position.y = i * levelHeight;

            aulaStairs.add(stairs);
        }

        let floor = new THREE.Mesh(floorGeom, stepMat);
        floor.position.z = floorZ / 2;
        floor.position.y = i * levelHeight - .15;
        aulaStairs.add(floor);
    }

    cover.position.y = coverHeight / 2;
    cover.position.z = - 3/2*coverRadius;

    wall1.position.y = wall2.position.y = cover.position.y;
    wall1.position.z = wall2.position.z = - latWallZ / 2;
    wall1.position.x = -coverRadius;
    wall2.position.x = coverRadius;

    return aulaStairs;
}
```

## Story.js

```
/*A01328937     Luis Francisco Flores Romero*/
```

```
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*25.nov.2018*/

/**
 * Constructor for a single building's story
 * @param {Number} level The number from 1 to n, that this story
is
 * @param {boolean} shouldHaveGlass Whether to add glass to the
story (ignored unless level > 1)
 * @param {boolean} isTopLevel Whether the level is the top-most
in the building
 * @param {Number} aulaCount The number of classrooms to have for
this story
 * @param {boolean} shouldHaveMagna Whether the story should have
a magna classroom
 * @param {boolean} shouldHaveFrontBarrier Whether the story
should have a banister (ignored unless level > 1)
 * @param {boolean} shouldExtendFrontBarrier If the story has a
banister, whether to extend it a fraction of an aula's length
 */
function Story(level, shouldHaveGlass, isTopLevel, aulaCount,
shouldHaveMagna, shouldHaveFrontBarrier = true,
shouldExtendFrontBarrier = false) {
    this.glassMat = new THREE.MeshPhongMaterial ({
        color: 0x003f3f,
        specular: 0x222222,
        transparent: true,
        opacity: 0.8,
           shininess: 50,
        side: THREE.DoubleSide,
        polygonOffset: true,
        polygonOffsetFactor: -0.12
     });
     this.concreteMat = new THREE.MeshLambertMaterial ({
           color: 0xbabcbc,
        side: THREE.DoubleSide,
        polygonOffset: true,
        polygonOffsetFactor: -0.1
    });
    this.waterProofMat = new THREE.MeshLambertMaterial ({
           color: 0xc43e2d,
        side: THREE.DoubleSide,
        polygonOffset: true,
        polygonOffsetFactor: 0.08
    });
    this.rockMat = new THREE.MeshLambertMaterial ({
```

```
            color: 0x77797c,
        side: THREE.DoubleSide,
        polygonOffset: true,
        polygonOffsetFactor: 0.11
    });

    this.object3D = this.makeStory(level, shouldHaveGlass,
isTopLevel, aulaCount, shouldHaveMagna, shouldHaveFrontBarrier,
shouldExtendFrontBarrier);
}

/**
 * Constructs the Story hierarchy
 * @param {Number} level The number from 1 to n, that this story
is
 * @param {boolean} shouldHaveGlass Whether to add glass to the
story (ignored unless level > 1)
 * @param {boolean} isTopLevel Whether the level is the top-most
in the building
 * @param {Number} aulaCount The number of classrooms to have for
this story
 * @param {boolean} shouldHaveMagna Whether the story should have
a magna classroom
 * @param {boolean} shouldHaveFrontBarrier Whether the story
should have a banister (ignored unless level > 1)
 * @param {boolean} shouldExtendFrontBarrier If the story has a
banister, whether to extend it a fraction of an aula's length
 * @returns An Object3D containing all hierarchy
 */
Story.prototype.makeStory = function(level, shouldHaveGlass,
isTopLevel, aulaCount, shouldHaveMagna, shouldHaveFrontBarrier,
shouldExtendFrontBarrier) {

    let aulaBase = new Aula().object3D;
    let aulaDimensions = (new
THREE.Box3().setFromObject(aulaBase)).getSize();

    let magnaBase = new Magna().object3D;
    let magnaDimensions = (new
THREE.Box3().setFromObject(magnaBase)).getSize();

    let aulaLength = aulaDimensions.x;
    let magnaDiameter = magnaDimensions.x;
    let magnaStairsX = 4;
    this.levelHeight = Math.min(aulaDimensions.y,
magnaDimensions.y);
```

```
    //Measures
    let walkwayWidth = Math.abs(aulaDimensions.z - magnaDiameter);
    let walkwayHeight = 0.3;
    this.levelLength = aulaCount * aulaLength + (shouldHaveMagna ?
magnaDiameter +  magnaStairsX: 0);
    let walkwayLength = this.levelLength - (shouldHaveMagna ?
magnaDiameter/ 2 : 0);

    let barrierY = aulaDimensions.y / 2;
    let barrierZ = 0.3

    let glassY = aulaDimensions.y - barrierY;
    let glassZ = barrierZ / 5;

    //Geometries
    let floorGeom = new THREE.BoxGeometry(walkwayLength,
walkwayHeight, walkwayWidth);
    let barrierGeom = new THREE.BoxGeometry(walkwayLength +
(shouldExtendFrontBarrier ? 2*aulaLength / 3 : 0), barrierY,
barrierZ);
    let glassGeom = new THREE.BoxGeometry(walkwayLength +
(shouldExtendFrontBarrier ? 2*aulaLength / 3 : 0), glassY,
glassZ);

    //Create hierarchy
    var storyObj = new THREE.Object3D();
    var currentX = walkwayLength/2;
    magnaBase.position.x = currentX;
    magnaBase.position.z -= walkwayWidth / 2 * .92;

    //if this floor should have an aula magna
    if (shouldHaveMagna) {
        storyObj.add(magnaBase);

        currentX -= magnaDiameter / 2 + magnaStairsX / 2;

        //make stairs to downwards if level > 1 && !isTopLevel
        if (level > 1 && !isTopLevel) {
            let stairs = makeUShapedStair(magnaStairsX,
this.levelHeight);
            storyObj.add(stairs);

            stairs.position.x = currentX;
            stairs.position.y -= this.levelHeight;
            stairs.scale.z *= -1;
```

```
            stairs.rotation.y += Math.PI;
        }
        currentX -= magnaStairsX / 2;
    }
    currentX -= aulaLength / 2;

    //Add aulas
    if (!isTopLevel) {
        for (var i = 0; i < aulaCount; i++) {
            let aula = aulaBase.clone();
            storyObj.add(aula);
            aula.position.x = currentX;
            currentX -= aulaLength;
        }

        //Floor
        let floor = new THREE.Mesh(floorGeom, this.rockMat);
        floor.position.z = walkwayWidth / 2;
        floor.position.y = -walkwayHeight / 2;
        storyObj.add(floor);
    }
    else {
        let bigFloorGeom = new THREE.BoxGeometry(walkwayLength,
walkwayHeight, magnaDiameter);
        var bigFloor = new THREE.Mesh(bigFloorGeom,
this.waterProofMat);
        bigFloor.position.y = -walkwayHeight / 2;
        bigFloor.position.z = magnaBase.position.z;
        storyObj.add(bigFloor);
    }

    //Create barriers if level > 1 on both sides
    if (level > 1) {
        let frontBarrier = new THREE.Mesh(barrierGeom,
this.concreteMat);
        frontBarrier.position.z = walkwayWidth - barrierZ / 2 +
.01;
        frontBarrier.position.x = shouldExtendFrontBarrier ?
-aulaLength / 3 : 0;

        let backBarrier = frontBarrier.clone();
        backBarrier.position.z = - aulaDimensions.z -
walkwayHeight / 2 - .01;

        if (shouldHaveFrontBarrier)
            storyObj.add(frontBarrier);
```

```
        storyObj.add(backBarrier);

        //Add glass if needed
        if (shouldHaveGlass) {
            let glassCover = new THREE.Mesh(glassGeom,
this.glassMat);
            storyObj.add(glassCover);
            glassCover.position.z = frontBarrier.position.z;
            glassCover.position.y = frontBarrier.position.y +
barrierY / 2 + glassY / 2;
            glassCover.position.x = shouldExtendFrontBarrier ?
-aulaLength / 3 : 0;

        }
    }
    return storyObj;
}
```

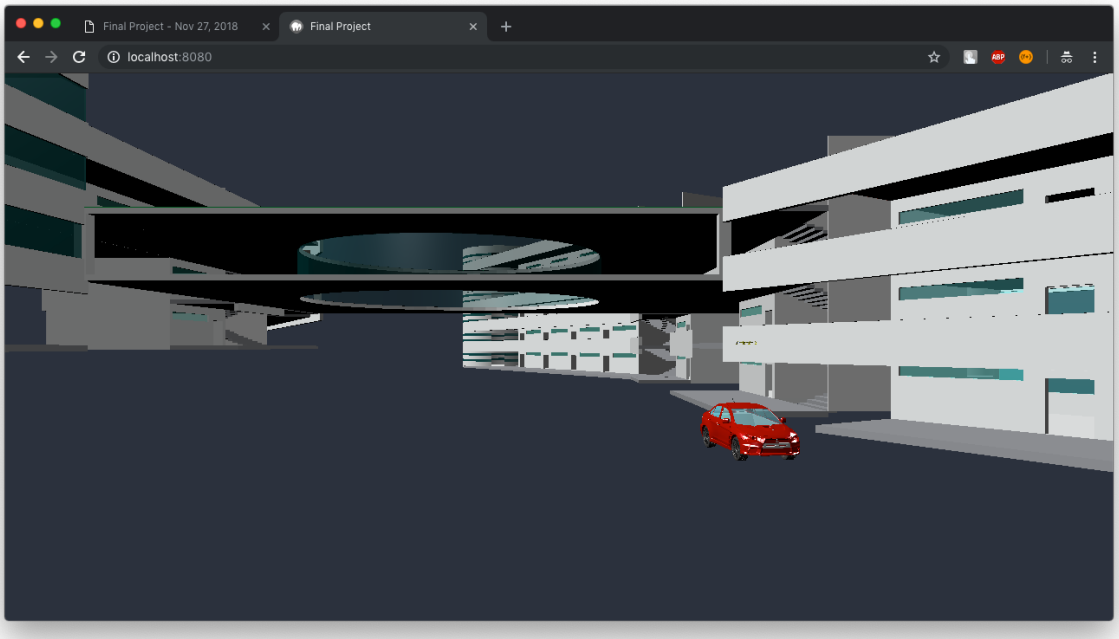
## util.js

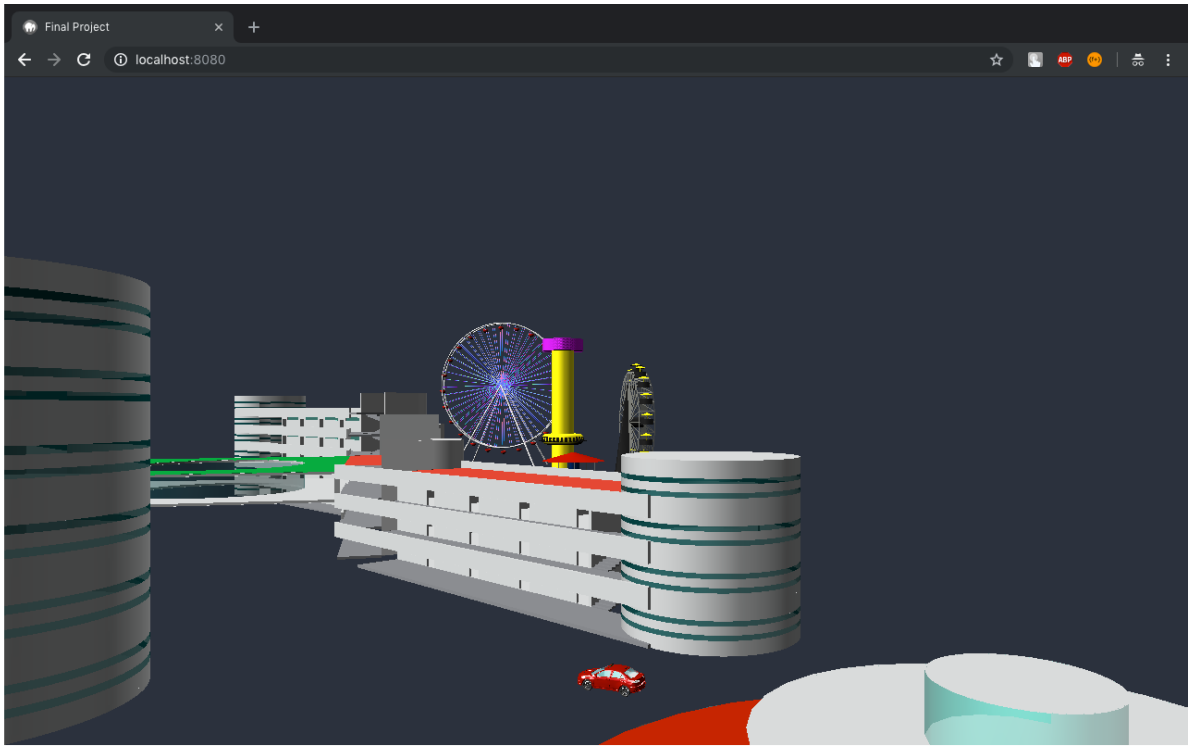
```
/*A01328937     Luis Francisco Flores Romero*/
/*A01324276     Rafael Antonio Comonfort Viveros*/
/*16.nov.2018*/

//   Convert degrees to radians
let toRad = (degrees) => {
     return degrees * Math.PI / 180;
};

//   Convert radians to degrees
let toDeg = (radians) => {
     return radians * 180 / Math.PI;
};
```

# Appendix B: Screenshots




Final Project
localhost:8080


Final Project
localhost:8080